UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CARLOS RANIERY PAULA SANTOS

Performance Management of IT Service Processes Using a Mashup-based Approach

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Dr. Lisandro Zambenedetti Granville
Advisor

Porto Alegre, November 2013

CIP – CATALOGING-IN-PUBLICATION

Santos, Carlos Raniery Paula

Performance Management of IT Service Processes Using a Mashup-based Approach / Carlos Raniery Paula Santos. – Porto Alegre: PPGC da UFRGS, 2013.

162 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2013. Advisor: Lisandro Zambenedetti Granville.

I. Granville, Lisandro Zambenedetti. II. Título.

“To succeed, people need a sense of self-efficacy, strung together with resilience to meet the inevitable obstacles and inequities of life.”

— ALBERT BANDURA

ACKNOWLEDGMENT

First, I sincerely thank my beloved parents and grandparents, who had toiled, sacrificed, and dedicated their lives for my progress. Words are simply not enough to express my profound feelings of love and gratitude for them. To my wife Emanuella, for being such a lovely person. It is my proud and privilege to live on her side.

I also want to thank so many good friends I've made during the last years. To Rafael Esteves, Giovane Moura, Jéferson Nobre, Guilherme Machado, and Flávio Santos for sharing good moments in the conferences. A special thank to Rafael Bezerra, who helped me in the development of this work.

Thinking back to all the years that I have been in research, I am very fortunate to have had the opportunity to interact with great minds and truly inspirational figures. I would like to acknowledge Profa. Rossana Andrade, Prof. Luciano Gaspary, and Profa. Liane Tarouco for making me a better researcher. I would also like to thank my thesis committee, for their feedback on my work and the many good questions they raised to make this a better thesis.

There are also some people at the IBM that I would like to mention. I record my deepest gratitude to Nikos Anerousis, who selflessly extended his incomparable help in various phases of this research and provided essential input to the formulation of my study. My sincere thanks to David Loewenstern, Winnie Cheng, and Larisa Shwartz whose knowledge in ITSM is par excellence and enabled me to find new vistas in the analytical study.

Finally, I want to make a special acknowledgment to my advisor, Prof. Lisandro Zambenedetti Granville, a brilliant computer scientist who I admire the most for his dedication to excellence in his work. Among all of his teachings, one is of special interest for me: be resilient. After so many hurdles in the last years, he always inspired me to keep moving forward. He had not only been a good advisor but a good friend.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

3PCC	3rd Party Call Control
API	Application Programming Interface
APOA	Assessed Proportion of Affect
AI	Acquire and Implement
AS	Autonomous System
BGP	Border Gateway Protocol
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMN	Business Process Model and Notation
BS	British Standards
BSI	BS Institution
C&C	Command and Control
CLI	Command Line Interface
CMMI	Capability Maturity Model Integration
COBIT	Control Objectives for Information and related Technology
CSI	Continual Service Improvement
CCTA	Central Computer and Telecommunications Agency
CT	Complex Ticket
CM	Complexity Model
DS	Deliver and Support
DDoS	Distributed-Denial-of-Service
EPC	Error Producing Condition
ETA	Event Tree Analysis
ETS	External Ticketing System
FTA	Fault Tree Analysis
GEMS	Generic Error Modeling System

GUI	Graphical User Interface
GTT	Generic Task Type
HEP	Human Error Probability
HRA	Human Reliability Assessment
HEART	Human Error Assessment and Reduction Technique
JEDHI	Justification of Human Error Data Information
IDE	Integrated Development Environment
IT	Information Technology
ITS	Internal Ticketing System
ITIL	IT Infrastructure Library
ITSM	IT Service Management
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
itSMF	IT Service Management Forum
ISACA	Information Systems Audit and Control Association
IFC	Information Flow Control
IMS	Information Management System
KLM	Keystroke-Level Model
LIFO	Last-In-First-Out
ME	Monitor and Evaluate
MLR	Multiple Linear Regression
MOF	Microsoft Operations Framework
NVE	Network Virtualization Environments
OGC	Office of Government Commerce
PDCA	Plan-Do-Check-Act
PMBOK	Project Management Body of Knowledge
PO	Plan and Organize
PSF	Performance Shaping Factor
QoS	Quality of Service
RSS	Rich Site Summary
RPC	Remote Procedure Call
RMSE	Root Mean Square Error
SA	System Administrator
SLA	Service Level Agreement

SLO	Service Level Objective
SME	Subject Matter Expert
SMS	Short Message Service
SOA	Software-Oriented Architecture
SOAP	Simple Object Access Protocol
SPD	Service Provider's Directory
SLIM	Success Likelihood Index Methodology
SRK	Skill, Rule, Knowledge
SIP	Session Initiation Protocol
ST	Simple Ticket
THERP	Technique for Human Error Rate Prediction
THETA	Top-down Human Error and Task Analysis
TQM	Total Quality Management

LIST OF SYMBOLS

$T(n)$	Sequence of n characters on a keyboard
$W(t)$	Waiting for the system to respond
T_total	Duration time of an IT process
x_i	IT management complexity metrics
β_i	Time for all factors not explained by the complexity model
T_a	Becoming aware time
T_k	Time derived from the mechanical execution
T_c	Time derived from the complexity in carrying on a task
Ap_i	Proportion assessment factor
σ	Standard deviation for the assessed proportion of affect
$var()$	Variance for the times predicted by the quantitative analysis
y	Measured time for dispatch process
$\widehat{y}$	Time predicted by the complexity model
R^2	Variability between predicted and real time measurements
r^2	Coefficient of determination

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Modern IT service provider organizations are under a continuous pressure to increase their competitiveness. Ways to reduce costs while improving performance – in terms of productivity and quality – of services are a key focus area for companies in the IT industry. However, despite all the solutions that have been proposed, modelling and optimizing human-centered processes remains a burdensome task. The human operator may be influenced by multiple factors and execute the process in a different way each time, thus introducing a significant variability in the final process outcome.

Although the research proposed so far have introduced improvements on service management, there are several challenges (*e.g.*, rapid change, budgetary constraints, skill shortages, system complexity, current and future user requirements, and growing user expectations) that are not completely covered by the current efforts. Among the new technologies available today, a set of novel ones, referred to as Web 2.0, has not yet been investigated by both industry and academia in the ITSM context. In the myriad of technologies and applications that define the Web 2.0, one is of special interest in this thesis: the *mashups*. Mashups are Web applications created through the composition of pre-existing Web resources (*e.g.*, interactive maps, Web services, traditional HTML pages, or even Flash presentations). Easiness of use, extensibility, and context specific development are examples of characteristics presented by mashups that candidate them to be a viable technology for improving IT Service Management.

Therefore, the goal of this thesis is to investigate the feasibility of mashups as an effective approach to improve performance (*i.e.*, in terms of productivity and reliability) of human-centered ITSM activities. In particular, this thesis aims to define management solutions required to deliver and demonstrate improvements in performance of human-centered ITSM processes. Specifically, the focus is on individual steps in the process that can be assessed through instrumentation or observation, and can be improved through design and automation. The analysis of exogenous events are not addressed, such as answering telephone calls or other interruptions.

The introduced management solutions are examined through a real case study, related to the Request Fulfillment process. The focus of this case study is on dispatch, an activity centered on human operators called dispatchers, with knowledge of standard fulfillment procedures. In this context, mashups are analyzed as an effective approach to cope with inefficiencies and errors introduced by human operators while performing their daily activities in the context of ITSM. An extensive analysis of tasks performed by a group of Subject Matter Experts (SMEs) in a global IT Service Support and Delivery organization was performed. The results demonstrate the viability of the mashups technology as a means for improving IT Service Management.

Keywords: IT Service Management, Mashups, Web 2.0, Performance Management,

Quantitative Analysis.

Gerenciamento de Desempenho e Modelagem Quantitativa de Processos de Gerência de TI Usando Mashups

RESUMO

Modernas provedoras de serviços de TI estão sob constante pressão para aumentar sua competitividade. Meios de reduzir os custos e aumentar o desempenho (*i.e.*, produtividade e qualidade) dos serviços oferecidos são temas centrais na indústria de TI. Contudo, apesar de todos os esforços feitos até hoje, modelar e otimizar processos de TI que envolvem operadores humanos continua sendo uma tarefa complexa. O humano pode ser influenciado por inúmeros fatores e executar o processo de uma forma diferente a cada vez, portanto, introduzindo uma significativa variabilidade no resultado final do processo.

Apesar de todas as pesquisas feitas até hoje terem introduzido melhorias significativas no gerenciamento de serviços, ainda restam muitos desafios (*e.g.*, mudanças rápidas, limites orçamentários, falta de conhecimento, complexidade dos sistemas, atuais e futuros requisitos dos usuários e finalmente aumento nas expectativas dos usuários) que não foram completamente resolvidos pelos esforços atuais. Dentre todas as tecnologias disponíveis atualmente, uma em especial, conhecida como Web 2.0, ainda não foi investigada pela indústria e pela acadêmia no contexto da gerência de serviços de TI. Dentre todas as tecnologias e aplicações que definem a Web 2.0, uma é de especial interesse nesta tese: os mashups. Mashups sao aplicacoes Web criadas a partir da composicao de recursos disponiveis online (e.g., mapas interativos, Web services, paginas HTML). Facilidade de uso, extensibilidade e desenvolvimento específico de contexto são exemplos de características apresentadas pelos mashups que os candidatam como uma solução viável para aprimorar o gerenciamento de serviços de TI.

Desta forma, o objetivo desta tese é investigar a aplicabilidade de mashups como uma solução efetiva para melhorar o desempenho (*i.e.*, em termos de produtividade e confiabilidade) de atividades de gerência de serviços de TI que envolvem operadores humanos. Em particular, esta tese tem como objetivo definir soluções de gerenciamento necessárias para melhorar e avaliar o desempenho de processos de TI. Especificamente, esta tese foca nos passos do processo que podem ser medidos através de observação ou intrumentação e que podem ser aprimorados através de projeto e automação.

As soluções de gerenciamento introduzidas são investigadas através de um estudo de caso real, relacionado ao processo de Cumprimento de Requisição. O foco deste estudo de caso é nos operadores humanos com a responsabilidade de receber as requisições dos clientes e encaminhá-las paras administradores de sistema responsáveis por resolver as requisições. Neste contexto, os mashups são investigados como uma solução efetiva para lidar com ineficiências e erros introduzidos pelos operadores humanos enquanto executam suas atividades diárias relacionadas à gerência de serviços de TI. Foi realizada uma extensa investigação das tarefas realizadas por um grupo de especialistas em um centro global de suporte e entrega de serviços de TI. Os resultados demonstram a viabilidade da tecnologia de mashups como solução para aprimorar a gerência de serviços de TI.

Palavras-chave: Gerenciamento de Serviços de TI, Mashups, Web 2.0, Gerenciamento de Desempenho, Modelagem Quantitativa.

1 INTRODUCTION

Over the last few years, the increasing complexity and importance of Information Technology (IT) environments have been leading researchers and practitioners from both academia and industry to pay more attention to the Information Technology Service Management (ITSM) area. ITSM encompasses the practices for managing large-scale IT infrastructures in order to provide services efficiently and in a cost-effective manner for IT customers. Along the years, several management frameworks and methodologies have then been conceived to help IT service providers towards a coherent service management experience. Examples of such frameworks and methodologies include Control Objectives for Information and related Technology (COBIT) (ISACA, 2008), IT Infrastructure Library (ITIL) (OGC, 2008), Microsoft's Operations Framework (PULTORAK; HENRY; LEENARDS, 2008), IBM's Systems Management Solutions Lifecycle, HP's IT Service Management Reference Model (DRAKE, 2000), and Code of Practice for IT Service Management (ISO/IEC 20000-2, 2005).

In ITSM, a very large percentage of the work is performed by humans, rather than machines. Because of its unpredictable nature, human behavior and performance are harder to model, and consequently, to optimize. In a service management operation, organized according to ITIL, the presence of humans in the critical path for performing work, while often necessary, increases variability in performance and service quality. Even if the nature of work is exactly the same, a human operator may execute it in a different way each time: he/she may use a different process; or different tools; or a different sequence of steps; or be interrupted a number of times by external factors such as a phone call or email messages. As a result, enforcing and obtaining tight performance bounds in a human-staffed organization is far more difficult than in a process executed by a machine.

The competitive nature of IT service provider organizations calls for a continuous improvement process: IT operators need to find ways to increase performance in terms of effectiveness, productivity, and quality. At the same time, operational costs are expected to be reduced. Diverse approaches have been proposed so far to meet such goals. Automation of IT operations has been of attention for the last two decades (MILLIKEN *et al.*, 1986), with on-going development of new technologies (CANDEA *et al.*, 2006) (YEMINI *et al.*, 1996) and dozens of automation related products on the market (KELLER *et al.*, 2004). Those efforts claim that ideally a system should be self-managed without any human intervention. Process optimization methodologies, on the other side, evolved from methods for optimizing automotive manufacturing (LIKER, 2004) and initiatives to eliminate defects in the semiconductor industry (PYZDEK, 2003). Basically, such optimization methodologies are based on the use of right skills levels, high volume workload, and reduced human interaction requirements (ANEROUSIS; DIAO; HECHING, 2010).

Although the research proposed so far have introduced improvements on service management, there are several challenges (*e.g.*, rapid change, budgetary constraints, skill shortages, system complexity, current and future user requirements, and growing user expectations) that are not completely covered by the current efforts. Among the new technologies available today, a set of novel ones, referred to as Web 2.0, has not yet been investigated by both industry and academy in the ITSM context. Web 2.0 designates a new kind of Web applications where users are motivated to actively create and organize contents available on the Web (O'REILLY, 2005). In the myriad of technologies and applications that define the Web 2.0, one is of special interest in this thesis: the *mashups*. Mashups are Web applications created through the composition of pre-existing Web resources (*e.g.*, interactive maps, Web services, traditional HTML pages, or even Flash presentations) (YU *et al.*, 2008). Despite their widespread usage, no effort towards the investigation of mashups for IT service management has been conducted so far, and the benefits and drawbacks of this approach remains unclear. Therefore, the goal of this thesis is to investigate the feasibility of mashups as an effective approach to improve performance (*i.e.*, in terms of productivity and reliability) of human-centered ITSM activities. In particular, this thesis aims to define management solutions required to deliver and demonstrate performance improvements in such activities. Specifically, the focus of the investigation is on individual steps that can be assessed through instrumentation or observation, and can be improved through design and automation. The analysis of exogenous events are not addressed, such as answering telephone calls or other interruptions. The hypothesis of this thesis is presented as follows.

***Hypothesis*: "The employment of mashups enhances the performance of human-centered ITSM processes."**

In order to guide the investigation of this thesis, **fundamental questions** associated with the hypothesis are defined and presented as follows:

- **What are the major causes for a poor performance in the execution of IT Service Management processes?**
- **What methods could be employed in the development of mashups aiming at performance enhancement?**
- **How models available in the literature can be used to assess performance improvements obtained with mashups?**

The investigation carried out during this thesis was conducted based on Six Sigma (PANDE; NEUMAN; CAVANAGH, 2000), which is a widely accepted management methodology for service quality improvement. Six Sigma focuses on understanding causes for process variability and driving to reach higher levels of consistency. The introduced solutions management solutions are examined through a real case study, related to the Request Fulfillment process. Specifically, the focus of this case study is on dispatch, an activity centered on human operators called dispatchers, with knowledge of standard fulfillment procedures.

1.1 Contributions

The investigation of mashups in the context of ITSM will led to the following major contributions.

- Identification of key performance problems that degrade the overall performance of human-centered ITSM processes and that can be improved through design and automation.
- Introduction of new methods to be employed in the development of mashups and that represent effective approaches to eliminate performance problems.
- Definition of quantitative models to assess the performance improvements obtained with the application of the mashups technology in ITSM processes.

1.2 Organization

The organization of this thesis is described as follows in the next section.

Chapter 2 first presents an overview of the ITSM field. The most widely accepted frameworks of best practices and performance improvement approaches are presented in details as well as their major limitations for employment. In a second moment, this chapter provides a survey about the mashups technology, discussing common applications and challenges for mashups in IT service provider organizations.

Chapter 3 provides an extensive study of performance management in human-centered ITSM processes. The analysis identifies key performance problems that may arise during the execution of ITSM processes, combines quantitative models to observe and quantify the impact of mashups when deployed over ITSM processes, and finally introduces a set of appropriate methods (*i.e.*, *Mashup Patterns* (MP) and error prevention modules) to solve root cause problems.

Chapter 4 demonstrates the application of mashups in the Request Fulfillment process. Based on interviews with a group of SMEs in a global IT Service Support and Delivery organization, it was possible to determine the workflow of activities performed during the dispatch assignment process. This Chapter also identifies the areas where performance improvement is possible through the usage of the mashups technology and presents a quantitative analysis of such improvement in dispatch.

Chapter 5 presents the final remarks and conclusions associated to this thesis. The answer for the fundamental questions are exposed and justified. In addition, opportunities to develop future works are identified and detailed.

2 STATE OF THE ART

The goal of this chapter is to present the background of the two research topics this thesis touches: IT Service Management and mashups. This chapter starts discussing the current investigations about the ITSM area. The most widely accepted frameworks of best practices are presented in details. Next, performance improvement approaches are discussed as well as their major limitations for employment. In a second moment, it is provided an overview of Web 2.0 technologies, focusing mainly in the applications (*i.e.*, mashups) created from the composition of heterogeneous resources. After, this chapter presents the current research status of this area, followed up by a discussion of the common applications and challenges for mashups in IT service organizations.

2.1 IT Service Management

Information Technology Service Management (ITSM) encompasses the practices for managing information technology systems and was originally introduced by the government of the United Kingdom, during a recession in the early 1980s. The government was forced to decrease costs and to improve the management of IT service Delivery. In this scenario, ITSM emerged as an attempt to provide stable, cost-effective, and agile service delivery capabilities for IT organizations. In essence, ITSM provides a framework to align IT operations with customer's business in order to improve it. Therefore, ITSM is centered on the delivery and support of quality IT services, rather than on technology *per se*.

During the last three decades a number of ITSM-related frameworks have been developed in order to help managers improve IT operations. Many of these frameworks have been proposed by companies, such as Microsoft's Operations Framework (MOF) (PULTORAK; HENRY; LEENARDS, 2008), IBM Systems Management Solutions Lifecycle (IBM, 2008), and the HP IT Service Management Reference Model (HP, 2008). These frameworks, however, provide their own approach to ITSM and focus on proprietary software of the organizations. At the same time, open best practice collections for ITSM – such as the Control Objectives for Information and related Technologies (COBIT) and the Information Technology Infrastructure Library (ITIL) (OGC, 2008) – are becoming popular, and are being employed in a diverse set of environments and scenarios (LACY; MACFARLANE, 2007).

From all the ITSM-related frameworks proposed so far, ITIL has become the most popular and influential. It was developed by the Britith government's Office of Government Commerce (OGC), formerly called Central Computer and Telecommunications Agency (CCTA), in 1980s and its current version (ITIL v3) consists of five publications

and associated tools. Its development influenced the creation of BS 15000 [1] in 2000 by the British Standards Institute (BSI). This ITSM standard was quickly accepted as an international standard, released in December 2005 as ISO/IEC 20000 [2].

Although they are integrated, ITIL and the ISO/IEC 20000 serve different purposes. ISO/IEC 20000, for example, does not specify how to design processes. It is rather a set of requirements which must be met in order to qualify for certification. ITIL, on the other hand, provides good practice guidelines, advice, and options that can be selectively adopted and adapted. In essence, ITIL should be the first set of guidelines to be followed in order to achieve ISO/IEC ISO 20000 certification. This evolution of ITSM from the ITIL framework to the BS 15000, and them to the international standard ISO/IEC 20000 reflects the changing global demands placed on IT organizations to deliver IT services. Figure 2.1, depicts this relationship between ISO/IEC 20000 and ITIL.

In addition to these frameworks, ITSM shares a common theme with the process improvement movement, such as Total Quality Management (TQM) (AHSEN, 1996) and SixSigma (PANDE; NEUMAN; CAVANAGH, 2000). This movement touches the issue of quality, *i.e.*, the frameworks, processes, and metrics that measure effectiveness from the point of view of the receiver of such services. The usage of one single proposal may not be enough for managing properly an IT infrastructure for service delivery. In order to provide a comprehensive model for evaluating and optimizing performance in human-centered ITSM processes, the most accepted frameworks and methodologies are examined in details in the next further sections.

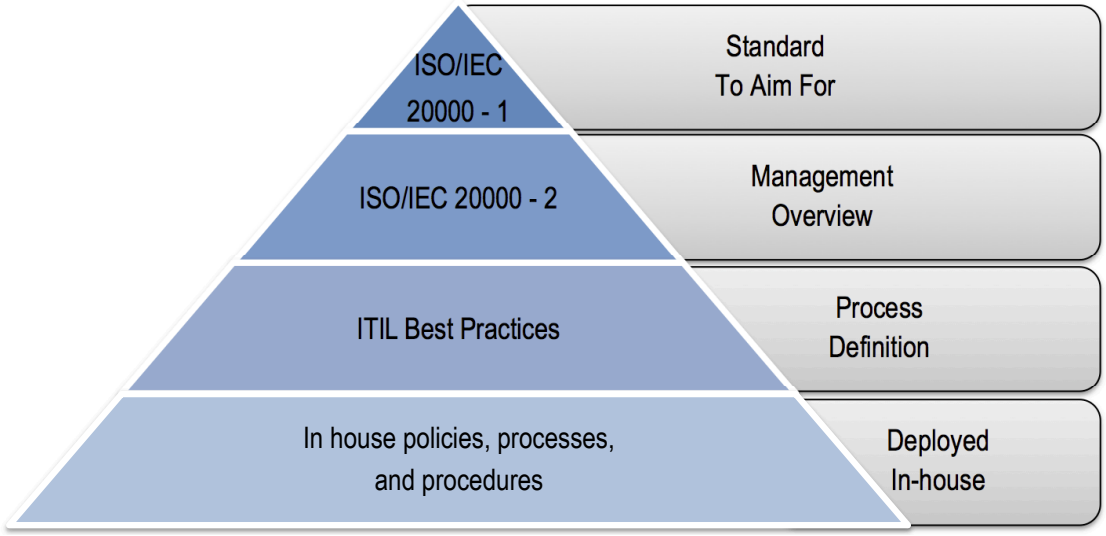

Figure 2.1: Relationship between ISO/IEC 20000 and ITIL

2.1.1 Information Technology Infrastructure Library (ITIL)

ITIL is a framework of best practices, based on a process-based approach, with the objective to improve the delivery of high quality IT services at a low cost. Before its creation, agencies and private sector contractors were independently creating their own IT management practices and duplicating efforts. The content of ITIL is independent of

[1] www.bs15000.org.uk
[2] www.isoiec20000certification.com

tools, vendors, or industry in which the service is executed, and can be applied to organizations of any size. However, it is not intended to be applied as-is, organizations are motivated to adapt it to meet their own business needs. As will be seem in Chapter 4, although the service delivery organization investigated during this study is organized according to ITIL best practices, specific activities are adapted to increase the overall process performance by reducing costs and waste.

According to ITIL, service management is a set of specialized organizational capabilities for providing value to customers in the form of services. The act of transforming resources into valuable services is at the core of service management. Without these capabilities, a service organization is merely a bundle of resources that by itself has relatively low intrinsic value for customers. However, ITIL considers service management as more than just a set of capabilities. It is also a professional practice supported by an extensive body of knowledge, experience, and skills (OGC, 2008).

ITIL also defines the distinction between functions and processes. Functions are specialized organizations with certain types of work and responsible for specific outcomes. Such organizations are self-contained, with all the necessary capabilities and resources available for their performance and outcomes. For example, the Service Desk is a function with the role to be the primary point of contact for customers when there is a service disruption. Processes, on the other hand, can be assumed as closed-loop systems, providing changes and transformations towards a specific goal and using feedback for self-reinforcing and self-corrective actions. Processes are measurable, has specific results delivered to customers, and respond to specific events. For example, the Event Management is a process responsible for monitoring all the events occurred throughout the IT infrastructure.

Up to the version 2, the ITIL focus was on processes, but since its version 3 the focus changed to business value. This change occurred as an attempt to strength the relationship between organization's business needs and operational IT processes. Version 3 also recognizes the value and applicability of other standards, such as COBIT and CMMI. The current ITIL structure is composed of two components: the ITIL Core, which provides best practices applicable to organizations of all sizes and types; and the ITIL Complementary Guidance, which comprises a complementary set of publications with guidance specific to industry sectors, operating models, and technology architectures. The ITIL Core is composed of five publications, which provide guidelines necessary for an integrated approach for service management. These publications are discussed below:

- **Service Strategy**: provides guidance on how to view service management not only as an organizational capability but as a strategic asset. It helps a service provider to decide on a strategy to serve customers. Starting from an evaluation of customer needs and the market place, the Service Strategy process determines which services the IT organization should offer and what capabilities needs to develop. Its ultimate goal is to make the IT organization think and act in a strategic manner. Some topics discussed in Service Strategy are: development of service markets, characteristics of internal and external provider types, service assets, service portfolio, and implementation of strategy through the Service Lifecycle;

- **Service Design**: has as objective to design and develop services and service management practices to be employed into the production environment. Such services should be designed with the business objectives in mind and considering the impact of changes. It is not centered only on new services, but also in the changes and im-

provements necessary to increase or maintain value to customers. Among the key topics in Service Design are: service catalogue, availability, capacity, continuity, and service level management;

- **Service Transition**: helps to manage and control the changes of IT services, that are implemented in the working environment of a company, in a coordinated way and ensuring their continuity. This publication provides guidance on how the requirements of Service Strategy encoded in Service Design are effectively realized in Service Operation while controlling risks of failure and disruption. Guidance is provided while the control of services is transferred between service providers and customers. It combines practices in change, configuration, asset, release, and deployment Management.

- **Service Operation**: embodies the practices in the management of the day-to-day operation of services to ensure they are delivered effectively and efficiently. This includes fulfilling user requests, resolving service failures, fixing problems, as well as carrying out routine operational tasks. Strategic objectives are ultimately realized through Service Operation, therefore making it a critical capability. Some topics covered by this book are: event, incident, problem, request, application, and technical management practices;

- **Continual Service Improvement** (CSI): aims to provide guidance in creating and maintaining value for customer through better design, transition, and operation of services. In this context, quality is a key point to achieve and maintain high levels of service provision. Thus, principles, practices, and methods from quality management, change management, and capability improvement are combined in order to learn from past successes and failures. Cost is also considered and should be consistent with the customer satisfaction. A closed-loop feedback system, based on the Plan-Do-Check-Act (PDCA) model, is established and capable of receiving inputs for improvements from any planning standpoint. Guidance on service measurement, demonstrating value with metrics, developing baselines and maturity assessments are among the key topics.

The Figure 2.2 bellow, depicts the ITIL service management process areas and where they fit in the ITIL version 3 books. Among all the processes defined by ITIL, one if of fundamental importance for this thesis, *i.e.*, the Request Fulfillment. This process deals with service requests and it also includes the service desk functions. The service desk is the central contact point between users and IT staff (CATER-STEEL, 2008). It is also the first place that customers contact when they have a problem or any request. If these requests and problems are not handled immediately this can cause trust issues between customer and the service provider. Service Desk activities include managing control, communication and promotion, and providing management information.

Due the global demands placed on IT organizations to deliver IT services, the ITIL become widely observed in the IT service industry. Diverse success cases around the world have evidenced the importance of properly managing IT infrastructures. For example, in 2000 the response time for resolving Web incidents at Cartepillar was 30 minutes, however this goal was meet only in 30% of the time. After implementing ITIL, Cartepillar has been able to hit this goal in 90% of the time. Other examples of success in implementing ITIL, include the Proctor & Gamble which saved $125 million, according to company officials.

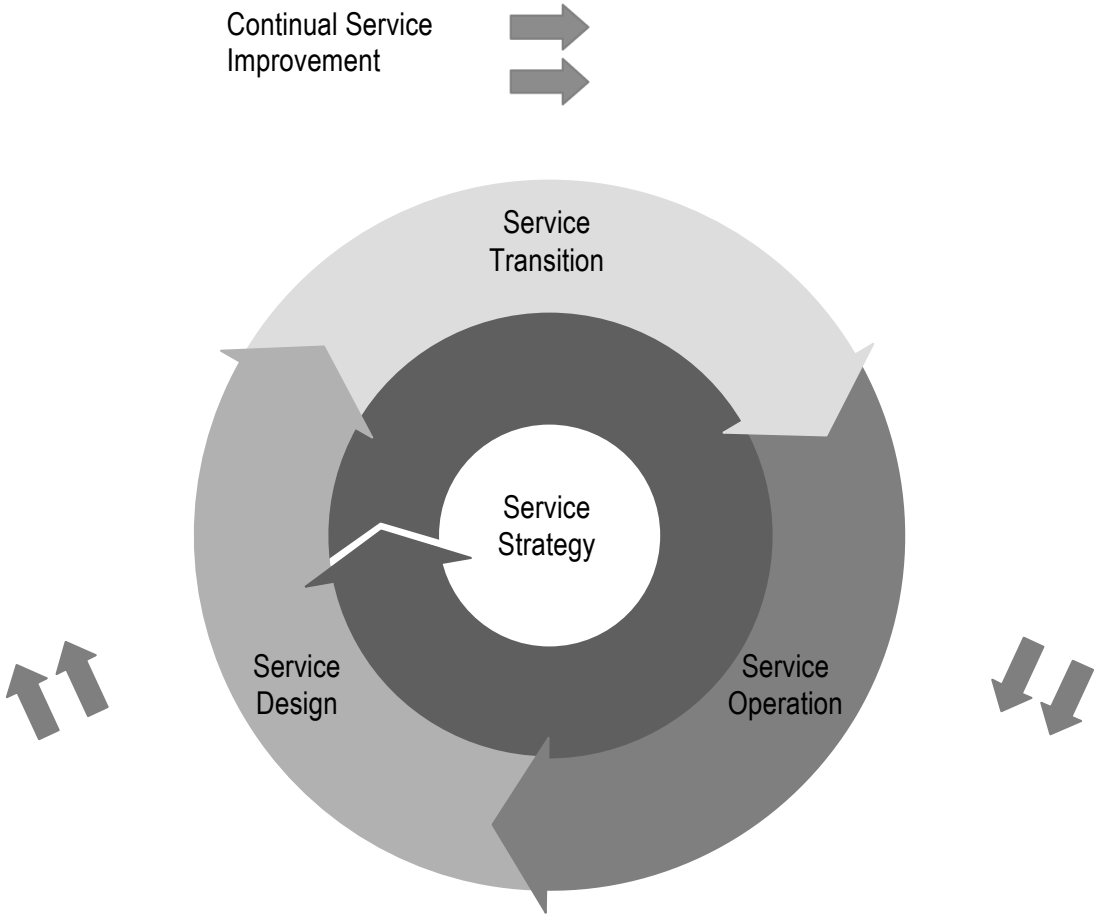

Figure 2.2: ITIL service management process areas – ITIL Core (OGC, 2008).

2.1.2 Performance Improvement Approaches

Performance Management is a discipline concerned with the definition of metrics and associated management solutions required to deliver and demonstrate improvements in performance, in terms of quality and productivity. Several researches (ANDERSON; BAGGETT; WIDENER, 2009) (BOLTON; LEMON; BRAMLETT, 2006) have demonstrated that variation in performance often affects both customer dissatisfaction and service delivery profitability. Additionally, process variation has a negative impact on the rate of learning of staff, which translates to lower productivity.

Multiple solutions have been proposed to decrease such variation, thus improving the performance of service delivery. These works are usually based on two complementary approaches, (1) automation and (2) system design, which are discussed as follows.

Automation. Automation is often used by the companies to obtain tight performance bounds and has been of attention for the last two decades (MILLIKEN *et al.*, 1986), with on-going development of new technologies (CANDEA *et al.*, 2006) (YEMINI *et al.*, 1996) and dozens of automation related products on the market (KELLER *et al.*, 2004). Despite all the definitions that can be found in the literature, most of the research efforts claim the same basic that, ideally a system should be self-managed without any human intervention.

Among the works proposed in this field, (BROWN; KELLER, 2006) and (KELLER

et al., 2004) introduced a methodology to analyze ITIL processes and to incrementally introduce automated service management implementations. (KAMINSKY *et al.*, 2008) proposes a policy-based self-managing system to optimize data center operations according to high-level business policies, thus addressing enterprise pressures in terms of cost and complexity. There also has been interest in process automation through workflow based solutions (BROWN; KELLER, 2006) (SHEN, 2008). Such workflow systems bring the additional advantage of easy integration of manual and automated activities within the same overall workflow, usually expressed in a general-purpose workflow language such as WS-BPEL (JURIC, 2006).

All these works have a common message, that automation provides a way to reduce labor costs and error rates as well as to increase the uniformity with which IT operations are performed. Although their benefits are evident, their application should be analyzed under a holistic view of the processes used to deliver IT services, since there may be hidden costs associated with its usage. When these extra costs exceed the benefits of automation, there happens a situation described as an *irony of automation* – a case where automation intended to reduce costs has ironically ended up increasing it (BAXTER *et al.*, 2012).

Before adopting an automation approach, it is essential to detect such hidden costs and weigh them against the benefits of automation. Although the automation advocators tend to focus on the automation infrastructure, and specifically on the variable costs (*e.g.*, per-instance costs) to promote their solutions, fixed costs – such as those related to the installation and maintenance of the infrastructure, adaptation of inputs to structured formats, and contingency to automation failures – should not be ignored when automating processes. This is specially true when processes are constantly changing, are too complex to be automated, or have a limited lifetime. For example, the request fulfillment process may be too complex to be automated due to the variability of the environment. Less skilled system administrators may need to undergo a temporary training, and so complex requests may be intentionally assigned to them in a "hands-on" approach to training. Finally, automation can also hide information essential for decision making. Blindly automating all tasks that are simple and intuitive can hurt IT operations in the long run, because a human operator can start to lose awareness of operational aspects of the IT environment (LEE; WONG; KIM, 2008).

System Design. System design optimization methodologies evolved from methods for optimizing automotive manufacturing (LIKER, 2004) and initiatives to eliminate defects in the semiconductor industry (PYZDEK, 2003). Basically, these works are based on a methodological approach for executing a quality improvement program, where the first step is the definition of key performance metrics for the environment under study. Then, such metrics should be used to observe and quantify the most relevant issues in the current environment (*i.e.*, baselining). The final step is to implement the appropriate mechanisms to attain control objectives (*e.g.*, reduce variability, reduce labor and penalty cost).

The work in improving the IT services space begins with two methodologies: Six Sigma (PANDE; NEUMAN; CAVANAGH, 2000) (PYZDEK, 2003) and Lean (KRAFCIK, 1988). Lean (sometimes called Lean Thinking or Lean Transformation) focuses on eliminating waste and delivering customer value in the shortest possible timescale. Six Sigma focuses on understanding variation (especially of

quality of product or service) in a process and driving to progressively higher levels of consistency. Lean Sigma (sometimes called Lean Six Sigma) (ALLEN, 2010) combines the two techniques to deliver increased quality at an increased speed and at a reduced cost. The fusion of Lean and Six Sigma improvement methods was required because Lean itself cannot bring a process under statistical control, and or reduce invested capital. The fusion of both achieves the maximum value. The core values of both methodologies can be seen on Figure 2.3.

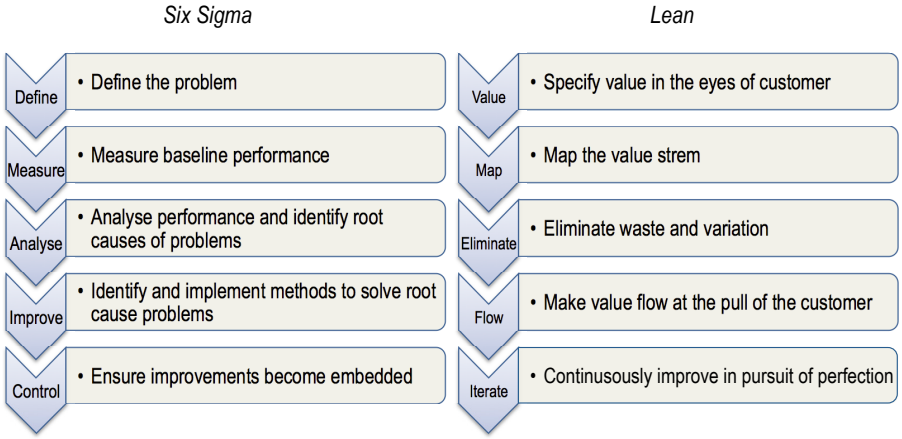

Figure 2.3: Lean and Six Sigma methodologies (ALLEN, 2010)

The application of Six Sigma in the services domain has been studied in (ANTONY, 2006) (JOHANNSEN; LEIST; ZELLNER, 2011) (ANTONY *et al.*, 2007), whereas (MCCLELLAN, 2008) (APTE; GOH, 2004) address the Lean framework. (BARASH; BARTOLINI; WU, 2007) presents an approach to improve service incident handling based on a set of performance metrics and a methodology inspired on Six Sigma for guided root cause analysis. In (ANEROUSIS; DIAO; HECHING, 2010) the authors adopt the Lean Sigma methodology as a quality control framework and propose an optimized system model for managing predictability and reducing cost in the IT incident management process.

Although a number of textbooks and tactical guides have been written on the topic, implementing Lean Sigma in a certain environment requires significant skill and expertise. Lean Sigma defines the principles of how to organize a process improvement project and provides guidelines for detecting waste and spotting opportunities for improvement. However, its successful implementation depends on the skill of the individuals entrusted with the execution, and the specific nature of the service delivery environment. In addition to using Lean Sigma principles, there is a need to resort to quantitative modeling, because the actual implementation of Lean depends on the real system characteristics, and a simulation model helps to implement and evaluate the system design optimization in a concrete way.

Another challenge in implementing Lean Sigma in a service environment is that service processes are subject to more noise or uncontrollable factors (*e.g.*, psychological factors, sociological factors, personal factors, etc) as compared to manufacturing processes. Another important remark is that the optimization of service processes is generally much more dependent on human and organizational change than on modifying manufacturing processes. Finally, the start-up cost for solidifying

Lean Sigma-based strategy into a corporate culture can be a significant investment, which might discourage many small and medium enterprises from the introduction, development and implementation of Lean Sigma strategy.

Considering such challenges in improving the performance of IT Service Management processes, mashups are an interesting technology that can take advantage of the current approaches by leveraging their relative merits. Again, the objective of mashups is not replacing current approaches, but to support the development of user-centric applications using disparate techniques. Therefore, next section reviews key concepts related to mashups technology, investigates the ITSM scenarios that most benefit from its implementation, and examines the challenges associated with its usage.

2.2 Mashups

Over the past few years, Web applications have taken on renewed interest both in industry and academia. These applications, termed as Web 2.0, are focused on the end-users and more importantly, encourage them to create their own applications. As a result, there is an increasing demand for principles that support re-use of content, collaborative sharing and compilation of content among users (O'REILLY, 2005) (MAJCHRZAK; MORE, 2011). These principles guide applications known as Mashups, which are, in turn, central to this thesis. Mashups have emerged as a class of Web applications that composes content from a variety of data sources such as Web pages, RSS feeds, Web services, and online APIs (MERRIL, 2006) (EDBERG; IVANOVA; JR., 2012).

From this definition, mashups can be considered as a specific implementation of Software-Oriented Architecture (SOA) within the Web framework. An important difference between traditional SOA applications and mashups is that third-party resources are not limited to services. Furthermore, mashups can be considered as a specialization of traditional Web applications, however, it is important to notice that mashups show some distinctive peculiarities. For example, the main objective of mashups is to graphically combine and integrate disparate assets (*e.g.*, presentation, data, and functionality) in new ways (WILSON *et al.*, 2012) (SCHOBESBERGER; CARTWRIGHT, 2013). Traditional Web applications, on the other hand, are characterized by consistent layout. Differences in the project size, duration time, costs, and development method are also evident when comparing both (KOSCHMIDER; HOYER; GIESSMANN, 2010) (TRIMBLE; DAYTON; HORN, 2012).

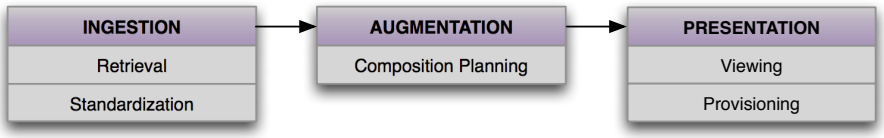

Figure 2.4: Mashup development methodology (JHINGRAN, 2006)

Typically, mashups are built following the methodology proposed by Jhingran *et. al* (JHINGRAN, 2006), presented in Figure 2.4. In this methodology, three main steps are defined: ingestion, augmentation, and presentation. During the ingestion step, data is gathered from disparate sources (*e.g.*, RSS Feeds, Web Services, and online APIs). Since these sources tend to be heterogeneous, wrappers are often used to standardize and retrieve the interested data. In the augmentation step, the user creates content defining

modules that: operate over the data, transform it, and generate more meaningful information from the aggregate. The result of this process is presented to the mashup creator and viewers in the presentation step. The presentation can range from a simple text file containing the results of the composition, to a complex and highly interactive map display.

Mashups can be ad-hoc programmed by a developer through traditional programming techniques, but they may also be defined by users with no programming expertise through *mashup systems*. Mashup systems are Web-based development tools that enable end-users to create their own mashups (CHOWDHURY *et al.*, 2013); they are analogue to what an Integrated Development Environment (IDE) is to a programming language. Usually, mashup systems perform three main tasks: provide users with visual tools to define new mashups, store created mashups in mashup repositories or libraries, and execute mashups when requested. Three tasks are often performed from a single mashup system, but it is possible for a mashup to be created in one system, executed in another, and stored in a third to form a remote mashup repository. Figure 2.5 depicts a reference architecture for mashup systems.

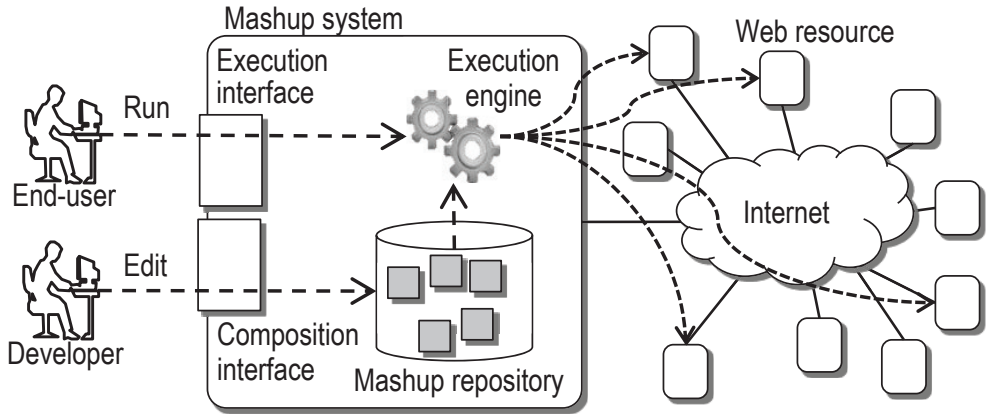

Figure 2.5: Reference architecture for mashup system (DOS SANTOS *et al.*, 2010)

Figure 2.5 shows the two most important actors related to mashups: the developer and the end-user. Developers define or edit compositions that result in the final mashups stored in a mashup repository. End-users then access the mashups that interest them. When a mashup is accessed from the repository, it runs on top of an execution engine to materialize the composition previously specified. In this process, the execution engine retrieves information from remotely located Web resources (using technologies like SOAP, RPC, and Syndication protocols), performs operations (*e.g.*, sorting, filtering, and aggregating) on the retrieved information, and finally builds an Web page that presents the combined results.

Although Figure 2.5 separates developers and end-users, both roles can be played by the same person. In fact, the possibility of unskilled end-users acting as developers is one of the key benefits advertised by mashup advocators. Whether such benefits really hold depends, among other factors, on the problem the end-user aims to solve and the capability of the mashup system to hide technical details without losing expressiveness in the composition definition. A third actor, not depicted on Figure 2.5, can be seen as the component developer, who is responsible for the technical details related to the mashup system. His/her duties include, for example, development and deployment of

new wrappers and mashup basic operators in the mashup system (DOS SANTOS *et al.*, 2010).

In order to allow users inexperienced in programming techniques to build their own applications, high-level abstractions are employed, in the form of operators. These operators hide how the original Web resources are orchestrated from mashup developers, so that developers can describe the logic of their compositions without being aware of technical details such as underlying network protocols and multiple data parsers. The coupling of such mashup operators can be guided in several ways. For instance, compatibility rules and quality criteria can be used to suggest the most appropriate operators for a given one.

To achieve that, a set of interaction elements is introduced (DOS SANTOS *et al.*, 2010). In the Composition Interface, mashup operators are represented by interaction elements. These elements are available as visual boxes to developers so that they can define the management logic behind a mashup. Seven types of interaction elements are presented bellow:

- **Visual**: supports different ways of displaying (*e.g.*, map, table, tree) the results of a composition;
- **Control**: represents basic programming logic, such as loops and conditions;
- **Transform**: manipulates retrieved data, like filtering, merging, aggregating, arithmetic operations, and boolean logic operations;
- **Adaptation**: translates original data from Web resources into formats more easily handled inside mashups
- **Input**: allows end users to feed mashups with their particular information, for example, through text fields in Web forms or by uploading files;
- **Execute**: triggers the asynchronous background execution of actions without the explicit request of the end user, which is typical in background monitoring systems and similar applications;
- **Reuse**: allows users to extend the available mashups to build more sophisticated compositions.

The access and representation of external information by mashup systems could be further enhanced by the more recent concept of Web 3.0. This refers to a semantic Web (FENSEL *et al.*, 2005), where both information and services have meta-information to describe them, enabling data to be processed by both humans and computer software. The availability of meta-information could allow mashup systems to use more dynamic methods of retrieving and representing external information (LE-PHUOC *et al.*, 2009) (SIRIN; HENDLER; PARSIA, 2002). However, Web 3.0 models and concepts are still in the early stages of definition and are not ready to be effectively employed in mashup systems as of today, thus will not be discussed in this thesis.

2.2.1 Literature Review

The literature has been presenting an increasing number of proposals related to the development and deployment of mashup-based solutions. In this section, the most significant researches are split into two separate categories: consumer mashups and enterprise mashups (NESTLER, 2008).

Consumer Mashups. Sometimes referred to as presentation mashups (BHATTARAI; ZHAO; CRESPI, 2010) (FUNG *et al.*, 2012), they are aimed at the general public and consist on the combination of different resources available on public Web sites, such as online APIs and feeds. The result is presented through a simple browser user interface, usually created by one of the sites participating in the mashup. A quintessential example of a consumer mashup is *HousingMaps.com*, which displays the available houses in an area by combining a listing from Craigslist.com with a display map from Google. It is possible to find consumer mashup-based solutions in a very large range of Web applications. There are several proposals regarding tools to enable end-user to create these consumer mashups. Details regarding such proposals are described below:

- Baresi and Guine (BARESI; GUINEA, 2010) developed a service composition framework called Mashlight that extends the idea of mashups to support the combination of disparate widgets by non-technical users. In their approach services are assumed to be widgets specially designed as autonomous entities that can collaborate by exchanging data. The authors developed a set of tools based on common client-side Web technologies. These tools are responsible, at design time, to support the creation of new widgets and for aggregating them, and at runtime, to provide interfaces for traditional browsers (*e.g.*, Safari and Firefox) and for iOS and Android mobile phones. They use a process-based composition style, where the composition is modeled using directed graphs: nodes represent the widgets to be composed, while arcs represent the data flows. As an addition to the common widget model, they proposed a new abstraction called *super-widget*, which acts as a container for widgets that are activated in parallel. The authors advocate that notions of control and data flows are too complex for the average non-technical user. To solve that, control-flow templates are used to semi-automatically create processes based on the user's needs. The following three templates are considered in the work:

  1. *Sequential*: creates a sequence in which the widgets are activated one at a time;
  2. *Container*: creates a single super-widget that contains all the widgets being composed;
  3. *Star*: creates a star-like topology in which every widget can be accessed from a central core widget that acts as a dictionary.

- Ennals and Garofalakis (ENNALS; GAROFALAKIS, 2007) proposed a tool, called MashMaker, for editing, querying, manipulating, and visualizing live semi-structured data. The main goal of MashMaker is to allow even non-expert users to easily create their own mashups based on data and queries produced by other users. The following principles form the basis of MashMaker:

  1. *Untyped Tree Data Model:* data is structured as a tree, where nodes have content, child nodes, and properties;
  2. *Mixed Data and Queries:* users can extend/enhance their view of the data by adding computed nodes whose values are computed based on the surrounding data;

3. *Sharing Queries as Widgets:* queries can be bundled and exported as widgets to be shared with friends;
4. *Overlayed Editing of Live Data:* users can apply local edits to their data views;
5. *Example-Driven Queries based on Interactive Data Exploration:* all queries and data manipulation tasks can be formulated through interactive data browsing and exploration;
6. *Collaborative Exploration of Data:* users can share data and widgets with friends and other members of their social network.

The authors also state that their work differs from previous ones through its focus on ad-hoc, interactive data exploration and manipulations rather than structure extraction and support for just querying the data. It also supports live data and query results, ability to package up queries as widgets, and its exploration-based, formula free query model.

- Liu et al. developed an approach (LIU *et al.*, 2007) (LIU; HUANG; MEI, 2007) (LIU; HUANG; MEI, 2008) to support the implementation of mashups through Service Oriented Architecture (SOA) technologies. In their approach services are assumed to be published and tagged with semantic information in the form of service description tags. Based on these tags they proposed an approach to search the existing services' tags and based on it present a "cloud tag" to the users. Once the user selects an specific service, the system proposes other services that can be composed together. Additionally, quality of service (QoS) information, helps the user to chose an specific service. Their architecture is divided in three levels:

  1. *Service Layer:* where a crawler is used to create the service repository and a service analyzer to mine existing services and service tags, and present matchings given the user decisions;
  2. *Knowledge Layer:* where the cloud tag is created, the service advisor presents suggestions to the user, and the composition graph is managed;
  3. *User Layer:* where the user interacts with the service composition mashup, receives suggestions from the service advisor, manages and executes the composition.

  The authors assume two distinct ways to create cloud tags:

  1. *Top-down:* by creating ontologies used to describe services semantically. They claim that this approach may be too complex, and difficult to maintain;
  2. *Bottom-up:* by employing techniques used in social networking websites, which provide user-based lightweight semantic annotations (*e.g.*, by using tags, folksonomies and simple taxonomies). They claim that this approach will allow the creation of "service communities", which promote self-organization and possibly automatic semantic annotation of services.

Enterprise mashups. In the industry, an important application of mashups consists on enterprise mashup systems (HOYER *et al.*, 2008) (GEBHARDT *et al.*, 2012). The main objective of these systems is to allow that personnel from a company to be able

to rapidly deliver products known as enterprise mashups (HOYER; STANOEVSKA-SLABEVA, 2009). These mashups can integrate data from both internal and external resources in order to solve operational problems through the development of specific applications. Examples of research efforts regarding enterprise mashups are presented below.

- Banerjee *et al.* (BANERJEE; DASGUPTA; MUKHERJEA, 2007) proposed a mashup architecture to support the combination of content (*e.g.*, call control, presence, and messaging) already available in the network of Telecom operators in order to provide new value-added functions to the users. The authors assume that the traditional combination of data from different applications usually employed in the mashups development is not sufficient to bring telecom services to the Web 2.0 domain. They argue that the Web 2.0 model blurs the line between software and services, making complicated to provide complex services such as telecommunication services. In order to enable the development of rich user-interactive services, this work claims that it's necessary to run a mediation platform between the mashup layer and the service control layer in the IP Multimedia Subsystem (IMS). Their proposed multi-layered architecture consists of:

  1. *Resource tier*: all the network elements that provide control to various network level services such as call, location, presence, SMS;
  2. *Service tier*: provides a service abstraction of network level services;
  3. *Client tier*: provides the tools necessary for developing the mashup applications using resources from the service tier;
  4. *Mashup tier*: consists of the mashup applications developed with support of the underlying tiers.

  In order to evaluate the proposed architecture, the authors developed an initial prototype of a *call-a-cab application*, which is based in three main components: (i) Google Map, (ii) BusinessFinder matchmaking service, and (iii) SIP-based 3rd Party Call Control (3PCC) Service. Although they refer that the development is supported by non-expert users, the proposed architecture is tailored to a single mashup that can not be used in other telecommunication scenarios.

- Enterprise mashups may be required to handle and display sensitive data that require well-defined disclosure policies. Protecting sensitive data in a highly collaborative and dynamic mashup environment poses several non-trivial questions. To deal with this issue, Santos *et al.* (SANTOS *et al.*, 2011) proposes a solution centered on an enhanced composition programming model that specifically captures and enforces disclosure policies. This way, data protection mechanisms are built directly into these applications. The model should be easy to understand, sufficiently expressive, and able to represent broad categories of security concerns among a wide range of mashup applications. To satisfy such requirements, the authors designed an architecture, called Maestro, based on the concept of Information Flow Control (IFC) (CHENG *et al.*, 2012), which is a security model that tracks the flow of sensitive data, enables the specification of disclosure policies and enforces them. This architecture centers around three key concepts: principals, tags, and labels. Principals

represent entities with an interest in security, such as individuals or network providers. Tags provide a way for principals to categorize their information. Labels are sets of tags and are used to enforce information flow. The work was validated through two realistic network management scenarios where multiple information sources are aggregated to create new applications for specific purposes: quality management and traffic monitoring. These scenarios have specific constraints on data disclosure and Maestro was able to capture and enforce these constraints, validating the value of the proposed secure mashup development methodology. In addition, the usage of visual interaction elements provided an ease-of-use development environment with little compromise on usability.

- Zou and Pavlovsk (ZOU; PAVLOVSKI, 2007) investigated the issue of accountability. The goal of their work is to define an accountability framework for mashup services. They consider that accountability issues in mashup services are a broader and more complex theme when compared to non-repudiation in an eCommerce transaction. They have proposed a framework that includes the service or content creator as well as the new owner of the resulting mashed-up service. They state that the first step towards enabling accountability in mashup services is to add more disclosure and trust. This includes identities of all the parties involved and traceability in service composition. They also suggest that the concepts of involved part roles are essential in the service ontology model. Their architecture is proposed as a meta-model and ontology for modeling solutions in accountability for the mashup domain. The meta-model focuses on the roles and responsibilities from an information system perspective and is intended for IT developers. The ontology, in turn, focuses upon the liabilities and agreements aspects of the mashup definition, which are useful to establish the contractual terms and definition between the respective parties.

2.2.2 Common Applications of Mashups in IT Operations

Are many the key benefits advertised by mashup advocators – such as easy-of-use, extensibility, and context specific development – and that candidate mashups to be employed in multiple scenarios for IT operations. Some of those scenarios are described as follows:

Situational Needs: One of the main characteristics of mashups is the possibility to be created easily and quickly, through the integration of already existing resources. This characteristic makes them an adequate choice to fulfill situational needs, often found in IT Service Organizations. This resembles the concept of situational applications (MOHAMMADI; KHALILI; ASHOORI, 2009), which are small, disposable pieces of software created in an ad-hoc fashion to meet a transient need. The traditional development paradigms are not suited for that kind of scenario because they require too much time and expertise to build even a simple, ad-hoc system. In this context, mashups are an interesting approach to reduce costs of the IT companies by allowing employees themselves to provide their own solutions to meet unique needs.

Perpetual Beta: Mashups are, by definition, easily modifiable (MERRIL, 2006), allowing the creation of entire new and more sophisticated applications through the

modification and/or reuse of existing compositions. This enables mashups to be employed in the perpetual beta applications. Such applications are always being changed, evolving, and being re-adapted to new requirements. The main ideas behind the perpetual beta principle are: there is never a final version and every iteration of the software is stable to be used in a production environment. In contrast to the classical software construction paradigm (*i.e.*, requirements, design, implementation, and test), mashups are always undergoing changes and improvements based on users feedback, supporting a perpetual beta development cycle.

Long-tail Applications: Current service organizations are operating in a dynamic environment and are frequently restructuring internal processes and organizational structures to meet external challenges and opportunities. The needs of IT employees for IT support are constantly evolving as well. As a result, a phenomenon called the long tail can be observed with respect to the requirements of such employees for new services and functions (HOYER *et al.*, 2008). Even with the development of sophisticated and complex systems, there is a large percentage of situations that can not be tackled appropriately by general purpose tools (*e.g.*, traditional management systems). Therefore, a mashup-based approach is well suited to cover this long-tail of daily user needs and to provide individual and heterogeneous enterprise applications in a shorter time.

Cooperation and Re-use: The ability of being executed by other systems, allows mashups to be shared to other interested users, that may re-use them and adapt to the new environment. In fact, one composition can be created by many sub parts, which themselves are mashups, and are created by different users, that may cooperate around the same goal. Furthermore, the modular nature of mashups encourages reuse and sharing of modules between different users, enabling the building of more sophisticated applications. In IT service provider companies, several problems can share a similar structure. In these cases, a standardized set of modification templates (*i.e.*, mashup patterns) can be applied as proven and reusable solutions, thus providing an additional level of stability.

2.2.3 Challenges in Applying Mashups in Service Industries

There are various challenges when applying mashups in the service sector, such as facing data collection issues and dealing with highly dynamic processes. Some of the potential difficulties and challenges observed from the published literature are presented below:

Data Heterogeneity. To create a mashup an user can retrieve just one type of external data (*i.e.*, homogeneous data), or he/she can also merge information represented in two or more data formats (*i.e.*, heterogeneous data). The composition of homogeneous data is the integration of similar types of media and information from multiple sources into a single representation (ALTINEL *et al.*, 2007). Heterogeneous composition, on the other hand, is defined by the combination of different data types, which are generally visual elements and data from multiple sources. A common way to deal with data heterogeneity is through the use of wrappers. Such wrappers are software elements that act as gateways, translating and normalizing external information into a representation that can be handled by a composition

engine, during the augmentation process. On the other hand, the creation of compositions based only on homogeneous information is less bureaucratic. The process of retrieving such information is usually coded directly on the composition engine. For example, mashups could be created to integrate information just from data feeds (*e.g.*, RSS, ATOM) or just from spreadsheets. In this kind of composition, the complexity of the mashup system is lower, since it is not necessary to retrieve and adapt many and different kinds of data.

Data Structure. Currently, there is a large amount of available data that could be employed in mashups for most different scenarios. Such data, however, differ from each other on how they are structured. There are three approaches to data organization: structured, semi-structured, and unstructured. Structured data refers to data that is identifiable because it is organized in a structure. Usually, structured data is decomposed in semantic entities and similar entities are grouped together into relations (BERGMAN, 2007). Entities in the same group have the same descriptions (*i.e.*, attributes), and descriptions for all entities in a group (*i.e.*, schema) have the same defined format, are all present, and follow the same order. Unstructured data, on the other hand, is data that can be of any type, not necessarily follows any format or sequence, does not follow any rules, and is not predictable (SIMMEN *et al.*, 2009). Examples of unstructured data include general text, video, sound and images. The last kind of data is the semi-structured (AUMüLLER; RAHM, 2007), which is also organized in semantic entities, with similar entities grouped together. However entities in the same group may not have same attributes, the order of attributes is not necessarily important, not all attributes may be required, the size of same attributes in a group may differ, and the types of similar attributes in a group may differ. Such data can be found, for example, in database systems, file systems, bibliographic data, or exchange formats.

Data Dynamicity. One key point of mashup creation is the capability to retrieve and adapt third party data. However, such data might be dynamic, imposing some issues for the mashups creators. Dinamicity, in this scenario, refers to the possibility of change on data structure, access methods, and results for the same queries. The information providers can for example, alter the way of how information is organized and accessed without any notice. In this case, static wrappers are inadequate, since they have to be replaced to allow mashups continue retrieving external data. In this scenario, the dependability of the composition relies on third party data providers. Without mechanisms to detect changes on the external data, mahups can stop working properly, or at least start providing misleading information. To deal with these issues, rollback mechanisms can be employed to allow the definition of atomic transactions in the composition logic. The creator of mashups should be able to explicitly define if some external information is not controlled and if a set of combined components need to be run in a guaranteed way. This allows compositions to flow in a different logical path in case of data access errors. To handle the issue of similar queries providing different results, versioning control mechanisms can be employed. With this control, will be possible to identify changes on the results provided by the same queries and so, define if the new information is still useful for the mashup end-user.

2.3 Summary

This chapter presented an overview of the state of the art regarding to IT service management and mashup areas. Despite its importance, ITIL and other widely accepted frameworks of best practices only provide high-level generic guidelines to IT service organizations, without proposing, for example, concrete models and methods for capturing metrics and evaluating the quality of IT processes. Such evaluation is important for the IT service providers to quantify, measure, and most importantly to predict the deployment impact of IT solutions. The scientific community has worked to propose solutions to fill this gap. The most significant works will be examined in details in the following Chapter. Through mashups, human operators can create their own specialized tools. Such mashups can also be reused by different operators to build more sophisticated applications in a cooperative fashion. The development of mashups occurs through mashup systems, which employ high-level abstractions and usability-oriented interfaces to hide technical details from operators and allow them to integrate disparate information. It is important to observe that mashup-based systems does not replace traditional models and tools; rather it allows traditional technologies to be more easily used to create new applications, pushing such technologies beyond their original purposes

3 PERFORMANCE MANAGEMENT OF IT SERVICE PROCESSES

This chapter provides an extensive study of performance management in human-centered ITSM processes. Exogenous events are not addressed, such as answering telephone calls or other interruptions. Rather, the focus is on individual steps in the process that can be assessed through instrumentation or observation, and can be improved through design and automation. In particular, it is carried out an analysis of tasks performed by a group of Subject Matter Experts (SMEs) in a global IT Service Support and Delivery organization. The analysis aims to identify problems in the sequence of events performed by human operators during the execution of ITSM processes and identifies the areas where performance improvement is possible. This chapter also introduces the concept of mashup patterns along with a new type of mashup basic operators, both representing effective approaches for eliminating defects in ITSM processes.

3.1 Performance Improvement

The competitive nature of IT service provider organizations calls for a continuous improvement process. Ways to reduce costs while improving performance and quality of services are a key focus area for companies in the IT industry. A significant body of work in the IT Service Management (ITSM) field addresses the issue of quality, *i.e.*, frameworks, processes and metrics that evaluate effectiveness from the point of view of the receiver of such services. This thesis, on the other hand, focuses on the service provider's perspective, particularly aspects of performance that have direct implications on the efficiency and cost of the operation from the provider's point of view. Although different metrics may be used to assess performance, this thesis concentrates on productivity and reliability. These metrics are widely used by IT organizations (HARRIS; HERRON; IWANICKI, 2008) because of their direct implications on the overall performance of ITSM processes. For example, less time deploying, maintaining, and correcting system issues translates to lower operational costs. Depending on the type of business, time savings can also result in greater revenue and/or greater value to the business; faster service deployment leads to improved customer satisfaction levels. Reliability, like productivity, is also a key performance indicator. For instance, having an improper configuration on a critical application can result in performance degradation or downtime for customers, leading to loss of revenue, credibility, and/or funding for IT departments. Mitigating failures enable customers and end-users to experience the service level they expect, thus helping IT to reduce the time and resources spent "firefighing" and resolving environment failures.

In ITSM, a very large percentage of the work is performed by humans, rather than

machines. Due to its unpredictable nature, human behavior and performance are much harder to model, and consequently, to optimize. Considering a modern data network, where packets are received at an entry point and needs to be transferred to a destination, the data packet in its path will be processed by a variety of system elements, each programmed to perform a specific task with a high amount of accuracy and predictability. The number of events (exceptions) that can interrupt a normal processing path can be large, but are always finite, and in many cases can be accounted for in the design itself through redundancy and error handling programs. By contrast, considering a service management operation organized according to the Information Technology Infrastructure Library (ITIL) standards, the presence of humans in the critical path for performing work introduces significant variability in the final outcome. Even if the nature of work is exactly the same, a human operator may execute it in a different way each time: the human operator may use a different process; or different tools; or a different sequence of steps; or be interrupted a number of times by external factors such as a telephone call or email. Enforcing and obtaining tight performance bounds in a human-staffed organization is far more difficult than in a process executed by a machine.

3.2 Productivity

ITSM processes can be laden with segments of the process where the human becomes a bottleneck and slows down the entire process. These inefficiencies are usually caused by insufficient design of the process itself, or defects in the tools being used. Moreover, these inefficiencies also present direct implications on the productivity, which is essentially defined as units of work performed per unit of time. However, as it will be seen in the next sections, even with rigorous definitions, productivity can be evaluated in many ways and at different levels of granularity.

3.2.1 Inefficiencies in Service Management Processes

Inefficiencies are portions of a service management process characterized by suboptimal execution of activities. This thesis concentrates on inefficiencies characterized as segments of the process where suboptimal human productivity reduces overall throughput for the process. Inefficiencies can appear at fundamentally different levels of analysis. For the purpose of analyzing inefficiencies within ITSM two levels are analyzed: higher level inefficiencies due to the complexity of the activity itself, and lower level inefficiencies due to the mechanical execution involved in performing the activity. As an example of a potential complexity inefficiency, in a process with many decision points, the operator needs to spend time determining the correct choice. In ITSM the lower level takes into consideration the interaction of human operators with the available software tools. For example, a Web application created with poor usability can impose a significant amount of wasted time for the operator due to added mouse-clicks and keystrokes required to retrieve, create or update information.

In order to discover common inefficiencies, descriptions of the tasks, performed by a group of human operators involved in a common activity (discussed in Chapter 4), were collected in a global service delivery organization. Based on such descriptions, the processes were recreated in a form of sequence of tasks (*i.e.*, workflows), which were later validated by the operators. Then, these workflows were drilled down to subtasks representing individual actions performed by the human operators when interacting with the systems. With this detailed process, it was possible to analyze both levels of inefficiencies.

The non-exhaustive list below represents several groups of inefficiencies found during the investigations:

Basic Inefficiencies. Are associated with the most simple and low-level inefficiencies, occurring independently from the others.

- *Context-Switching*: the operator needs to switch to different application from the one he/she is currently working on;
- *Locating Data*: after reaching the place where the information is available, the operator needs to search for the specific data across the screen and;
- *Entering Data*: the operator has to input data manually in the screen he/she is working on.

Information Management Inefficiencies: Are inefficiencies formed by the combination of several basic ones.

- *Copy/Paste*: manual copying of data from one system to another;
- *Consistency checks*: the operator needs to guarantee that information is consistent in different places and;
- *Information Lookups*: navigate between multiple screens to assemble information.

Skill-dependent Inefficiencies: Relate to the reasoning capabilities or training of the human operator.

- *Retaining Information*: remembering information for a subsequent step;
- *Combining Information*: all the data is in one screen and the operator needs to extract their meaning and;
- *Data transformation*: the data require some simple manual processing (*e.g.*, reformatting dates to a local format) when transferring from one screen to another.

Synchronization Inefficiencies: Are those incurring delays due to factors such as waiting for an external input.

- *Contacting a Person*: the operator needs to talk to someone by e-mail, instant messenger or in person and;
- *Becoming aware*: the operator needs to access a tool repetitively to be aware of new service requests.

Generally, it can be noted that synchronization, information-management, and skill-dependent inefficiencies typically include both high-level and low-level components. For example, converting a time from one time zone to another can represent "information management", and thus include inefficiencies that are both lower-level (*e.g.*, reading and typing times) and higher-level (*e.g.*, figuring out the appropriate time zone and doing the calculation).

3.2.2 Mashup Patterns for ITSM Inefficiencies

By using mashup basic operators (DOS SANTOS *et al.*, 2010), a user is able to specify mashups for a variety of different purposes. Where mashups share similar logic, however, it is often convenient to consider the employment of *Mashup Patterns* (MP) (OGRINZ, 2009). In the context of this thesis, these patterns describe the core of solutions to recurring problems that can be frequently found in IT service management activities. Their usage allow mashups to be instantiated even more quickly by reducing mashups design and modeling. In addition, because patterns enable previously proven mashups to be reused in new scenarios, mashups patterns provide an additional level of stability.

The service delivery organization investigated during this study provides the guidelines that human operators should follow in order to achieve a coherent service management experience. Based on the analysis of these guidelines, it was possible to identify a set of four candidate mashup patterns to tackle the previously depicted inefficiencies in service management processes. These patterns are: *Alerter*, *Importer*, *Transform*, and *Displayer*. They are described using a consistent format divided into the following sections: a *pattern name*, a *description*, an illustrative *example*, a description of the *problem* it addresses, *specific issues*, a couple of *design choices* (determining different pattern variants), a reference to *related patterns*, and finally remarks regarding pattern *implementation*. BPMN 2.0 (OMG, 2011) was used to specify which mashup basic operators (here acting as domain specific building blocks) need to be executed and in what order (*i,e.*, the control flow).

Pattern MP1: ALERTER

Description: Mashups are not restricted to constantly interacting with users to perform some action. An *Alerter* pattern periodically monitors a system of interest on behalf of the user and, based on previously established conditions, sends notifications only when events of interest take place. These alerts can take the form of visual elements on the user's console, e-mail messages, or SMS (text) messages. Another advantage of using an *Alerter* mashup pattern relates to situations where multiple systems must be monitored at the same time, eventually overloading the human operator with too much information. In that case, correlated events from different systems can be summarized to decrease the number of notifications.

Example: In the service dispatching scenario, service tickets are created at no specific time, and a dispatcher responsible for assigning those tickets needs to constantly access the customer's ticketing system to check for new service requests.

Problem: The constant monitoring of management systems to look for new events can become a problem if the dispatcher does not access the system sufficiently often, or if the time spent in unnecessary repeated accesses degrades the dispatcher's productivity. It can be even worse when the amount of monitored information is very large, or when the dispatcher needs to promptly react to time-sensitive events.

Issues:

a. The process does not have to wait for any response of the actors receiving the notification.

b. A response of the external system is mandatory.

c. The notification must be sent electronically to one participating human.

d. The requested information is provided by a software agent.

Design Choices:

Major design choice is whether multiple events should be aggregated into one single notification or not. This results in two pattern variants with the following informal semantics:

1. *Single-Alerter* (Fig. 3.1): All individual events that may generate alerts are notified to the human operator.
2. *Aggregate-Alerter* (Fig. 3.2): The control basic operator may be replaced by a more sophisticated logic aimed to aggregate corresponding events.

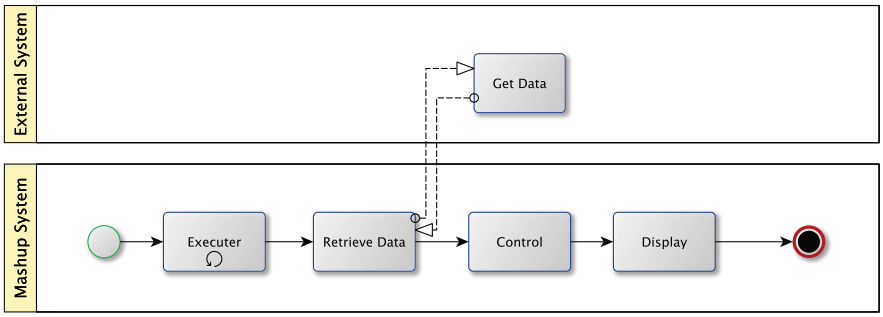

Figure 3.1: Single-Alerter

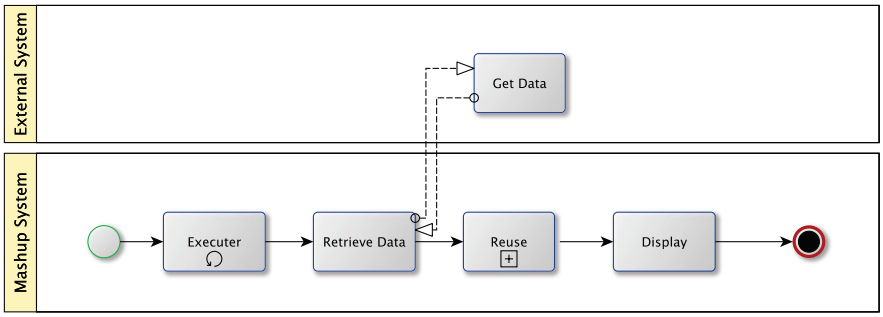

Figure 3.2: Aggregate-Alerter

Related Patterns: *Bi-Directional Performative* (WAP4), *Notification* (WAP 5) (THOM; REICHERT; IOCHPE, 2009), *Information Request* (WAP 6) (THOM; REICHERT; IOCHPE, 2009), *Decision* (WAP 7) (THOM; REICHERT; IOCHPE, 2009), *Send/Receive* (BARROS; DUMAS; HOFSTEDE, 2005), *One-to-Many Send Receive* (BARROS; DUMAS; HOFSTEDE, 2005), *Synchronous Transfer* (MULYAR; AALST, 2008).

Implementation: The *Single-Alerter* pattern can be implemented based on the *Send-Receive* pattern (BARROS; DUMAS; HOFSTEDE, 2005) or the *Single-Notification* (Design-Choice E(1)) (THOM; REICHERT; IOCHPE, 2009). For implementing the *Multi-Alerter* pattern, it is possible to use the *One-to-Many Send/Receive* pattern (BARROS; DUMAS; HOFSTEDE, 2005) or the *Multi-Notification* (Design-Choice (E(2)) (THOM; REICHERT; IOCHPE, 2009).

Pattern MP2: IMPORTER

Description: If external resources natively expose an application programming interface (API), then leveraging their information is just a matter of basic software programming. However, it is often the case that the most valuable content is locked away in closed or proprietary formats. In these cases, an *Importer* pattern abstracts the different methods used to access the external data so that data consistency maintenance becomes transparent to the user.

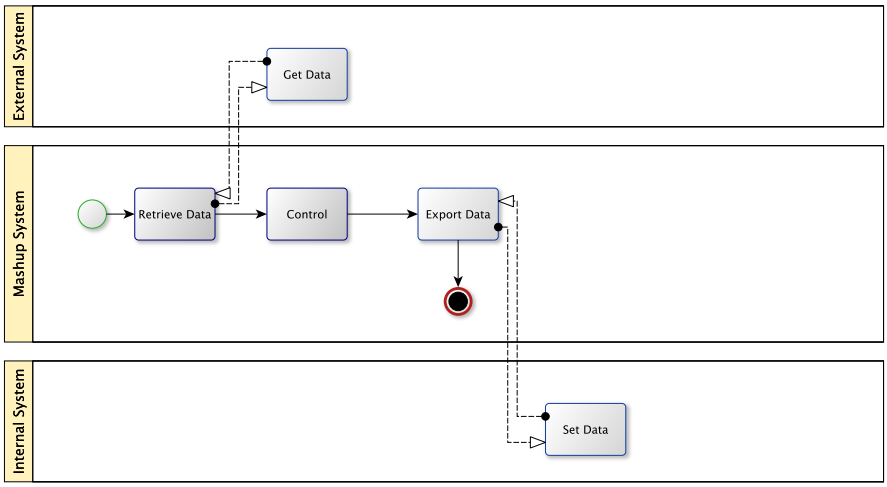

Figure 3.3: Single-Importer Pattern

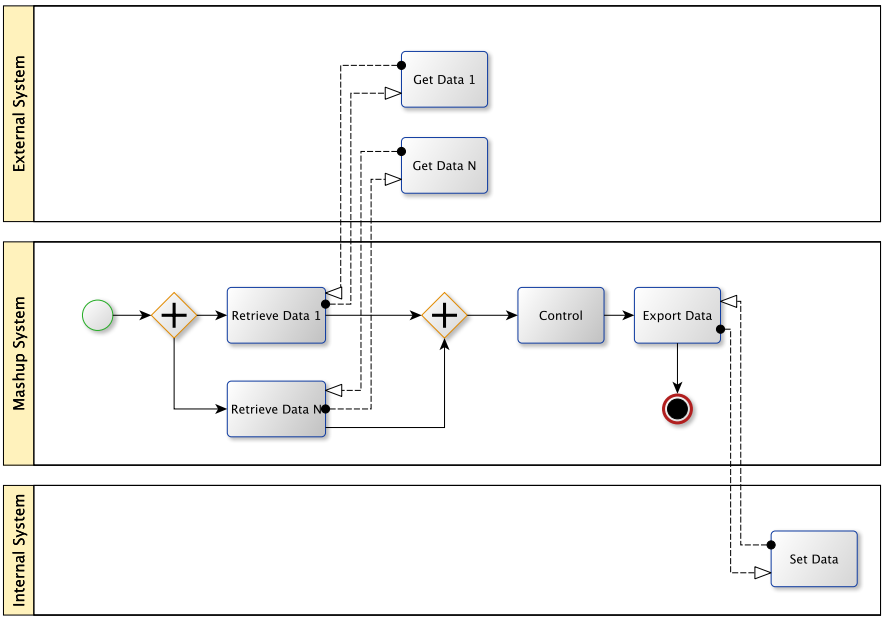

Figure 3.4: Multi-Importer Pattern

Example: Data adapters can grant one party access to the system of another's. When adapters are not available, screen scrapers can be used to access the Web interface of the remote system and retrieve the common data. Finally, users can access one another's system and manually copy and paste the common data into their own system's interface.

Problem: In ITSM, it is not uncommon to find scenarios where customers and service providers require the use of common data, although they use their own, particular database systems. To maintain data consistency across such systems, diverse methods can be used. In all these cases, maintaining data consistency is not transparent for the users because they need to consciously switch the integration method when accessing multiple systems.

Issues:

a. A response of the external data source is mandatory.

b. The sender of the request continues only after having received the requested information.

c. The requested information is provided by a software agent.

Design Choices:

Major design choice is whether multiple multiple systems should be accessed in order to import a complementary data. This results in two pattern variants with the following informal semantics:

1. *Single-Importer* (Fig. 3.3): The information on interest is retrieved from only one external system.
2. *Multi-Importer* (Fig. 3.4): Complementary data is retrieved from multiple external systems.

Related Patterns: *Information Request* (WAP 6) (THOM; REICHERT; IOCHPE, 2009), *Send/Receive* (BARROS; DUMAS; HOFSTEDE, 2005), *One-to-Many Send Receive* (BARROS; DUMAS; HOFSTEDE, 2005), *Synchronous Transfer* (MULYAR; AALST, 2008), *Copy In/Copy Out* (RUSSELL *et al.*, 2006).

Implementation: The pattern can be implemented based on the *Information Request* pattern (WAP 6) (THOM; REICHERT; IOCHPE, 2009).

Pattern MP3: TRANSFORM

Description: During the process of importing data, transform basic operators can be inserted into the mashup logic to enable the processing of certain types of data and thus both materializing the compatibility between systems and satisfying the requirements of the IT process. With the *Transform* pattern it is possible to reduce the number of manual interventions performed by the human through the automation of these adaptations.

Example: While copying a field, a user needs to apply rules to filter out confidential information, or the data needs to be reformatted before it could be used by a different system (*e.g.*, US and UK date formats). These data transformations are usually manually performed because ITSM systems are often created without having integration in mind.

Problem: While interacting with different systems, it is common to find cases where data needs to undergo some simple processing while being transferred from one screen to another. This requires additional overhead for the human operator, thus increasing the probability of human-errors and decreasing the productivity.

Issues:

a. A response of the external data source is mandatory.

b. The sender of the request continues only after having received the requested information.

c. The requested information is provided by a software agent.

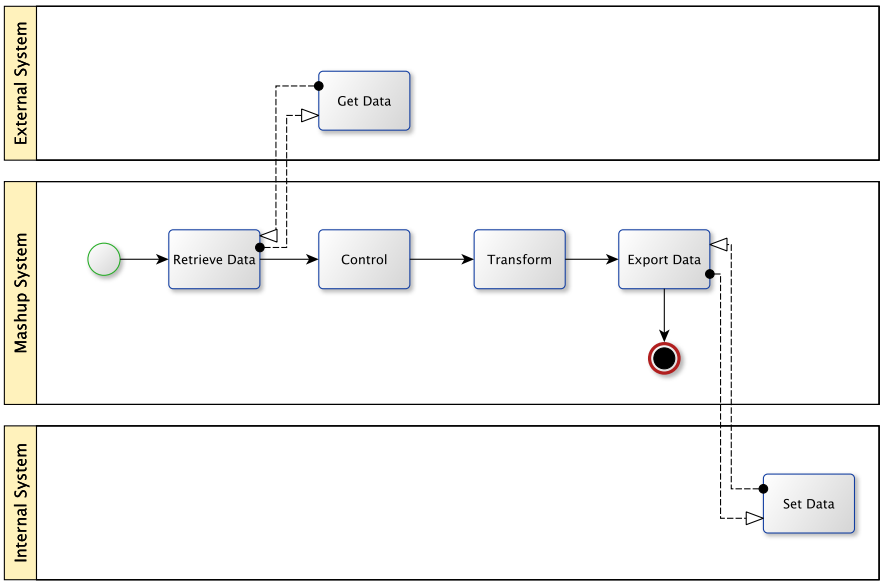

Figure 3.5: Single-Transform Pattern

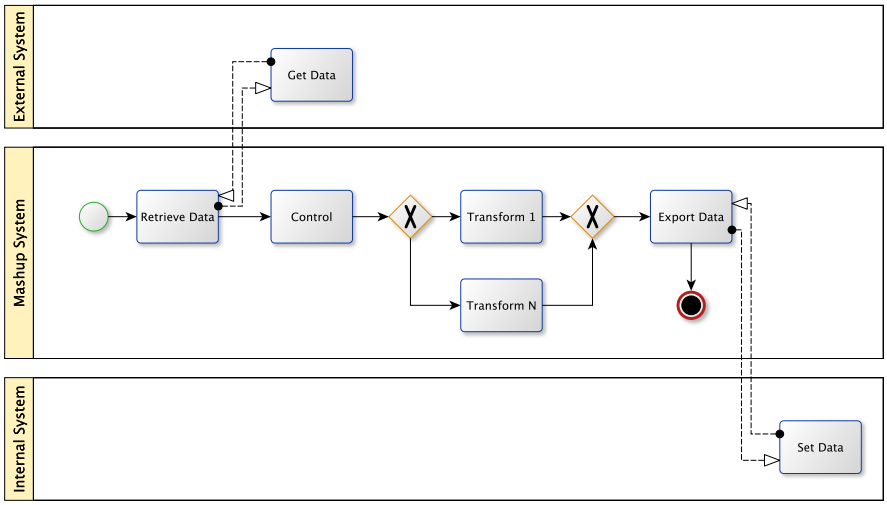

Figure 3.6: Multi-Transform Pattern

Design Choices:

Major design choice is whether data should be transformed in multiple ways (*e.g.*, from US and BR to UK date format). This results in two pattern variants with the following informal semantics:

1. *Single-Transform* (Fig. 3.5): Only one type of transformation is made during the augmentation step.
2. *Multi-Transform* (Fig. 3.6): Data may be transformed in multiple ways based on specific conditions.

Related Patterns: *Information Request* (WAP 6) (THOM; REICHERT; IOCHPE, 2009), *Send/Receive* (BARROS; DUMAS; HOFSTEDE, 2005), *One-to-Many Send Receive* (BARROS; DUMAS; HOFSTEDE, 2005), *Synchronous Transfer* (MULYAR; AALST, 2008), *Copy In/Copy Out* (RUSSELL *et al.*, 2006).

Implementation: The pattern can be implemented based on the *Information Request* pattern (WAP 6) (THOM; REICHERT; IOCHPE, 2009).

Pattern MP4: DISPLAYER

Description: By definition, mashups combine data from multiple sources and present the results of this combination in a Web page. However, this integration tends to occur only at the presentation level; it rarely occurs at the data level. This means that information from multiple systems can be presented alone in the same Web page as independent widgets. The employment of many *Displayer* patterns in one page enforces the concept of a "single pane of glass". This concept reduces the risks of having a poorly executed process, which would generate errors and impose costs to the company.

Example: The configuration database process can automatically generate a port number that needs to be remembered when installing another application.

Problem: In order to make better decisions, humans involved in ITSM activities use information from multiple systems. This information is often memorized or recorded for future use during the decision making process. If some information is forgotten or misremembered, errors may arise during the process execution.

Issues:

a. A response of the external data source is mandatory.
b. The sender of the request continues only after having received the requested information.
c. The requested information is provided by a software agent.

Design Choices:

Major design choice is whether multiple data sources should be accessed in order to compose the data to be displayed. This results in two pattern variants with the following informal semantics:

1. *Single-Displayer* (Fig. 3.7): The information on interest is retrieved from only one external system.
2. *Multi-Displayer* (Fig. 3.8): Complementary data is retrieved from multiple external systems.

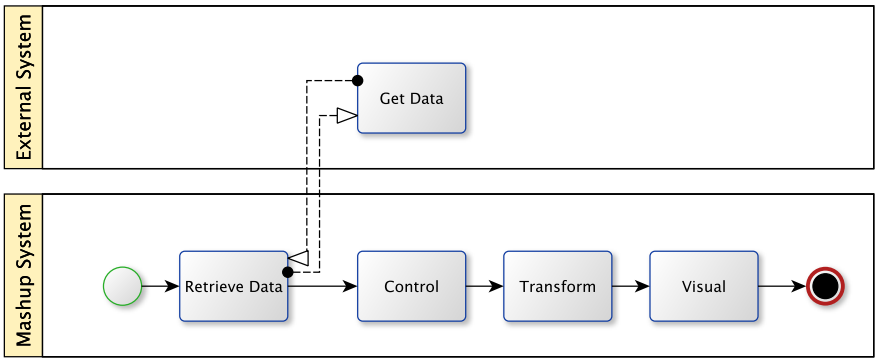

Figure 3.7: Single-Displayer Pattern

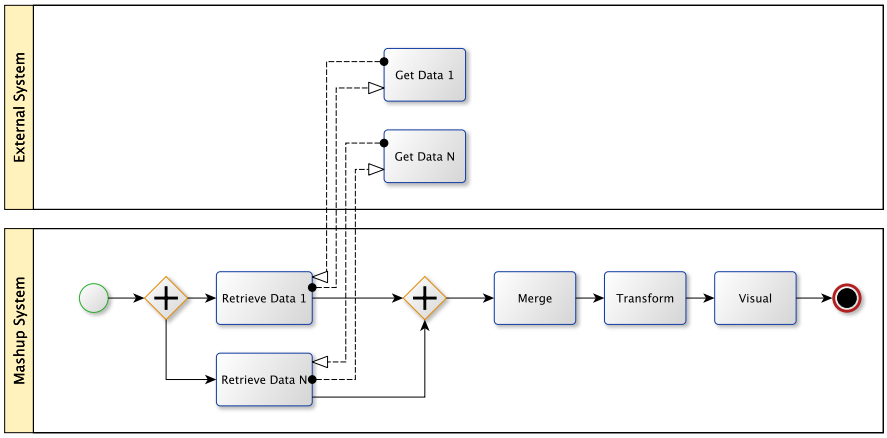

Figure 3.8: Multi-Displayer Pattern

Related Patterns: *Information Request* (WAP 6) (THOM; REICHERT; IOCHPE, 2009), *Send/Receive* (BARROS; DUMAS; HOFSTEDE, 2005), *One-to-Many Send Receive* (BARROS; DUMAS; HOFSTEDE, 2005), *Synchronous Transfer* (MULYAR; AALST, 2008).

Implementation: The pattern can be implemented based on the *Information Request* pattern (WAP 6) (THOM; REICHERT; IOCHPE, 2009).

As will be seem in Chapter 4, these patterns could be applied to the processes executed by a group of 5 human operators at two geographically distributed service delivery units of a global IT organization. The patterns were observed in the context of dispatching process, a Request Fulfillment activity based on human operators with knowledge of standard fulfillment procedures. It is important to remark that, although a single case study may not be sufficient to validate the presented patterns[1], they were defined in a high-level of abstraction, thus being able to be applied for recurring problems in disparate human-centered activities.

3.2.3 Quantitative Modeling for Productivity Assessment

Despite its importance, ITIL only provides high-level generic guidelines to IT organizations, without proposing, for example, concrete models and methods for capturing

[1]The validation of the presented mashup patterns is out of the scope of this thesis.

metrics and evaluating the quality of IT processes. Such evaluation is important for the IT service providers to quantify and most importantly to predict the deployment impact of IT solutions. The scientific community has worked to propose models, methodologies, and metrics to fill this gap. Two of such models are presented bellow, and then a combined model to account for inefficiencies throughout the IT process is proposed. These models represent the most comprehensive models available in the literature and were extensively (KIERAS, 2001) (KOESTER; LEVINE, 1994) (BROWN; HELLERSTEIN, 2004) (BROWN; KELLER; HELLERSTEIN, 2005) investigated by both academia and industry.

3.2.3.1 Keystroke-Level Model (KLM)

KLM was proposed to predict the time an expert user takes to perform a given task on a given computer system (CARD; NEWELL; MORAN, 2000) (KIERAS, 2001). Execution time is essentially estimated by listing the sequence of keystroke-level actions the user must perform to accomplish a task and then summing the times of the individual actions. The sequence of actions, and the respective average execution times, is taken from a set of gestures provided by KLM. The total task execution time is the sum of the time for each of the gestures in the sequence. The following are the standard actions and estimated times for each one.

K - Keystroke (.12 - 1.2 sec; .28 recommended for most users): Represents the pressing of a key or button on the keyboard. Pressing the SHIFT or CONTROL key counts as a separate keystroke. Different experience levels have different times for the K operator:

- Expert typist (90 wpm): .12 sec
- Average skilled typist (55 wpm): .20 sec
- Average nonsecretarial typist (40 wpm): .28 sec
- Worst typist (unfamiliar with keyboard): 1.2 sec

The average non-secretarial typist (.28 sec) is a good design point for characterizing the typical computer user; these are people familiar enough with the keyboard to use it fluently, but are not professional-grade typists.

T(n) - Type a sequence of n characters on a keyboard (n x K sec): This is simply a shorthand for a series of K operators, and would normally be used only when the user is typing a string of characters that is a single "chunk," such as a filename.

P - Point with mouse to a target on the display (1.1 sec): Represents the action of moving the mouse to point the cursor to a desired place on the screen. The actual time required can be determined from Fitt's law (ACCOT; ZHAI, 1997). For typical situations, it ranges from .8 to 1.5 sec, with an average of 1.1 sec. If great accuracy is not required, or the movement distances or target sizes are not unusual, this average can be used instead of more precise times.

B - Press or release mouse button (.1 sec): This is a highly practiced, very rapid movement. Figure .1 sec for pushing the button down or letting it up.

BB - Click mouse button (.2 sec): Pushing and releasing the mouse button rapidly, as in a selection click, counts as two B operators, for a total of .2 sec.

H - Home hands to keyboard or mouse (.4 sec): Since the targets are pretty large, and the movement well practiced, moving the hand between keyboard and mouse, and vice-versa, is relatively fast.

M - Mental act of routine thinking or perception (.6 - 1.35 sec; use 1.2 sec): How long it takes to perform a mental act depends on what cognitive processes are involved, and is highly variable from situation to situation or person to person. This operator is based on the fact that when reasonably experienced users are engaged in routine operation of a computer, there are pauses in the stream of actions that are about a second long and that are associated with routine acts such as remembering a filename or finding something on the screen. The M operator is intended to represent this routine thinking, not complex, lengthy, problem-solving, racking the brain, or creative meditations. In a variety of routine computer usage tasks such as word processing and spreadsheet usage, these routine pauses are fairly uniform in length, justifying the simplifying assumption that all Ms take the same amount of time, around one sec.

Based on the available results (OLSON; OLSON, 1990), a good overall estimate for the duration of an M is 1.2 sec. Choosing how many Ms are involved, and where they appear, is the hardest part of using the KLM.

W(t) - Waiting for the system to respond (time t must be determined): This is the time that the user must wait on the system before he or she can proceed. It is not necessarily the same as the time required by the system, because the user may be able to overlap other activities while the system is working.

One example of the use of KLM to predict interaction time is the case of a file deletion by a human operator. In this simple case, the procedure is to drag the file icon to the trash can icon. For this, the action sequence can be represented as follows:

Table 3.1: Keystroke-Level Model example

Description	**Operation**	**Time (sec)**
Initiate the deletion (decide to do the task)	M	1.2 sec
Point to file icon	P	1.1 sec
Press and hold mouse button	B	0.1 sec
Drag file icon to trash can icon	P	1.1 sec
Release mouse button	B	0.1 sec

$$T_{total} = P + P + B + B + M = 2 * 1.1 + 2 * 0.1 + 1.2 = 3.60 \text{ sec} \qquad (3.1)$$

From this equation, T_{total} represents the total time, estimated by KLM, an expert user takes to accomplish the file deletion task.

3.2.3.2 Complexity Model

Brown and Hellerstein (BROWN; HELLERSTEIN, 2004) introduced a methodology for quantitative benchmarking of configuration complexity of initial system setup. The authors consider this complexity as one of the major obstacles to the adoption of new

solutions by IT organizations and is largely responsible for the increasing costs. Unlikely traditional performance benchmarks (*e.g.*, workload-driven), configuration complexity benchmarks must be process-driven, with metrics based on an analysis of a captured process rather than on a system's response to a workload. The methodology is based on the following 4 steps:

1. **Process capture**: the expert operator performs the configuration process on the system under test while the benchmark infrastructure records his/her interactions in a response file. This step can be repeated as needed to ensure the response file reflects the best-case configuration process;

2. **Process decomposition and analysis**: The benchmark infrastructure maps each recorded interaction in the response file into a parameterized canonical configuration action. Examples of canonical configuration actions include selecting installable features and entering a configuration variable;

3. **Process scoring**: The process is scored as to its complexity by estimating the interaction time and success probability for the aggregate set of canonical actions it encompasses. This scoring will be based on a model of the complexity of the canonical actions and the interrelationships between them, calibrated by studies of how human operators perform. Note that once these calibration studies have been done, they can be applied repeatedly to many benchmarks;

4. **Process validation**: The benchmark verifies that the captured process has achieved its configuration goal subject to lifecycle-based quality constraints.

This work has been extended in subsequent works. Brown *et al.* (BROWN; KELLER; HELLERSTEIN, 2005) proposed a model of configuration activity based on three concepts: configuration goals, procedures and actions. This model, along with the earlier methodology, allows an analyst to assess the complexity of different systems through proposed metrics classified into:

- **Execution complexity**: This group of metrics evaluate the complexity related to the procedure execution;

- **Parameter complexity**: This subset of metrics is used to assess the complexity associated to the task of providing a procedure with parameters (by either the human operator or the system itself);

- **Memory complexity**: This group of metrics is used to evaluate the number of parameters that must be remembered during the execution of a procedure, as well as the interval they must be retained in memory. To compute these metrics, the administrator's memory is modeled as a Last-In-First-Out (LIFO) stack with non-associative lookup.

Looking at the problem from a business perspective, Diao *et al.* (DIAO *et al.*, 2007) extended these techniques to propose new complexity metrics and assess business-level performance indicators (*e.g.*, labor cost, productivity, quality). Finally, Diao and Keller (DIAO; KELLER, 2006) extended the metrics proposed in (BROWN; KELLER; HELLERSTEIN, 2005) to quantify the complexity of overall IT processes. The approach used by

Diao *et al.* can be summarized in three steps: assessing the complexity and timing a baseline scenario, construction of the regression model and evaluation of the model quality, and finally employing the model to predict labor costs, such as time. The relationship between time and complexity metrics are investigated using the multiple linear regression technique, with equation presented below:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \tag{3.2}$$

In the above equation, the x_i represent the IT management complexity metrics, and the least squares approach is employed to discover the β_i value. Further explanation of this methodology is beyond the scope of this thesis and can be found in (DIAO *et al.*, 2007).

A simpler version of this model was presented in (SHWARTZ; DIAO; GRABARNIK, 2009). In this version, the complexity metrics are grouped by subtask, and the coefficients of Equation 3.2 are assumed to be proportional to the time spent on each subtask and equally weighted among metrics associated with the same subtask. This simplifies the model to the point that it can be applied with nearly the same ease and generality as the KLM model of the previous section, but addressing a much larger array of metrics. This work also extended the model to cover more complex processes which include forks, merges, and joins.

3.2.3.3 Combined Model

The previous models (*i.e.*, KLM (KIERAS, 2001) and Complexity Model (DIAO *et al.*, 2007)) are combined to take advantage of their relative merits. The value of the combined model is in allowing an analyst to predict the performance improvements quantitatively before deploying mashups over their current ITSM processes. The KLM model is more widely used, is well corroborated by experiment (see for example (GRAY; JOHN; ATWOOD, 1993)), and provides a wealth of detail at the lower level of human-computer interactions, while the Complexity model addresses both levels of inefficiencies. Thus, the presented models are the current best starting points for creating a new model to better evaluate the performance of IT processes from a time productivity perspective. It is important to notice that to avoid double-counting, all the complexity metrics are discarded, except the memory and decision metrics, which capture higher level potential inefficiencies not addressed by KLM.

An analyst constructs the combined model in the following 6 stages inspired by Six Sigma methodology, discussed in 2.1:

1. The analyst works with a domain expert to determine the tasks and subtasks of the process.

2. The analyst measures the time to perform each subtask.

3. The analyst works with a domain expert to determine the complexity metrics for the Complexity model.

4. The analyst derives the Complexity model coefficients from the time measurements using the method of (DIAO *et al.*, 2007).

5. The analyst determines the KLM model through observation of user interactions.

6. Since β_0 of Equation 3.2 represents the expected time for all factors not explained by the complexity model, the time predicted by the KLM model can be subtracted from it.

The scenario described in the *Alerter* pattern, which is composed of several subtasks, can be used to illustrate the use of the proposed methodology. The first subtask is for the operator to notice that it is time to check for new events. This time spent "becoming aware" of the need to start the task is represented as T_a. Once the operator decides to look for new requests, the next subtask is to interact with tools to examine the new events. The time spent on this subtask is labeled as T_k. This second subtask can be modeled by KLM, while the first subtask demands investigation beyond the scope of this thesis. It is important to observe that this scenario does not include a subtask associated with memory complexity. This is because the human operator does not need to retain any information to look for new requests. In this example, it is considered that all requests are processed independently of the others. The *Alerter* pattern can decrease time spent on the task by reducing the awareness time T_a to zero. Once the requests arise, notifications are sent to the human operator automatically. In addition, a well-designed implementation of the *Alerter* pattern could reduce T_k, for example by allowing the dispatcher to access the ticket associated with an alert with just one mouse click.

The scenarios where the *Importer* and *Transformer* patterns can be applied present both kinds of time inefficiencies found on this thesis: mechanical execution (T_k) and task complexity (T_c). Since both operations can be completely automated by employing mashup patterns, the time reduction in those scenarios is 100%. The same applies to the *Displayer* pattern. Since the necessary information to process a request is provided in one single screen to the human operator, the time spent looking for the information is decreased to zero and the time associated with complexity is significantly decreased due to eliminating the need to remember one specific piece of information. Table 3.2 summarizes the time reduction estimation for each of the presented patterns.

Table 3.2: Time reduction estimation for mashup patterns

Pattern	Current	New
Alerter	$T_a + T_k$	T'_k
Importer	$N_{fields}(T_k + T_c)$	0 sec
Transformer	$T_k + T_c$	0 sec
Displayer	$T_k + T_c$	T'_c

3.3 Reliability

Automation is often used by companies to obtain tight performance bounds (KELLER *et al.*, 2004). However, that may not be feasible in some cases because of additional effort to deploy and maintain the automation infrastructure (BROWN; KELLER; HELLERSTEIN, 2005). This is specially true when processes are constantly changing or are too complex to be automated. In such cases, the existence of human operators, although required, may introduce defects in the process. Previous researches (BENSON *et al.*, 2010) (GRAY, 1986) have showed that human error is the largest contributor to reduced dependability in IT systems, and occur despite experience (OPPENHEIMER; GANAPATHI; PATTERSON, 2003). Even additional training and familiarity with systems cannot

eliminate all the errors (*i.e.*, mistakes, slips, or lapses) made by the human operator. For example, a database administrator can accidentally delete a semicolon when executing consecutive commands, which may lead to a system outage or to a permanently damaged data.

In IT service provide organizations, preventing human errors from affecting the system – by avoidance or interception – is critical because of the strict requirements for quality. Errors in such environment introduce a considerable variance in quality, thus reflecting in severe penalties from Service Level Objective (SLO) violations. Variability is assessed as the number of defective units at the output of the process, and is usually addressed by IT service quality engineers using techniques from the manufacturing domain. For example, SixSigma (PANDE; NEUMAN; CAVANAGH, 2000) and Lean (KRAFCIK, 1988) are two popular methodologies used in conjunction (*i.e.*, Lean Sigma (ALLEN, 2010)) to identify and remove the causes of defects and minimize the variability in both manufacturing and business processes.

This thesis addresses the problem of reducing the occurrence of human errors in the Request Fulfillment process by using mashups technology. A methodology (SANTOS *et al.*, 2013) inspired in the Human Error Assessment and Reduction Technique (HEART) is proposed to improve the performance of ITSM processes by eliminating defects and reducing variability. Again, exogenous elements are not addressed, such as answering telephone calls. After all, it is not possible to change the human condition, but it is possible to change the conditions under which humans work. In particular, the focus is on error-prone activities performed by human operators that can be identified and quantified.

3.3.1 Human Errors

Human beings have the ability to execute multiple and complex tasks (mental and physical) at the same time. However, although skills and expertise level can vary among people, all humans reach their natural limits. When such limits are reached or exceeded, humans become susceptible to making errors.

Several works are found in the literature aiming to define "human error". However, up to now, there is no agreement on a unique definition for the term. In this thesis, it is assumed the definition proposed by Reason (REASON, 1990), who considers that erroneous actions include all situations in which a planned sequence of physical or mental activities failed to obtain a result and whose errors cannot be attributed to interventions of external causes. Reason's work is based on the Skill, Rule, Knowledge (SRK) based approach (RASMUSSEN, 1983), which introduces a classification for errors common to all professions (*e.g.*, aviation, military, nuclear power, health care). These errors are of the following types:

1. **Knowledge-based errors** - Occur because of a deficit of knowledge. For example, a person may want to perform some plan or action, but the plan or action is implemented partially or incorrectly because of a lack of knowledge, thus not achieving the proper result;

2. **Rule-based errors** - Occur when a well-known rule is wrongly used, or a situation is misinterpreted. For example, when a person using a particular operating system is forced to use a different one, and then tries to perform the same actions even if the Graphical User Interface (GUI) is different;

3. **Skill-based errors** - Occur when actions are divergent to the original intentions. For example, when a user performs an automatic action, with little attention, and forgets to execute a specific task;

Reason (REASON, 1990) also introduced the Generic Error Modeling System (GEMS), which is an extension of the SRK approach. In that work, Reason observed that the type of errors can be either involuntary or intentional. Involuntary actions (*i.e.*, slips and lapses) are those that deviate from planned intentions and, thus, do not reach their goals. Intentional actions (*i.e.*, mistakes and violations), on the other hand, are performed consciously but the desired result is not achieved. Figure 3.9 presents GEMS human error taxonomy.

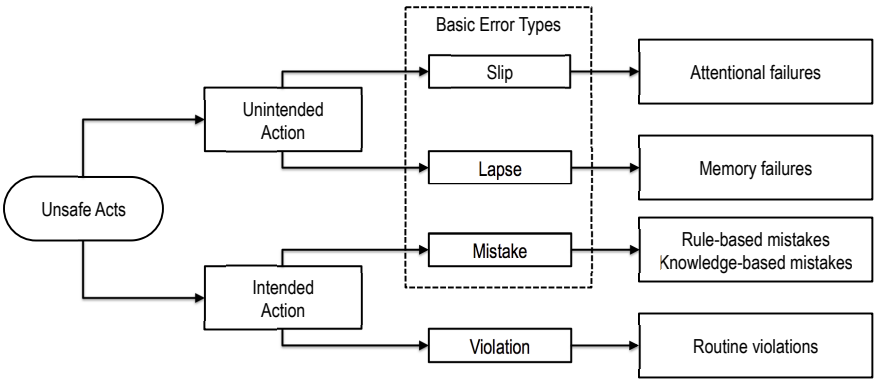

Figure 3.9: Generic Error Modeling System (GEMS)

Previous investigations (MATTHEWS *et al.*, 2000) found that 61% of the human errors are skill-based, which means that humans are prone to slip and lapse with familiar tasks. When such tasks become harder, rule-based (28%) and knowledge-based (11%) become the most common failures made by humans.

3.3.2 Human Errors in Service Management Processes

The continuous pressure on IT companies to increase their competitiveness forces human operators to reach or exceed their natural limits. Such constant effort, poorly designed systems, and lack of experience are examples why the human operators make errors during their daily activities (BROWN; HELLERSTEIN, 2004). In particular, this thesis addresses error-prone activities, *i.e.*, portions of an ITSM process characterized by high risk of human error, which can be assessed through instrumentation or observation, and can be improved through design and automation. It is important to note that, in the context of ITSM, human operators are well trained in specific areas of expertise and mostly of the work include routine tasks (*e.g.*, dispatching service requests, configure database server, restore password). These routine tasks are susceptible to skill-based errors, which can be tackled through proper instrumentation.

Errors in ITSM processes can occur due to a wide variety of causes, for example:

- **Attention Span**: As the number and complexity of tasks increase, the ability to focus and maintain attention decreases;

- **Memory**: The short term (or working) memory is limited to the equivalent of retaining around seven distinct items at a time. This may not be sufficient for active processing of some complex tasks;

- **Situation Awareness**: A person's awareness (or not) and perception of different elements in their environment affect their information processing and actions;

- **Personal Resources**: the ability to process information appropriately diminishes as stress and fatigue increase.

In order to discover common error-prone activities in ITSM, were analyzed the tasks performed by a group of SME in a global IT Service Support and Delivery organization – this group is characterized in details in Chapter 4. Based on documents related to service incidents, it was possible to identify the non-exhaustive list of human errors presented below, presented in rough order from lowest to highest level. These errors are illustrated using two scenarios: a change management scenario in which a dispatcher assigns change requests to the appropriate technician, who is responsible for resolving them; and a continuity management scenario in which an analyst designs a disaster-recovery plan which will then be tested manually.

- **Action Errors**: Are associated with actions performed by an operator on an object (*i.e.*, element of interest) and that change the state of the system. For example: the change request contains correct information, but the technician implements the change on the wrong server; the change request requires two technicians to operate in order (first reinstall the operating system, then install the data base) but the actions are performed out of order; the business recovery plan is complete but the tester fails to implement some part of the plan, causing the test to fail.

- **Retrieval Errors**: Are failures to retrieve correct information, either from memory or from a visual display, to be used in a further step. For example: the technician retrieves and acts upon a change request that had been assigned to a different technician; a business continuity analyst does not notice a list of required static IP addresses in the client configuration and so fails to include them in the business recovery plan.

- **Checking Errors**: Are errors that occur when the operator fails to check some information. It can be the combination of action and retrieval errors. For example: the technician marks the wrong change request as having been resolved.

- **Decision Errors**: Relate to the reasoning capabilities or training of the human operator and occur when the operator has to make an explicit choice between multiple alternatives. For example, a dispatcher assigns a change request to a technician who does not have the appropriate skill level for the task; an analyst suggests an unsuitable hardware substitution in a recovery plan.

- **Communication Errors**: Occur when the operator fails to pass information to another person, either in person or through a system (*e.g.*, instant messenger, e-mail). For example: a dispatcher fails to inform a technician of a change request that had been assigned to him or her;

3.3.3 Mashup Error Prevention Modules

In order to prevent human errors from occurring, or at least to reduce their frequency, a new type of interaction elements, called **Error Prevention** modules (SANTOS *et al.*, 2013), was created to be used by mashup developers. These interaction elements are

used to hide technical details from the mashup developer (DOS SANTOS *et al.*, 2010). Two types of this **Error Prevention** module were defined: *Buffer* module and *Forcing* module. The *Buffer* module allows developers to specify for how long time a specific action should be delayed before being executed, thus providing a recovery window during which an erroneous action can be canceled. According to Reason (REASON, 1990), roughly 78% of human errors can be detected immediately after they are executed. The *Forcing* module, on the other hand, allows developers to introduce confirmation points in the process workflow in order to force the human operator to consciously accept them before proceeding with the mashup execution.

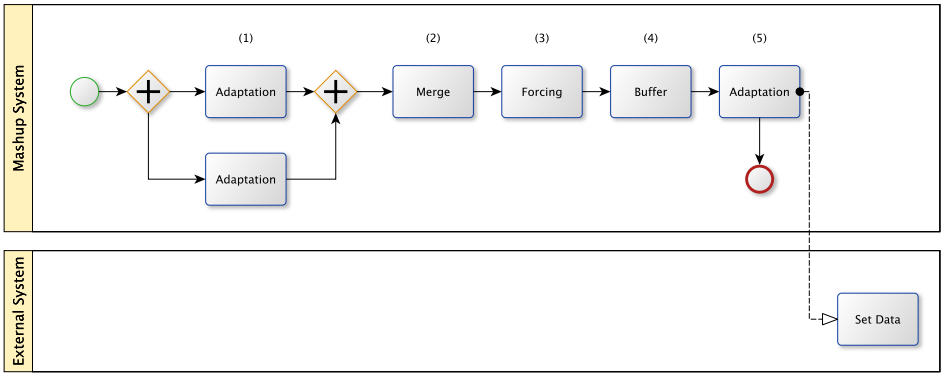

Figure 3.10: Usage example of error prevention modules

Figure 3.10 provides an example illustrating how the **Error Prevention** modules are used in the creation of a mashup application. In step 1, two adaptation modules are used to retrieve information from external data sources, which is merged in step 2. The result of this composition is sent to a third system using another adaptation module in step 5. *Buffer* and *Forcing* modules are used (steps 3 and 4) to first, force the human operator to confirm if the mashup should continue its execution, and second to wait for an amount of time before sending the merged data to the external system.

3.3.4 Quantitative Modeling for Human-Error Assessment

This subsection introduces a methodology inspired in the Human Error Assessment and Reduction Technique (HEART), which is widely used in the field of Human Reliability Assessment (HRA) for the purposes of evaluating the probability of a human error occurring throughout the completion of a specific task. Event Tree Analysis (ETA) is also used for identifying and evaluating the overall probability of human error in the sequence of events performed during the execution of ITSM processes.

3.3.4.1 HEART

Diverse techniques to account for human errors have been developed in the last few decades, such as Technique for Human Error Rate Prediction (THERP) (SWAIN; GUTTERMAN, 1980), Top-down Human Error and Task Analysis (THETA) (MAGUIRE, 2006), Success Likelihood Index Methodology (SLIM) (KHAN; AMYOTTE; MATTIA, 2006), and Human Error Assessment and Reduction Technique (HEART) (WILLIAMS, 1985a). Although such techniques might appear to be outdated, enhanced methods were not developed in the field of Human Reliability Assessment (HRA) (WHITTINGHAM, 2012). These techniques are still used in diverse high risk professions such as nuclear

power, health-care, and aviation. Previous investigations (KIRWAN, 1996) (KIRWAN *et al.*, 1997) (KIRWAN, 1997) have shown that all these techniques present a very significant correlation between their predictions and the corresponding real measurements of human errors.

In this thesis, the human reliability is quantified using HEART, which is considered the most comprehensive method in the field of HRA. HEART method does not require the use of real human error measurements to make the predictions; instead, it specifies 9 Generic Task Types (GTTs) and provides the nominal Human Error Probability (HEP) for each one. HEART also specifies 38 Error Producing Conditions (EPCs), which are factors that can affect human performance making it less reliable (*e.g.*, distractions, overload). HEART methodology is based on the following four steps:

1. Identify all sub-tasks a system operator would be required to execute in order to complete an specific activity;
2. Identify the generic task type corresponding to the activity and its HEP;
3. Select the perceptible EPCs in the activity and that have a negative effect on the outcome;
4. Calculate the final HEP using the equation 3.3.

$$HEP[final] = HEP * \left\{ \prod_{i=1}^{n} \left[(EPC_i - 1)\, Ap_i + 1 \right] \right\} \tag{3.3}$$

In Equation 3.3, $HEP[final]$ is the final probability of human error for an individual activity, HEP is the nominal human unreliability accordingly to the task type, EPC_i is the ith error-promoting condition, and Ap_i is the proportion assessment factor (from 0 to 1) defined by the analyst. The complete set of HEP and EPC values can be found in (WILLIAMS, 1985b).

3.3.4.2 Event-Tree Analysis

Since HEART method aims to quantify the reliability of individual tasks, it needs to be enhanced with different techniques in order to evaluate the overall human error probability in ITSM. In this study, Event-Tree Analysis (ETA) is used with this purpose. In essence, ETA is a model that represents the system dependability based on its subevents safeties. It is called an event tree because the graphical presentation of sequenced events grows like a tree as the number of events increase. As it is shown in Figure 3.11, an event tree consists of an initiating event, its subsequent events and the final results caused by the sequence of events.

The event tree is designed to allow an analyst to identify all of the actions that are required to maintain a safe process, and errors that may occur if certain actions do not occur, or the wrong action is substituted. This approach identifies the checks and measures available so that they may be included in quantitative calculations. The conceptual event tree does not, however, take into account the EPCs that may affect the actions identified in the event tree. Such influences must be accounted for during HEART analysis, subsequently applying those error frequencies to the quantitative event trees. In ETA subsequent events are independent to each other and the specific final result depends only on the initiating event and the subsequent events following. Therefore, the probability of a specific path of

events to occur can be obtained by multiplying the probabilities of all subsequent events existing in a path. This total probability can be obtained using the equation 3.4.

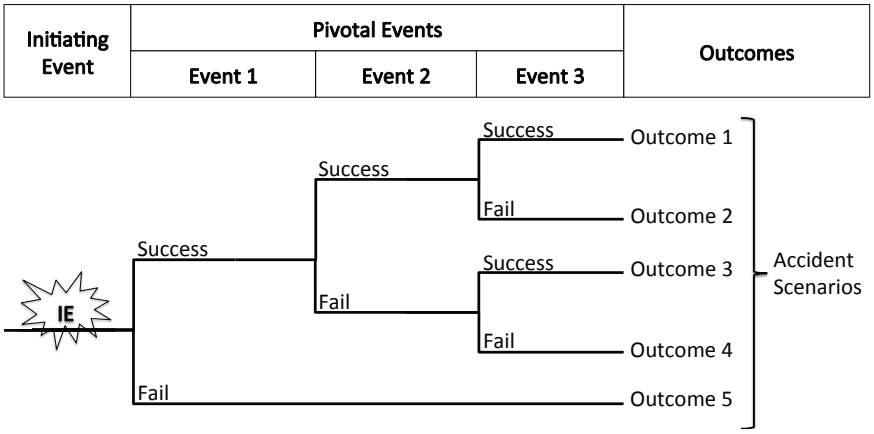

Figure 3.11: Event tree concept

In Equation 3.4, $P[failure]$ represents the probability of failure in a sequence of events and $HEP[final_i]$ is the final probability of human error for a specific task, which is obtained from Equation 3.3.

$$P[failure] = 1 - \prod_{i=1}^{n}(1 - HEP[final_i]) \tag{3.4}$$

3.3.4.3 Methodology for Quality Assessment

In this thesis, the Six Sigma methodology discussed in section 3.2.3.3 is combined with various techniques to improve the performance of ITSM processes by eliminating defects and reducing variation. The HEART technique, for example, was extensively validated by experiment in other fields and represent one of the most widely used technique for Human Reliability Assessment (HRA) (KIRWAN, 1996) (KIRWAN *et al.*, 1997) (KIRWAN, 1997). Event Tree analysis, in turn, is mainly used in the field of safety engineering and reliability engineering and can be used combined HEART to determine the probability of failure in a sequence of events. Finally, mashups represent an effective approach for eliminating root cause problems that degrade the quality in ITSM processes. The methodology introduced in this thesis is based in the following stages:

1. The analyst works with a Subject Matter Expert (SME) to determine the process workflow and root causes of problems;

2. Using HEART technique the analyst determines the Human Error Probability (HEP) of each activity in the workflow;

3. Event Tree Analysis is used to evaluate the performance of overall workflow;

4. The analyst identifies the most appropriate Mashup Patterns and interaction elements to solve problems;

5. The analyst evaluates the process once more to ensure improvements become embedded to the process.

Based on the obtained process workflow (stage 1) and on the individual HEP values for each activity (stage 2), the analyst builds an Event Tree (stage 3) to identify and evaluate the sequence of events in a failure scenario. Once the most error-prone activities are identified, the analyst selects the most appropriate mashup patterns to improve the process (stage 4). Finally, the analyst run through the methodology again (stage 5) – from stage 2 to stage 4 – in order to ensure the improvements become embedded, and to detect new improvement areas in the process.

Besides its simplicity and easy of application, the HEART technique presents several problems. For example, the Error Producing Conditions (EPCs) are not independent of each other; moreover, the use of the method is extremely subjective and relies heavily on the experience of the analyst; and finally, from a qualitative point of view, the EPCs are a useful list of factors to guide quality analysts, but the numerical values (*i.e.*, Assessed Proportion of Affect - APOA) are subjective and imprecise. In order to take into consideration such weaknesses, were analyzed values of 2024 EPCs chosen by Subject Matter Experts (SMEs) in 8 common ITSM activities.

Table 3.3: Assessed Proportion of Affect (APOA)

Class	Average	σ
Low	0.1833	0.086157
Medium	0.5032	0.106578
High	0.8316	0.077288

A clustering algorithm, called k-means, was employed to group these EPCs' values in clusters. k-means is a partitioning technique that groups a dataset into k clusters. The criterion used to assess the number of clusters is the average silhouette of the data. Three significant clusters – *i.e.*, low (L), medium (M), and high (H) – have been identified. For each cluster, was measured the EPC value average, which will be used as input to expression 3.3. In Table 3.3, these averaged results and the respective standard deviation σ is presented. "Linguistic variable" was employed to represent each cluster because it is very useful when situations are too complex of ill-defined to reasonably described in conventional quantitative expressions.

3.4 Different Scenarios of Mashups Application

Mashups are not restricted to the ITSM area. Other fields of study may benefit from the improvements obtained with the usage of the mashups technology. Basically, any scenario where a number of important tools already expose data through Web interfaces can take advantage of mashups. In the network management field, for example, it is common to find experienced network administrators using a variety of different tools (*e.g.*, MRTG, NTop, FlowScan) when managing their computer networks. These tools are usually built without considering the administrators' need to use them in an integrated fashion; integration among network management tools is rarely a design requirement. This situation forces network administrators to find their own ways of integrating such tools, usually through scripts written in a 'domestic' ad hoc way.

Although the issue of retrieving management information from network devices can be solved, for example, using the Simple Network Management Protocol (SNMP), there is no widely accepted or employed solution for combining and visualizing data from multiple network monitoring and performance tools. In this context, mashups are a feasible

solution for network administrators to create their own specialized management tools.

Detecting Border Gateway Protocol (BGP) peering problems:

The internet is composed by independently managed sub-networks called Autonomous Systems (ASes). These ASes use the BGP (Border Gateway Protocol) to define and manage how they connect to each other. This scenario can be referred to as BGP Peering, since two ASes connected through BGP are called BGP peers. In BGP peering the main issues that may arise are related to the unexpected behavior of the inter-AS relationship for example, when the amount of traffic routed from/to a peer suddenly and dramatically exceeds or falls below an expected, usually agreed upon, traffic amount. The cause is an inappropriate peer operation, for example, when a peer is not announcing its prefixes to its neighbor, or when a peer is announcing an unexpected number of routes, possibly violating an SLA. This can happen because of misconfigurations or malfunctioning of network devices. Whichever the reason, the issue should be solved as soon as possible, given the large volume of traffic usually exchanged between ASes and the financial costs incurred by these exchanges.

In this scenario, traditionally, gathering BGP peering information requires the use of several tools with distinct user interfaces. Integration of the data collected is a complex process requiring sophisticated scripts, handling of large amounts of data, and much effort to develop a suitable presentation of the results obtained. A mashup-based solution was proposed by us (DOS SANTOS *et al.*, 2010) (BEZERRA *et al.*, 2010) to reduce the number of interaction levels for the user down to a single user interface that do not require technical skills to be manipulated, while also keeping the complex technical details of the integration hidden from the end-user. Furthermore, the modular nature of mashups enables re-use and sharing of modules between different service providers and network administration staff, which in turn, allows composing even more sophisticated applications.

Identifying botnet communication channels: Botnets are one of the most serious security threats on the Internet (GRIZZARD *et al.*, 2007). These networks are composed by a large number of malware-infected hosts acting under a central command. They are usually employed to launch attacks such as Distributed-Denial-of-Service (DDoS), phishing scams, spamming, and malwares distribution. Mitigation of these threats is not trivial due the dynamic botnet nature. Frequently new and more sophisticated features are adopted to increase botnets' resilience.

With more sophisticated botnets, botmaster identification becomes a harder task. As a consequence, few efforts in the literature are able to handle the most sophisticated botnets. In this context, mashups are capable to integrate disparate botnet information and mitigation tools (*e.g.*, VirusTotal[2], Anubis[3], CwSandbox[4]). This is especially interesting for botnet mitigation since such networks are always evolving. New protocols for Command and Control Channels (C&C Channels), infection methods, and obfuscation techniques are always being used and obligating the employment of new techniques and tools to deal with them. Wrappers can be created to these new tools (SANTOS *et al.*, 2011), enabling them be integrated in a botnet

[2]https://www.virustotal.com/en/

[3]anubis.iseclab.org

[4]www.threattrack.com

mitigation mashup. Additionally, non-exclusive security tools can be employed to increase the effectiveness of the mashup approach. For instance, a module to analyze Netflow flows and find C&C channels can be developed, and a firewall module can dynamically create firewall rules to stop such communication

Management of Network Virtualization Environments (NVEs): Although the research on network virtualization is quite active today (PAN; PAUL; JAIN, 2011), few investigations are found concerning the management of both physical and virtual resources in an integrated manner. In fact, different virtualization vendors are primarily providing proprietary and incompatible Command Line Interfaces (CLIs) and Graphical User Interfaces (GUIs). This lack of standardization for managing both virtual and physical resources inevitably overloads the network administrator, who is forced to employ multiple tools, which may lead to serious consequences (*e.g.*, erroneous actions, increasing of operational costs) for the organizations. Even through management based on proprietary CLIs and GUIs may be enough for small-scale virtualized networks, it is certainly not suitable for the Future Internet environment.

The management of large Network Virtualization Environments (NVEs) (CHOWDHURY; BOUTABA, 2009), created using different virtualization technologies, may quickly evolve into a critical level, where it is impossible to have an integrated vision of the network. In this context, the mashups' composition model is an interesting approach to tackle the heterogeneity, complexity, and stiffness of NVEs. Mashups' technology can be employed by administrators to adapt, customize, and combine existing monitoring tools in order to improve management tasks. In addition, the employment of mashups also supports the monitoring of both physical and virtual network elements in an integrated manner, abstracting all the technical details associated with the access to these elements.

Through the development of mashups for these scenarios, it was possible to observe the occurrence of some of the previously introduced mashup patterns. For example, the *Displayer* pattern (MP4) was broadly employed in (BEZERRA *et al.*, 2010) to retrieve and display the following information to network administrators: MRTG graph images, number of announced routes, local and remote BGP peers, and traffic exchanged. In (SANTOS *et al.*, 2011) the *Transform* pattern (MP3) was used to obtain and process malware reports from online analysis tools (*i.e.*, CwSansbox and Anubis) while the *Importer* pattern (MP2) was employed to retrieve the ISP information where the C&C Server is based. It is important to remark that the presented models (*i.e.*, HEART, KLM and Complexity Model) were extensively validated in other domains, thus they represent a comprehensive foundation for evaluating different aspects of performance.

3.5 Summary

This chapter introduced the concept of mashup patterns and a new category of mashup basic operators (*i.e.*, the error prevention modules), both representing effective approaches to improve performance (*i.e.*, increasing reliability and productivity) of activities when human operators need to handle time-sensitive or error-prone activities. Through the analysis of tasks performed by a group of human operators it was possible to identify a set of common inefficiencies in their implementation of the ITIL Request Fulfillment process. Since these inefficiencies can be found in disparate scenarios, mashup patterns

can be applied as proven and reusable solutions. Productivity gains can observed through a combined model based on the Keystroke-Level and Complexity models. This combined model allows an analyst to assess lower level inefficiencies of human-computer interactions, and higher level inefficiencies of performing IT tasks and subtasks. Reliability improvements, on the other side, can be assessed using both HEART and Event Tree Analysis. These techniques allows an analyst to identify error-prone activities that can be tackled through the new type of mashup basic operators and mashup patterns.

4 CASE STUDY: DISPATCH PROCESS FOR IT SERVICE MANAGEMENT

Request Fulfillment is the Service Operation process – defined by ITIL and discussed in Chapter 2 – that aims to fulfill service requests. It is defined as "management of customer or user requests that are not generated as an incident from an unexpected service delay or disruption" (CANNON; WHEELDON, 2007). Request Fulfillment interfaces primarily with the Service Desk function and the Incident Management process – both defined by ITIL – and can be broken into two main activities: addressing those requests and management of requests made by the customers. Requests are tackled by system administrators (SAs) with technical knowledge to resolve specific requests. Requests are managed by human operators, called dispatchers, with responsibilities that include: monitoring of new requests, dispatching requests to appropriate SAs, and monitoring Service Level Agreements (SLAs).

To investigate the hypothesis presented in this thesis, a case study based on the Request Fulfillment process was conducted in a real global IT service delivery organization, which is structured according to the ITIL standards. The case study focus on the latter activity of Request Fulfillment, an activity centered on human operators called dispatchers, with knowledge of standard fulfillment procedures. In dispatch, once the customer creates a new request (*i.e.*, incident, problem or change request) in Service Desk, a ticket is sent to a dispatcher, who is responsible for analyzing the ticket and selecting the appropriate SA to handle it. The selected SA has the skills and expertise to solve specific requests, is responsible for taking the appropriate actions, and closing the ticket. Dispatching tasks to technicians or work teams takes a significant fraction of the time required to process a service request (VERMA *et al.*, 2011). Because of that, dispatching within service delivery is a good candidate for software to assist the manual process or automate it partially.

4.1 Environment Structure

The constant growth of IT infrastructures and the advent of new technologies impelled IT service organizations to adopt a better governance model in order to guarantee that services are provided consistently (*i.e.*, with low variability). Specifically, the rising complexity of IT environments demands human operators able to support remote infrastructures requiring diverse skills (*e.g.*, UNIX, AIX, Solaris) to handle an ever increasing variety of problems. With highly specialized staff, service providers began to consider grouping system administrators in teams with specialized technical background. Each team is under responsibility of a specific dispatcher, who receives the incoming service requests, determines its nature, and forwards the request to the most appropriate system

administrator according to SA's skills and expertise. This setting has some advantages, such as providing high-quality and agile support, and allowing system administrators to work more efficiently.

Specifically, a service delivery factory can be represented as a network of resources (*i.e.*, system administrators) under responsibility of a coordinator (*i.e.*, dispatcher). Service requests (*i.e.*, tickets) arrive at the entry point according to normal distribution (ANEROUSIS; DIAO; HECHING, 2010). To characterize each ticket, dispatchers use the following attributes: originating domain (or account, in IT terminology), type of skill required, severity (target completion time), and complexity (level of difficulty). Then, dispatchers use their knowledge of team's schedules, workloads and expertise of each system administrator to finally make the assignment. If the team is not able to solve the ticket, the dispatcher forwards the ticket to another dispatcher who is responsible for a different team of SAs. Figure 4.1 presents the elements of this scenario.

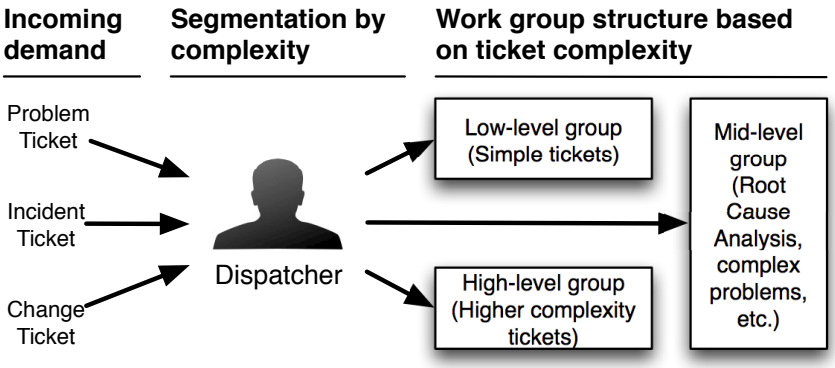

Figure 4.1: E-Ticketing system to Service Desk

System administrators can be characterized by having one or more skills and a level (*i.e.*, low, medium, and high) for each skill. Optionally an affinity attribute indicates familiarity with a particular account. The affinity attribute represents an agent's experience in dealing with issues from a specific domain. The motivation behind the affinity attribute is that the SA is better prepared to solve incoming work once he has performed work for a customer account before. However, in a dynamic environment, it may not be always possible to find an available SA with the required affinity. To prevent unnecessary delay, the service request can be assigned to another available system administrator.

Although fully automated solutions offer many advantages, they may not be feasible for the dispatch activity. Dispatch is constantly changing and is too complex to be automated. For example, in some cases a dispatcher may want to train a new administrator, and so may intentionally assign a request to a less skilled SA. Although required, the use of human dispatchers in service delivery centers may introduce defects in the process, thus reflecting in severe penalties from Service Level Objective (SLO) violations. In this context, mashups are a new technology that simplifies and makes more predictable the creation and execution of dispatching systems, by focusing on the activities performed by each dispatcher, thus decreasing the possibility of errors.

4.2 Data Collection

The investigations carried out during this thesis were based on observations of a global IT service delivery organization. IT service requests arrive from global customers and

may be routed to multiple service delivery units that are globally located. Two different types of data were collected during the observations: the effort data (*e.g.*, assignment handling time), and the demographic data (*e.g.*, team organization and dispatchers actions).

A two-week study was conducted at two geographically distributed service delivery units to gather both kinds of data. During this period, five dispatchers were required to describe the individual actions related to the assignment task. These dispatchers were also shadowed while performing their daily activities. By doing so, it was possible to discover bottlenecks and detect problems in the dispatch process. Each dispatcher is responsible for teams up to 15 system administrators with different skills and expertise, and supports up to 8 different customers (*i.e.*, accounts). A total of 10 assignment executions were observed and time-measured for each dispatcher. At the end of the timing study, a data cleaning was conducted to remove the outliers with unusually long duration (*e.g.*, the dispatcher paused his activities to answer a telephone call). The primary use of the effort timing data is to derive the performance baseline, which corresponds to the second step of the combined model introduced in Section 3.2.3.3.

Tickets associated with two levels of complexity were observed: Simple and Complex. Simple tickets require less time to complete (depending on the skill of the agent). Further, each ticket can also be classified according to one of three severity classes: high, medium, or low. The severity class is used to prioritize work in the queue. A target time is associated with each arriving ticket according to the severity class the ticket belongs to. The target time specifies the maximum total time permitted for the ticket processing in the system. If on a monthly basis the ratio of tickets that exceed the target time limit surpasses the threshold specified by the severity class, the service provider is required refund the customer for each new violation. The penalty varies according to the severity class. Even when the ticket is misrouted, the time countdown does not stops, thus it is important to improve productivity while reducing errors to avoid recurring requests.

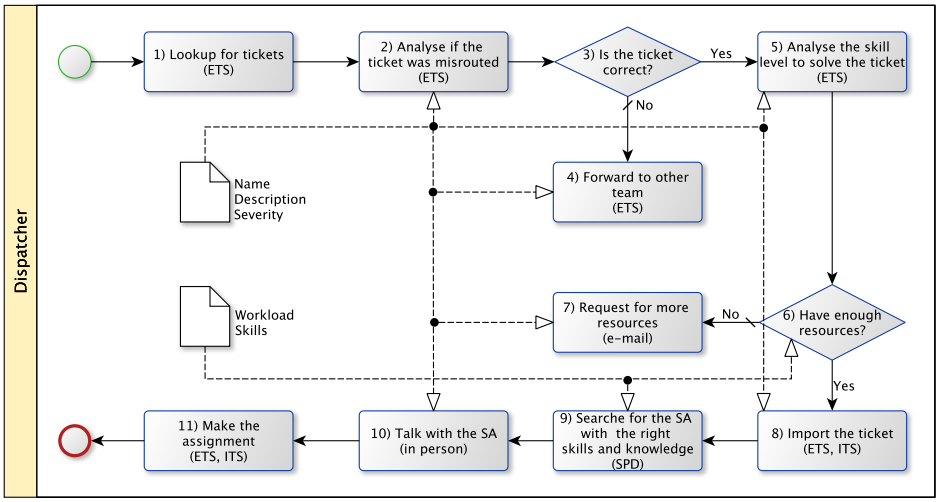

Figure 4.2: Workflow of activities performed by dispatchers in Service Desk

Through the interviews, shadowing daily activities, and analyzing documents related to service delivery, it was possible to determine the common workflow of activities performed by dispatchers in Service Desk, which is presented in Figure 4.2. Once a new request is received from a customer, the dispatcher determines if his team has the proper expertise to solve it, *i.e.*, he analyses if the ticket was misrouted. If the ticket was not

misrouted, the dispatcher analyses the required skill level to solve it. Next, the dispatcher imports data from the customer's ticketing system (*i.e.*, External Ticketing System (ETS)). This activity is usually performed manually because data usually needs to undergo some processing (*e.g.*, anonymising confidential information). Once the ticket is imported into the Internal Ticketing System (ITS), the dispatcher uses information about the system administrator (*i.e.*, availability, workload, skills, and expertise) to select the proper SA to solve it.

4.2.1 Productivity

In order to accomplish his/her tasks, dispatchers may interact with multiple tools (*e.g.*, multiple ticketing systems, calendar-based systems, spreadsheets). Furthermore, dispatchers may also be overloaded with customer requests, which can vary from just a few dozen to hundreds in a busy day. Several time-consuming issues can arise in this scenario, making it infeasible to resolve tickets within the times established in Service Level Agreements (SLAs), and, therefore, resulting in financial loss to service providers. For example, it is common for customers and service providers to use their own ticketing systems, making it necessary to import data and maintain consistency between the systems.

Dispatchers need to deal with data consistency and redundancy without violating any customer policies, such as data compliance for dealing with confidential information. In addition, the dispatcher and his/her team of system administrators may be responsible for multiple customers, where each customer has a different ticketing system. This increases assignment time due to switching between multiple systems.

Finally, information required to find the most appropriate SA for one specific ticket can reside in various systems, requiring different tools to access it. For example, schedules tend to be managed by calendar-based systems, while the SAs actual workloads would be most accurately represented in Request Fulfillment systems and finally it is common to have SAs skills and expertise associated with their user profile in the Service Provider's Directory (SPD).

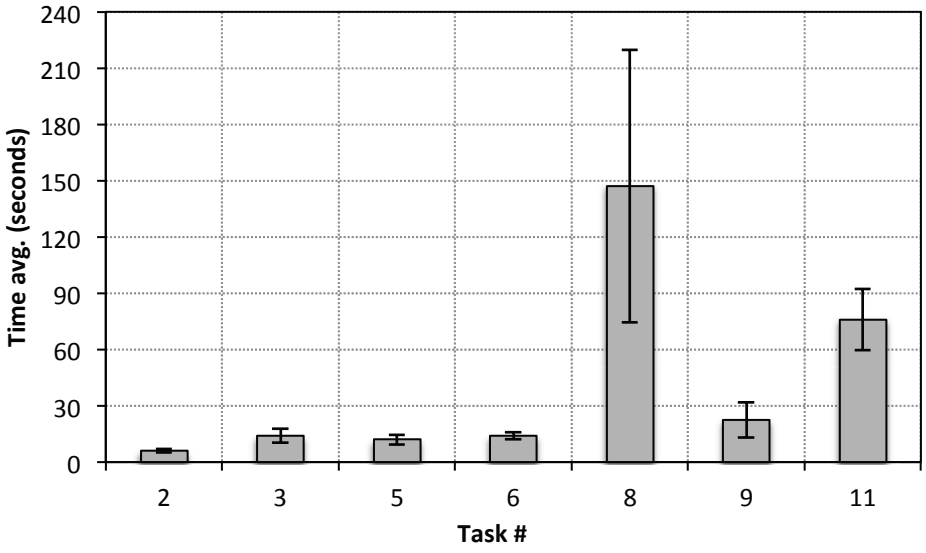

Figure 4.3: Real measurements

In order to discover bottlenecks in the dispatch process, it was performed a series of time measurements among different dispatchers in a service delivery organization. The obtained information also allowed the prediction of improvements that would be obtained by the usage of mashups in the existing process. Using a stopwatch, 10 assignment executions were observed and time-measured for each dispatcher. The dispatch process, represented in Figure 4.2 as a workflow, was obtained following the combined model proposed in Section 3.2.3.3.

The results of the measurements, summarized in Figure 4.3, showed a significant time variation according to the ticket's complexity and the dispatcher's familiarity with the reported issue. For simple tickets (ST), usually repetitive tasks that the dispatcher is habituated to assigning, the time average was 159 seconds (90% confidence level, 23.65 standard deviation), while for high complexity tickets (CT) this time was 357 seconds (90% confidence interval, 41.58 standard deviation). This difference can be justified by the need to spend more time reasoning about all the information related to the ticket, and also by the need to gather and provide detailed information to the system administrators.

The initial tasks (from 2 to 6) represent cognitive actions related to decision making and, as can be observed, they does not represent a significant time to be executed. This is due the fact that the interviewed dispatchers are habituated to his/her teams of system administrators (showing a low staff turnover) and to the same set of customers (*i.e.*, accounts). It was also observed that most of the time was spent importing manually the tickets (task 8), which consumed an average of 50% of the assignment time. Finally, a notable portion of time (*i.e.*, 76 seconds) was spent making the assignment (task 11), activity which involves updating the SA assignment information in both internal (ITS) and external ticketing systems (ETS).

4.2.2 Reliability

Further the time-consuming issues, dispatchers are also vulnerable to making errors. Based on interviews with dispatchers and on the analysis of documents related to service incidents, it was possible to identify the most common failures in dispatch, presented in Figure 4.4. These failures, may occur uniquely or by the combination of the different error types discussed in Section 3.3.

The dispatcher may, for example, accepts a misrouted ticket. This can happen due to a decision error, when he/she fails to decide if the ticket was misrouted; or due an action error, when he/she mistakenly clicks on the accept button instead of rejecting it. Another example of a decision error occurs when the dispatcher fails to detect the right SA to solve the ticket. This may happen when he/she underestimates the skill level to solve the ticket or when he/she fails to identify the proper SA's skills and availability. Finally, even if the dispatcher knows the right SA, he/she may still fail to make the assignment because selects the wrong SA in the ticketing system (*i.e.*, an action error). During the importing step, retrieval errors may also be introduced by a distracted dispatcher. This implies that SAs will base their actions in erroneous information. The consequences of any of such errors can vary largely, from the increasing time to process a customers' request, to a severe interruption in the services due to a wrong server configuration.

In order to discover the Human Error Probability (HEP) of each task performed by dispatchers, they have been classified as type G. According to HEART, the generic task type G involves:

> Completely familiar, well designed, highly practiced, routine task occurring several times per hour, performed at the highest possible standards by highly motivated, highly trained and experienced person, totally aware of the implications of failure, with time to correct potential error, but without the benefit of significant job aids. (FERNÁNDEZ; MÁRQUEZ, 2012)

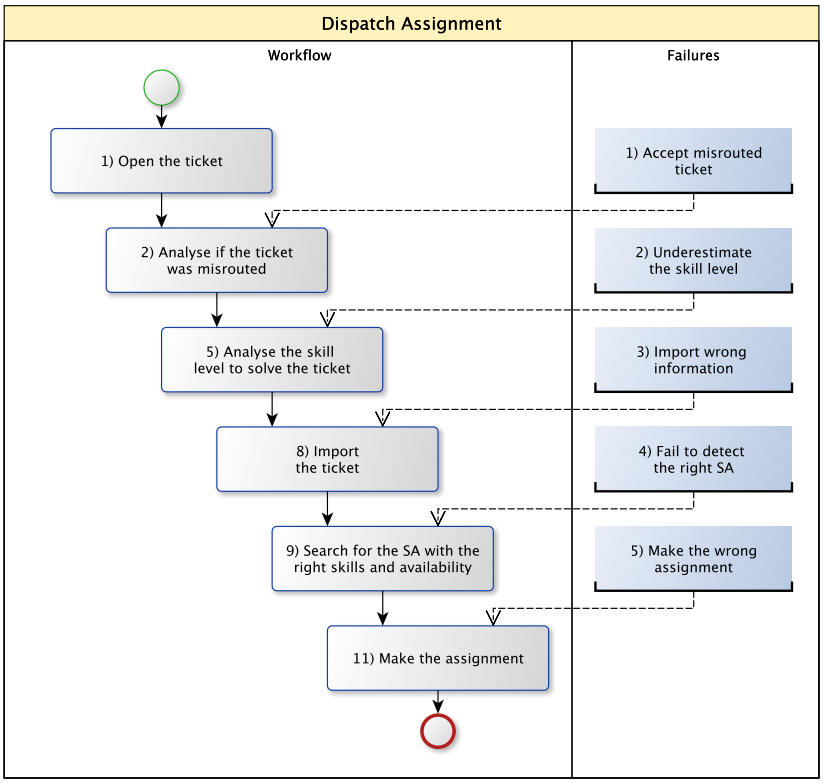

Figure 4.4: Common failures introduced by dispatchers in Service Desk and corresponding workflow task

This generic task type has a nominal human error probability (HEP) of 0.0004, with the $5th$ and $95th$ percentile bounds respectively of 0.00008 and 0.009. For each task, was also selected the EPCs that present a negative effect on the outcome and them calculated the $HEP[final]$ according to equation 3.3, discussed in Section 3.3. Table 4.1 illustrates HEART calculations for failure 2 (*i.e.*, underestimates the skill level) which occurs in task 5.

Based on these values, the total assessed EPC effect is $(3.51 * 5.02 * 1.18 * 2.28) = 74.41$ and the assessed human error probability is $(0.0004) * 74.41 = 0.029$. Therefore the probability of occurring a human error in Task 5 is 2.9%. Similar calculations may be performed to calculate the predicted $5th$ and $95th$ percentile bounds, which in this case would be 0.0059 and 0.66. As a total probability of failure can never exceed 1.00, if the multiplication of factors takes the value above 1.00, the probability of failure has to be assumed to be 1.00 and no more.

Table 4.1: HEART calculations for task 5 – without mashups

EPCs	HEART effect	Assessed proportion of effect	Assessed effect
Channel overload	x 6	Medium	$(6-1)*0.5032+1=3.51$
Suppressing information	x 9	Medium	$(9-1)*0.5032+1=5.02$
No veracity checks	x 2	Low	$(2-1)*0.1833+1=1.18$
No means of conveying info	x 8	Low	$(8-1)*0.1833+1=2.28$

By following the HEART methodology for the remaining tasks, it was possible to get the failure probabilities presented in Table 4.2. With these values it was possible to employ the Fault Tree Analisys (FTA) method in order to estimate the overall workflow failure rate. This failure rate was obtained using the equation 3.4, discussed in Section 3.3 and represented bellow.

Table 4.2: Original failure probabilities

i	Task	$HEP[final_i]$
1	3	0.002391857
2	5	0.019094864
3	8	0.022549899
4	9	0.019094864
5	11	0.002391857

In Equation 4.1, $P[failure]$ represents the probability of failure in a sequence of events and $HEP[final_i]$ is the final probability of human error for a specific task, which is obtained from Equation 3.3. By taking this equation, was possible to obtain a $P[failure]$ of 0.06401, which means that there is a chance of 6.4% for the dispatcher to make at least one error during the execution of the dispatch process. This probability of 6.4% is comparable to the number of human errors accounted from incident reports obtained from the service delivery center investigated in this study.

$$P[failure] = 1 - \prod_{i=1}^{n}(1 - HEP[final_i]) \tag{4.1}$$

4.3 Applying Mashups to Dispatch

In the context of Request Fulfillment, mashups are a new technology that simplifies and makes more predictable the creation and execution of dispatching systems, focusing on the activities performed by each dispatcher, thus, decreasing the possibility of errors and helping them to improve the efficiency of the assignment. This section demonstrates the application of the mashup patterns and error prevention modules, introduced in Chap-

ter 3, to improve the dispatcher's performance by eliminating inefficiencies and increasing reliability.

4.3.1 Mashup Patterns

In Chapter 3, a set of four mashups patterns have been introduced and discussed. These patterns represent an effective solution to create a mashup-based application for the above-mentioned dispatching scenario. By using them, the main objective is to improve dispatchers assignment performance by automating some tasks, and by implementing the single pane of glass concept. By definition, this concept states that all the required information a human operator may need to achieve a goal should be presented into a single screen (*e.g.*, a single management console) with the data already filtered, transformed, and integrated.

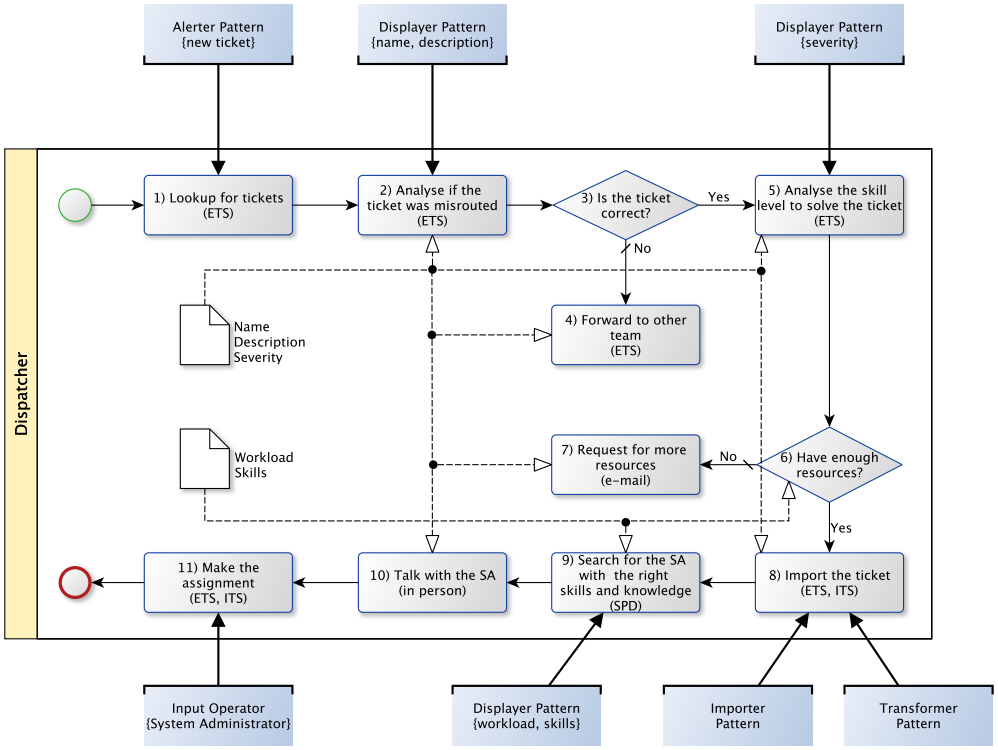

Figure 4.5: Relating mashup patterns with dispatching tasks.

Figure 4.5 shows how the proposed patterns relate with the dispatching tasks. Considering that name, description, and severity are the basic information from a ticket that a dispatcher uses most frequently to make the assignments, the *Displayer* pattern can be used to show all the required information into a single screen. This pattern can also be used to display the system administrator's workload and skills. Finally, with the *Displayer* pattern, it is possible to eliminate both *information lookup*, and *retaining information* inefficiencies that may appear during the execution of this task.

The nature of a service desk resembles to a push model, where an unpredictable number of requests from the customers are created and forwarded to dispatchers, who should be constantly monitoring their ticketing systems for new requests. In this scenario, the *Alerter* pattern can be used to notify dispatchers about new tickets as soon as they are created. Considering that dispatchers are ready to process a new request (*i.e.*, they do not

leave their work stations), this patterns helps to eliminate the *becoming aware* inefficiency. The *Importer* pattern is used in this case study to automate the task of importing tickets to the internal database, and the *Transformer* pattern is applied when the dispatchers need to modify (*e.g.,* augment, exclude) some information, for example, filtering confidential data (*e.g.*, phone numbers) on the ticket's description.

4.3.2 Mashup Operators

Preventing human errors from affecting the system is fundamental in IT service provider organizations. Errors in such environment may lead to financial loss, when SLAs are not satisfied, or to service outages, when servers are mistakenly configured. Besides productivity, the mashup patterns used previously can also be used to increase the dispatchers' reliability. In conjunction with the new set of mashup basic operators (*i.e.*, *Buffer* and *Forcing*), discussed in Section 3.3, the patterns enable mashup developers to create new applications that ultimately prevent human errors from occurring, or at least reduce their frequency.

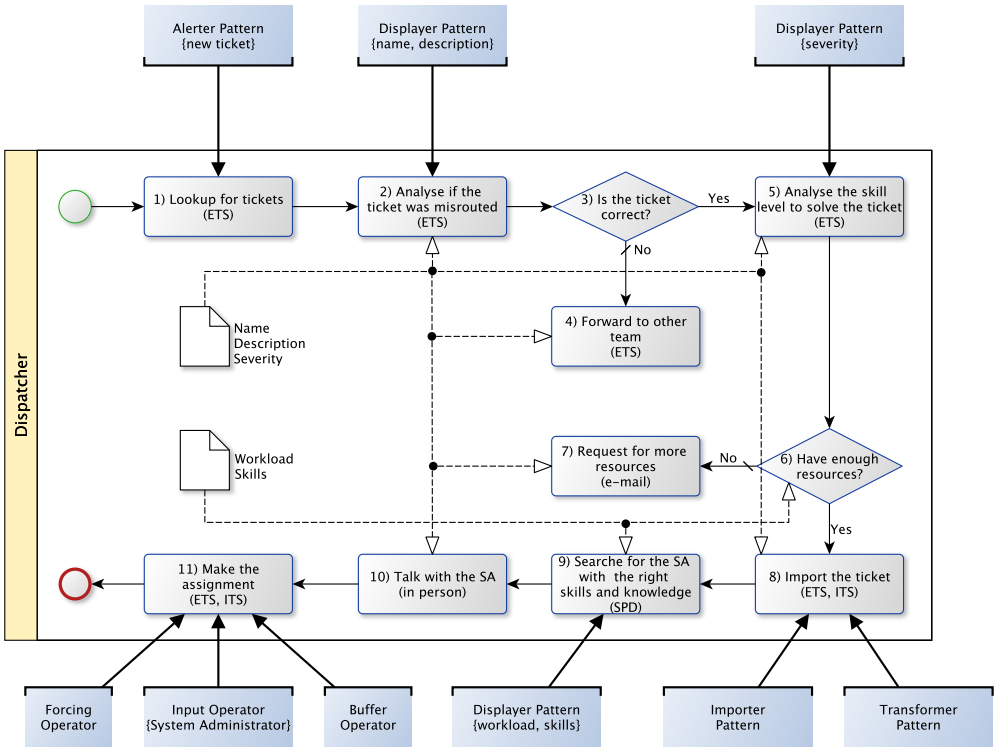

Figure 4.6: Relating mashup patterns and error prevention modules with dispatching tasks

Figure 4.6 shows how the mashup patterns and proposed error prevention modules relate with each specific task in the dispatching scenario. The **displayer pattern** aids to reduce the occurrence of **retrieval errors**, either from memory or from a visual display. The combined use of both *importer* and *transformer* patterns is useful to avoid **action** and **retrieval errors**. Even with the introduction of these patterns, the dispatcher may still make some error. In order to reduce this probability even more, the two error prevention modules introduced in this thesis are used just after the assignment task (dispatching task 11). If the dispatcher mistakenly selects the wrong SA to assign the ticket, the system will require a confirmation to execute the assignment task. Even if the dispatcher confirms that, he still has a few seconds to cancel the operation before it is executed.

4.3.3 Mashup-based Dispatch System

Although map mashups are currently the most widely used mashup category on the Web, mashups can also be created to integrate disparate data into a common dashboard. Such amalgamation of multiple systems in a single pane of glass is critical in the area of IT Service Management, in particular in the area of IT Operations. The mashup developed during this thesis replaces the multiple systems and tools used by dispatchers during the assignment process. It was constructed based on the proposed mashup patterns and error prevention modules discussed previously.

The developed mashup allows dispatchers to access information regarding customer requests (*i.e.*, tickets) and to assign those tickets to the appropriate system administrators. The mashup retrieves from the Internal Ticketing System (ITS) the system administrator's workload, while skills and expertise are recovered from the Service Provider's Directory (SPD). The mashup also retrieves ticket data from multiple External Ticketing Systems (ETS), such as Remedy [1] and ManageNow [2]. The ticket data is adapted before being imported into the ITS and the most relevant information is filtered for presentation. To build the mashup, the following interaction elements were employed:

- **Forcing** – This forcing element obligates the dispatcher to re-confirm the assignment – through an alert message – before executing it;
- **Buffer** – The buffer element was used to avoid the occurrence of human errors by delaying the assignment action in 10 seconds;
- **Visual** – Different visual elements were used in this mashup to display the dashboard consisting of both tickets and SAs data;
- **Execute** – The execute element is responsible for the periodic verification of new service requests without the explicit action of the dispatcher;
- **Adaptation** – Adaptation elements were employed in the mashup logic to hide technical details related to the access of external systems;
- **Transform** – Transform elements were used throughout the mashup logic to filter the retrieved data for different purposes;
- **Input** – An input element was used to allow the dispatcher to specify the system administrator who will be responsible for solving the ticket.

Figure 4.7 shows the composition of the mashup-based dispatch system, rendered in Business Process Modeling Notation (BPMN). First, the dispatcher enters his team identification number. Then, an adaptation element (*i.e.*, *Retrieve SA List*) accesses the SPD and retrieves the list of system administrators associated with the provided team number. A second element of this kind (*i.e.*, *Retrieve SA Workload*) retrieves the system administrators' workload, while a third element (*i.e.*, *Retrieve SA Data*) accesses the SPD to retrieve user profile data (*i.e.*, skills and expertise). Once the SA information is obtained, it is aggregated and displayed using the visual element *Display Table*. The *Display Widget* was used to present the ticket's severity, description, and title, which are fundamental

[1] http://www.bmc.com/products/remedy-itsm/
[2] http://www.support.managenow.info

for dispatchers during the assignment task. Finally, the visual element *Display Alert*, was used to display an alert once a new ticket arrives for the dispatcher.

The wrappers (visually represented as retrieve elements in Figure 4.7) have been developed as standalone applications. Thus, they can also be used by other composition systems that use a SOAP-based wrapper model or even a Java-based model, since the implementation under the SOAP interface has been coded in Java. In addition, wrappers have been designed with access interfaces that allow their internal implementation to be modified without affecting the composition logic. For example, the implementation of a wrapper can be modified in case of changes on the customer's ticketing system.

The runtime environment, presented in Figure 4.8, shows the mashup front-end, where both tickets and system administrators' information are presented in a single screen to improve performance in dispatch.

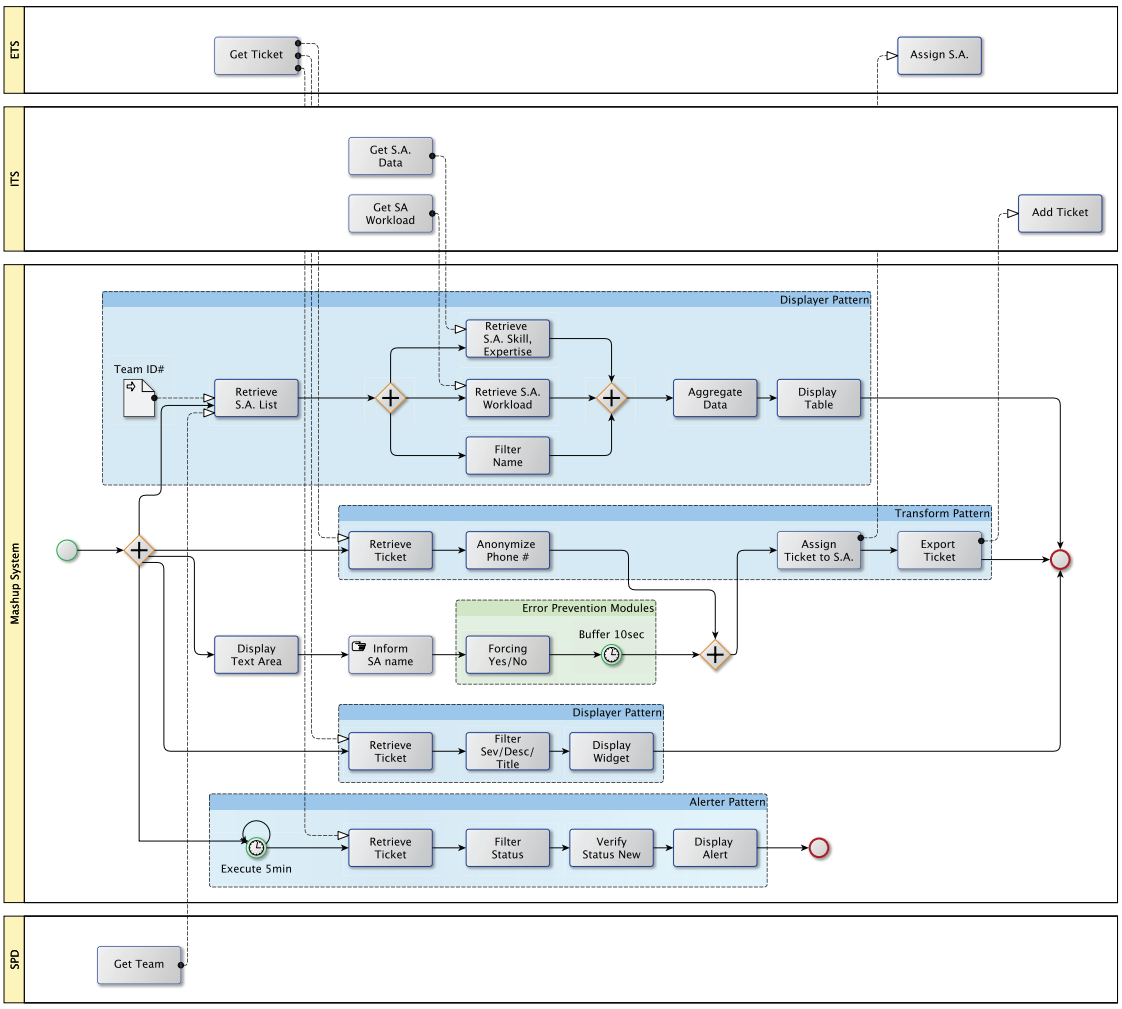

Figure 4.7: Mashup-based dispatch system design

The information aggregated on the mashup allow a dispatcher to directly visualize the most important data about the service requests provided by the customers (*i.e.*, ticket description and severity) as well as all the information related to his team of system administrators (*i.e.*, workload, skills, and expertise) – respectively items 1 and 2 in Figure 4.8. Proceeding through a few interaction steps, mainly mouse clicks on elements representing tickets or system administrators, allows the dispatcher to observe more detailed information (*e.g.*, contact info). In the same screen, the dispatcher also has the option to execute

to the assignment task – item 3. Notifications about new tickets are displayed on the top-right corner of the page – item 4 – allowing the dispatcher to be constantly aware of the incoming requests.

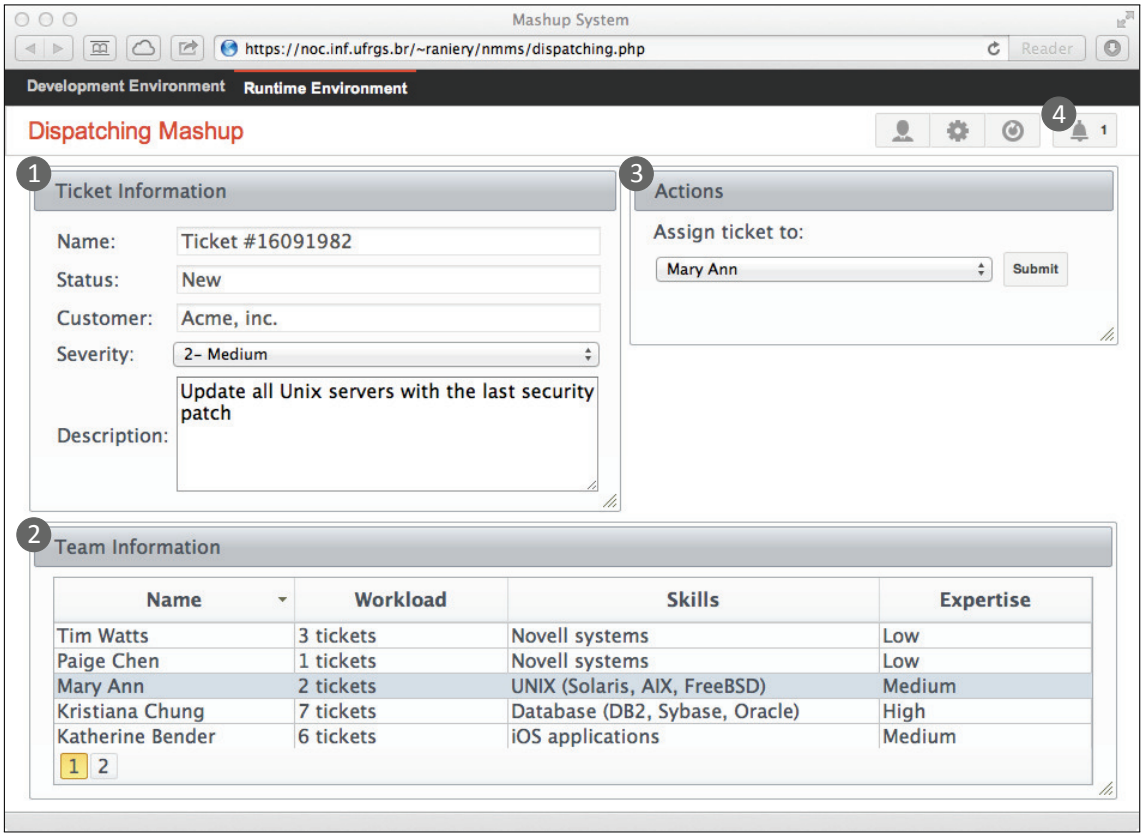

Figure 4.8: Graphical User Interface (GUI) of the mashup for the dispatching scenario

New wrappers can be introduced into the composition logic, thus supporting the access of additional external ticketing systems. Once modified, it is necessary to save the mashup and load it again. The page itself is not saved, since it is dynamically created. Instead, a mashup meta-description is stored. When a mashup is loaded, the mashup description is recovered and the composition is re-executed, rebuilding the mashup with updated information. Further explanation about mashups execution is beyond the scope of this thesis and can be found in (BEZERRA *et al.*, 2010).

4.4 Predictions on Mashup Usage

This section analyses the performance improvements obtained with the application of the mashup-based dispatching system, previously presented, on the request fulfillment process. Through the analysis it was possible to find an optimally modified process, along with the expected productivity and reliability improvements.

4.4.1 Productivity

The combined model, introduced in Section 3.2.3.3, is constructed in 6 stages. These stages allow an analyst to systematically examine different levels of inefficiencies occurred during the execution of the assignment process.

After the tasks and subtasks of the process are determined (stage 1), the next step

towards the construction of the combined model corresponds to the measurement of the time to perform each subtask (stage 2). Then, the analyst works with a domain expert to determine the complexity metrics for the Complexity model (stage 3). By interviewing dispatchers, it was possible to determine values for all the memory and decision metrics related to each task of the assignment process. As discussed in Section 3.2.3.3, to avoid double-counting, all the complexity metrics are discarded, except the memory and decision metrics, reducing the set of complexity metrics to: *Decision nBranches*, *Decision gFactor*, *Decision cFactor*, *Decision vFactor*, *Memory Size*, *Memory Latency*, and *Memory Depth*. It is important to remark that the decision metrics *gFactor*, *cFactor*, and *vFactor* are quantified with a three-level scale of increasing complexity. For the dispatching process analysis, it was considered that these three metrics have a value of 1 and there is only one role (*i.e.*, the dispatcher) involved in the execution of the process. According to Equation 2, provided in (DIAO; KELLER, 2006), this leads to a reduction of the decision complexity score to the number of decision branches in the task.

Figure 4.9 summarizes all the accounted metrics and their respective values. These results show the large number of information a dispatcher needs to keep in memory while executing the assignment process (as can be observed from the *MemorySize* metric). In the worst case, the dispatcher needs to retain at least 10 items, which exceeds in 3 the usually accepted number of disparate information that a human may retain in the short-term memory (MILLER, 1956).

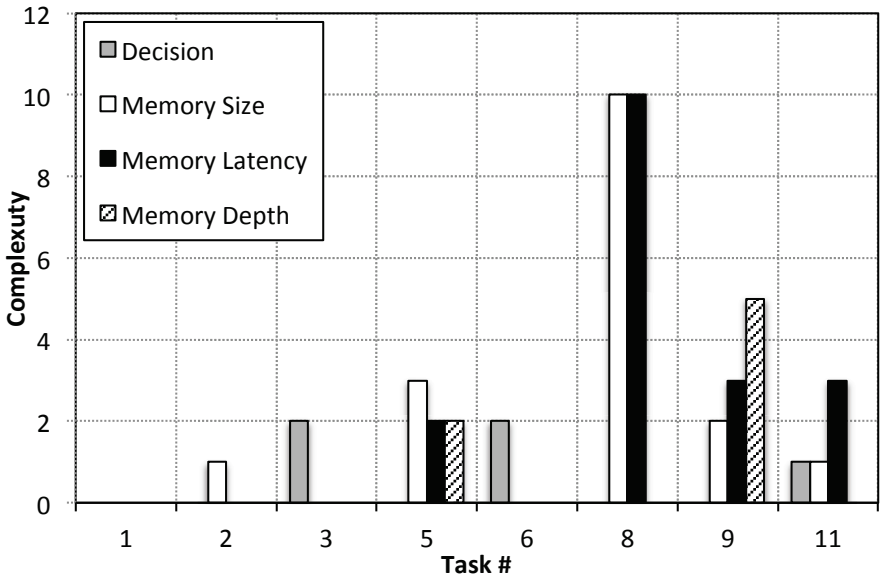

Figure 4.9: Complexity metrics for dispatch process

Once the complexity metrics are determined, the analyst derives the Complexity model coefficients from the time measurements (stage 4). It can be achieved by using Multiple Linear Regression (MLR) (HARRELL, 2001). This technique is used to create a linear equation (*i.e.*, a model), aimed to correlate the time spent in each assignment task and the complexity metrics obtained from the previous step. To determine which metrics should be used in the quantitative analysis, it is required to evaluate how the model's quality changes as new metrics are included in the evaluation. To aid this choice, two metrics can be employed: the R^2 and the Root Mean Square Error (RMSE). The prior refers to the variability of a given dataset that can be represented by a quantitative model. It can be

calculated using Equation 4.2:

$$R^2 = 1 - \frac{var(y - \widehat{y})}{var(y)} \tag{4.2}$$

In Equation 4.2, $var()$ denotes the variance, y represents the measured time and $\widehat{y}$ represents the predicted time. A R^2 equals to 1 means that the model is able to capture all the variability and a R^2 equals to 0 means the model is not able to capture any variability. RMSE, in turn, is used to compare the differences between the real measurements and the values predicted by the model. RMSE values close to 0 are preferred, since it represents the accuracy of the quantitative model. The RMSE can be calculated using the following equation:

$$RMSE = \sqrt{\frac{1}{k}\sum_{k=1}^{K}(y(k) - \widehat{y}(k))^2} \tag{4.3}$$

A model with higher R^2 values and lower RMSE values represents a good fit between the employed complexity metrics and the execution time of individual tasks (DIAO; KELLER, 2006). On the other hand, a low quality model can indicate problems occurred during the determination of complexity metrics or an insufficient analysis of the human behavior. Table 4.3 shows the evolution of R^2 and RMSE metrics as new complexity metrics are incorporated into the model. For example, at stage 2 the model is composed by two complexity metrics (*i.e.*, Decision and Memory Size) and presents a R^2 value of 0.94, a RMSE of 24.78 seconds, and does not present any negative B_i.

Table 4.3: Quality evolution of the complexity model as new metrics are added

Step	Added metric	R^2	RMSE	B_i negative
1	Decision	0.01	39.09	no
2	Memory Size	0.94	24.78	no
3	Memory Latency	0.96	4	no
4	Memory Depth	0.84	5.04	no

When analyzing Table 4.3, it is possible to observe that R^2 increases as new complexity metrics are included in the evaluation and, at the same time, RMSE decreases. However, as can be observed, the R^2 value is affected mainly by the addition of the first three complexity metrics. With the inclusion of the last metric (step 4), the R^2 value decreases and the RMSE increases, thus, the creation of the quantitative model should stop at step 3, which represents the best R^2 and RMSE values. Considering only the complexity metrics up to step 3, the resulting linear equation is:

$$y = 3.2029 * ComplexityMetric + 2.7732 \tag{4.4}$$

By using this equation, it was possible to predict the time (spend at each task) associated with the complexity encountered in carrying out the tasks of the assignment process. These predicted times and the real measurements are presented in figure 4.10. The results show that the obtained and estimated times present a close fit, with a R^2 error of 0.96, which means the model can explain 96% of the time variability. The obtained value for RMSE was 4, which shows the accuracy of the proposed model.

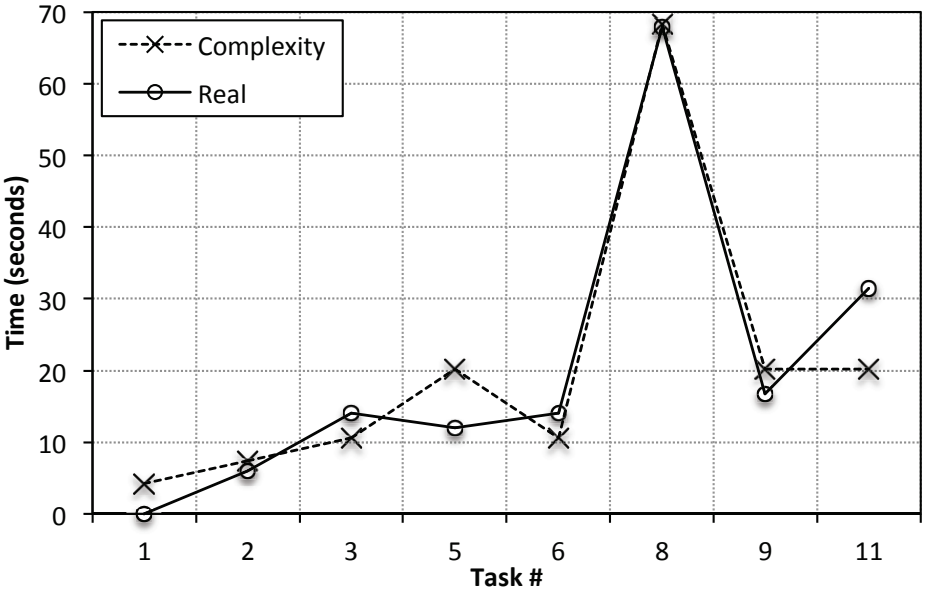

Figure 4.10: Predicted and real times associated with the complexity of the assignment process

As can be observed in Figure 4.10 the model was able to predict a peak at task 8, related to the action of copying manually data from the external ticketing system to the internal one. This action requires the dispatcher to handle multiple information, such as ticket's severity, complexity, and description. Once the complexity times are obtained, the next step in the analysis is to introduce the predictions obtained from the KLM model (stage 5). Figure 4.11 presents the times obtained by using KLM and the complexity model (stage 6). These represent both higher level inefficiencies due to the complexity of the activity itself, and lower level inefficiencies due to the mechanical execution involved in performing the activity.

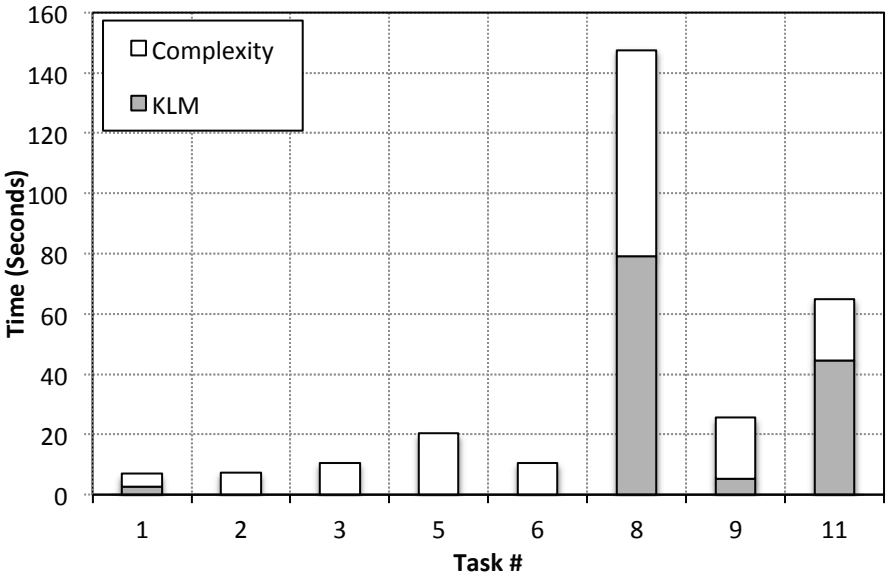

Figure 4.11: Predicted times from the complexity and KLM models

Once the analysis of the dispatching scenario is concluded and the time measures are estimated, it's possible to evaluate productivity enhancement by using the mashup's technology. Figure 4.12 presents the predicted productivity enhancement. The bars indicate the labor cost (*i.e.*, time) for each task of the dispatching process. The different colors indicate the obtained and estimated (with and without mashups) times for the dispatching process.

For the mashups predictions scenario, it were used the following complexity metrics values: 0, 1, 2, 2, 2, 0, and 1. The absence of tasks 4, 7, and 10, in Figure 4.12, is a consequence of unobserved actions during the evaluations. The tickets weren't routed to wrong dispatchers and always had system administrators available to solve customer's requests. It is important to observe that this graph does not represent the awareness time since the KLM or the Complexity Model cannot predict it. The results also show the significant time reduction of 64.42%, which mostly was due the automation of step 8, by using the *Importer* pattern. The *Displayer* pattern allows the reduction of the tasks 9 and 11, since the dispatcher does not need to look up any information in a different system, thus eliminating all the possible keystrokes, and also does not need to remember any information from a previous task, helping to decrease the memory complexity. Real measurements with mashups were not observed because of restrictions in the evaluated service delivery organization to adopt new tools.

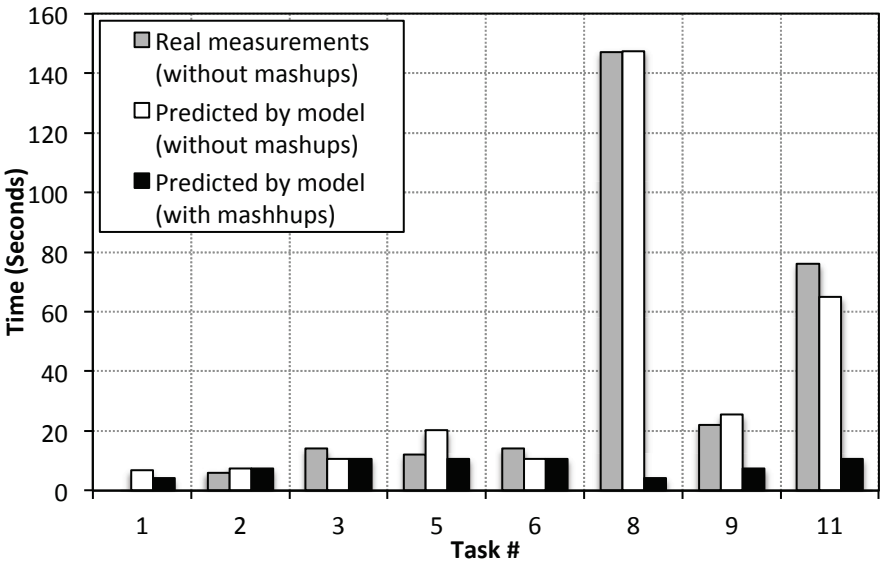

Figure 4.12: Quantitative model validation with predictions of mashup cost

4.4.2 Reliability

The introduction of new interaction elements, *Buffer* and *Forcing*, in conjunction with a set of mashup patterns enable mashup developers to create new applications that ultimately prevent human errors from occurring, or at least reduce their frequency. In Chapter 3 it was possible to observe two ways to prevent human errors: avoidance and interception. Error avoidance (*i.e.*, keeping people from making errors) was possible through the use of mashup patterns and the *Forcing* module. Error interception (*i.e.*, stopping the errors from reaching the system) was possible through the *Buffer* module.

Once the process workflow (Figure 4.2) and root causes of problems (Figure 4.4) are

determined, the next step in the reliability assessment is the determination of Human Error Probability (HEP) of each activity in the workflow. Again, each task of the assignment process have been classified as Type G. Table 4.4 illustrates the calculations of HEART technique for task 5, however, considering the scenario where mashups technology is employed to improve dispatchers reliability.

Table 4.4: HEART calculations for task 5 – with mashups

EPCs	HEART effect	Assessed proportion of effect	Assessed effect
Suppressing information	x 9	Low	$(9-1)*0.1833+1=2.46$
No veracity checks	x 2	High	$(2-1)*0.8316+1=1.83$
No means of conveying info	x 8	Low	$(8-1)*0.1833+1=2.28$

Based on these values, the total assessed EPC effect is $(2.46*1.83*2.28)=10.31$, thus the human error probability in task 5, with the employment of mashups, is $(0.0004)*10.31=0.0041=0.41\%$. The individual HEP values for all the tasks in the assignment process are presented in Table 4.5. Each line of this table represents a specific task of the dispatching process. Since task 8 was completely automated by employing mashup patterns, the human error probability on this task is 0%.

Table 4.5: Individual HEP values for the assignment process

Failure	Task	HEP
1	3	0.00091324
2	5	0.00412552
3	8	0.00000000
4	9	0.00476030
5	11	0.00091324

The next step in the reliability assessment involves the construction of an Event Tree, based on the individual HEP values, to analyze the overall workflow performance, *i.e.*, discover what is the exact probability to occur at least one human error during the process execution. The Event Tree enables an analyst to evaluates risk by tracing backwards in time or backwards through a cause chain. By using equation 3.4, it was possible to obtain a $P[failure]$ of 0.011004691, which means that the use of mashups can reduce to 1.1% the chance of dispatchers to make at least one error during the execution of the dispatch process.

Figure 4.13 presents a comparison between the failure probabilities of individual tasks before and after the employment of mashups. As it can be observed, the complete automation of task 8 (corresponding to failure 3) through the *Importer* pattern is the most responsible for the reduction in the probability of human error in the assignment process. The usage of the *Displayer* pattern contributed to reduce the chance of failure in the execution of tasks 5 and 9 (corresponding to failures 2 and 4 respectively). In both tasks,

the dispatcher no longer needs to look up for any information in a different system, thus eliminating all the possible retrieval errors. Although was not observed the reduction of Knowledge-based errors (*i.e.*, decision errors), the presented patterns and error prevention modules avoided the occurrence of Rule-based and Skill-based errors (*i.e.*, action and retrieval errors), which represent 89% of human errors according to previous investigations (HUMAN ERROR: MODELS AND MANAGEMENT, 2000).

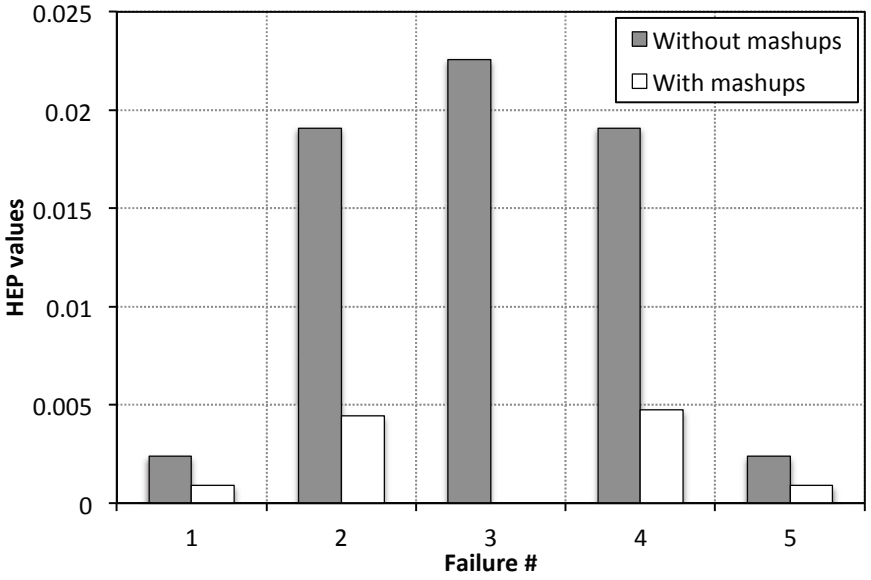

Figure 4.13: Human error probabilities comparison

4.5 Correlation Analysis

The single observation of the productivity or the reliability, although accurate, does not provides a clear view about performance improvements achieved with the usage of the mashups technology. In order to have a better perspective on this issue, this section analyses the total performance impact over ITSM processes in a systematic way.

The correlation between productivity and reliability can be observed through the Pearson product-moment correlation coefficient r, with values ranging from -1 and 1. A value of 0 indicates that there is no association between the two variables. A value greater than 0 indicates a positive association; that is, as the value of one variable increases, so does the value of the other variable. A value less than 0 indicates a negative association; that is, as the value of one variable increases, the value of the other variable decreases. Analyzing the individual tasks of the dispatching process (with and without mashups), it was possible to observe r values of 0.33 and 0.30 respectively. These results indicate a fair and positive relationship among the variables, which means that the reduction of the execution time (*i.e.*, increasing productivity) also contribute to the reduction of human errors (*i.e.*, increasing reliability).

In order to identify what are the key characteristics of individual tasks that relate to the investigated performance aspects (*i.e.*, productivity and reliability), it is necessary to observe both metrics together [3]. To this end, based on the results previously discussed in

[3] Although its not the scope of this thesis, one consolidated number for performance could be developed in order to simplify the comparison between different processes.

this Chapter, Figure 4.14 shows the evolution of the error probability over time. The marks represent the instant of time when each task is concluded and the corresponding error probability. In this case, curves with marks closer to the origin have a greater reliability and productivity, thus a better performance. As can be observed, the overall performance of the process with the usage of mashups is significant greater than the original process (*i.e.*, without mashups).

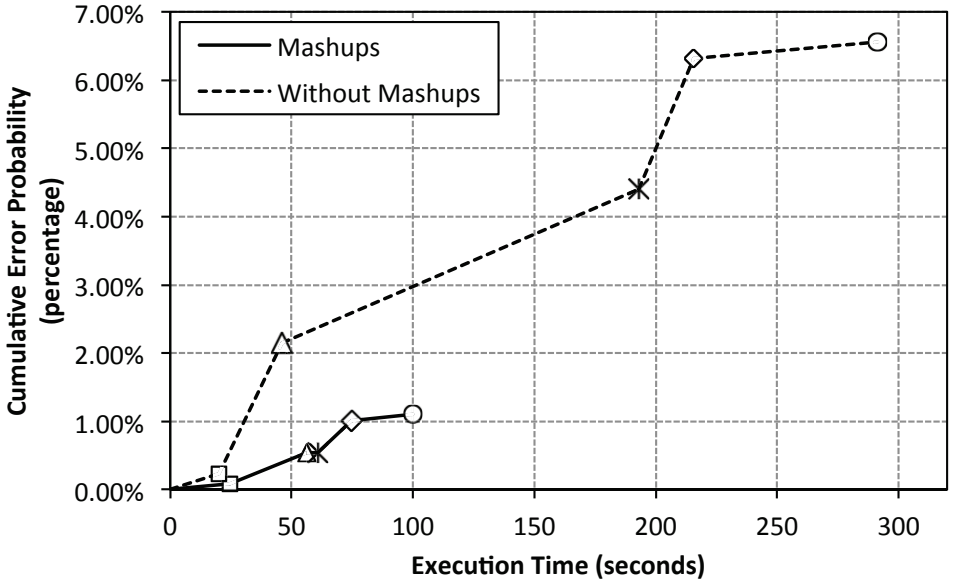

Figure 4.14: Relating task's execution time and error probability. Task 2: □; Task 5: △; Task 8: ∗; Task 9: ◇; Task 11: ○

The first observation from Figure 4.14 is that tasks 5 and 9 present a very similar behavior for both scenarios (with and without mashups), *i.e.*, the human error probability increases fast in a relative short period of time. The execution of these two tasks requires a high level of cognition, whereas involve few mechanical actions. Even in an optimized process – obtained with the application of mashups – such behavior remains. This can be explained by the fact that dispatchers are usually very familiar with the dispatching environment (*i.e.,* teams, customers, and service requests), thus they tend to analyze and assign tickets to system administrators quickly. Such complex decisions, however, are highly susceptible to human error. The number of disparate information – regarding tickets and SAs – a dispatcher needs to remember and analyze before making the ticket's assignment is significant. This confirms the results demonstrated in (MATTHEWS *et al.*, 2000), where humans are prone to slip and lapse with familiar tasks. The analysis of task 11, on the other side, demonstrates almost no increasing in the probability of error, even requiring a significant amount of time to be executed. This task involves simple mechanical actions, basically switching between systems and providing a single information (*i.e.,* the system administrator's name). It is important to note that, besides task 8 also involves mechanical actions, the dispatcher needs to remember more information regarding the tickets, such as its severity, description, and name. This leads to the conclusion that mashups are more suited to reduce errors in cognitive actions, and to increase the productivity in mechanical actions.

4.6 Final Remarks

This chapter demonstrated the application of mashups in the Request Fulfillment process. Based on observations of a global IT Service Delivery organization it was possible to determine the workflow of activities performed during the dispatch assignment process. In order to detect performance issues and perform a root cause analysis, several dispatchers were shadowed and interviewed during their daily activities. Finally, the proposed models were employed with the objective to assess performance improvements obtained by using mashups technology.

In the development environment, the IT operator (when acting as mashup developer) sets well know parameters (*i.e.*, team identification number) so that the integration can be performed automatically by the underlying engine. The runtime environment displays the results of the integration on a dashboard style dynamic Web page. The popularity of dashboards in IT organizations is a strong indication of their benefits, which the developed mashup system inherits. The substantial decrease in the number of interaction steps and interfaces involved also contributes to the overall usability improvement of the prototype over traditional methods.

The improvements in productivity and reliability demonstrate the viability of the mashups technology as a means for improving IT Service Management. Analysis of the tasks performed by a group of human operators enabled the discovery of common inefficiencies in their implementation of the ITIL Request Fulfillment process. Since these inefficiencies can be found in many different ITSM scenarios, mashup patterns can be applied as proven and reusable solutions. The construction of the mashup for the dispatching scenario enabled the observation of a good fit between the times predicted by the proposed combined model with real measurements, thus indicating the feasibility of the proposed framework.

The new mashup basic operators, *Buffer* and *Forcing*, in conjunction with a set of mashup patterns enable IT operators (when acting as mashup developers) to create new applications that ultimately prevent human errors from occurring, or at least reduce their frequency. Two ways to prevent human errors were observed: avoidance and interception. Error avoidance (*i.e.*, keeping people from making errors) was possible through the use of mashup patterns and the *Forcing* module. Error interception (*i.e.*, stopping the errors from reaching the system) was possible through the *Buffer* module. Although was not observed the reduction of Knowledge-based errors (*i.e.*, decision errors), the presented patterns and interaction elements avoided the occurrence of Rule-based and Skill-based errors (*i.e.*, action and retrieval errors), which represent 89% of human errors according to previous investigations (HUMAN ERROR: MODELS AND MANAGEMENT, 2000).

5 CONCLUSIONS

The investigation carried out during this thesis demonstrated the application of the mashups technology in the IT Service Management field. Specifically, an extensive analysis to support the hypothesis that **"the employment of mashups enhances the performance of human-centered ITSM processes"** is presented. Different from most of the works proposed in the literature, where a significant part focus on the quality aspect (*i.e.*, frameworks, processes and metrics that assess effectiveness from the point of view of the receiver of such services), this thesis investigated the service provider's perspective, in particular, aspects of performance that have direct implications on the efficiency and cost of the operation from the provider's point of view. Although different metrics may be used to assess performance, the focus of this thesis was on productivity and reliability, two of the most used metrics by IT service delivery organizations to assess the performance of ITSM processes.

The analysis was conducted based on the Six Sigma principles, which focuses on understanding causes for process variability and driving to progressively achieve higher levels of consistency. Initially, an analysis of causes for both inefficiencies and human errors in service management processes was performed. Through this investigation, it was possible to observe common inefficiencies and common errors in ITSM. In a second moment, the mashups technology was investigated to identify feasible methods to tackle such problems. The concept of mashup patterns and error prevention modules – based on error avoidance and interception – represent effective approaches to obtain an improved orchestration of the process through design and automation. It is important to remark that mashup-based solutions do not aim to replace traditional management models or tools; instead mashups allow traditional technologies to be easily used to create new applications, pushing such technologies beyond their original purposes. Finally, the most comprehensive models available in the literature are combined to quantify and predict the deployment impact of the mashups solution over ITSM processes.

Through a real case study, related to the Request Fulfillment process, it was possible to examine the introduced methods to tackle performance issues that may arise during the execution of human-centered processes. The focus of this case study is on dispatch, an activity centered on human operators, called dispatchers, with knowledge of standard fulfillment procedures and responsibilities that include: monitoring for new requests, dispatching the requests to the appropriate system administrators, and monitoring compliance with Service Level Agreements. Traditionally, gathering information for assigning tickets has required the use of several tools with distinct user interfaces. Integration of the data collected has been a complex process requiring sophisticated scripts, handling of large amounts of data, and high effort to provide a suitable presentation of the results obtained. The developed mashup-based dispatching system reduced the number of

interaction levels for the user down to two user interfaces that do not require advanced technical skills to be manipulated, while also keeping the complex technical details of the integration hidden from the human operator. The observed improvements in productivity and reliability of ITSM processes provided by the application of mashups demonstrate the viability of the technology as a suitable alternative for improving IT Service Management.

Summarizing, this thesis provided a performance improvement program in the dispatch activity. Initially, key performance metrics for the environment under study (*i.e.*, productivity and reliability) were defined. Then, quantitative models were combined to observe and quantify the most relevant issues in the current environment. Finally, a set of appropriate methods (*i.e.*, mashup patterns and error prevention modules) were identified and implemented to solve root cause problems.

5.1 Answers for the Fundamental Questions

The motivation behind the definition of the fundamental questions was to define the main points to be analyzed during the investigation of the hypothesis and to establish the roadmap to achieve the contributions of this thesis. Therefore, the description below summarizes and highlights answers for each fundamental question.

Fundamental question I: *What are the major causes for a poor performance in the execution of IT Service Management processes?*

Answer: ITSM processes can be laden with segments where the human becomes a bottleneck and slows down the entire process. These inefficiencies are usually caused by insufficient design of the process itself, or defects in the used tools. They may appear at fundamentally two levels of analysis: higher level inefficiencies due to the complexity of the activity itself, and lower level inefficiencies due to the mechanical execution involved in performing the activity. In ITSM, the lower level takes into consideration the interaction of human operators with the available software tools, while the higher level tasks are related to the reasoning capabilities of the human operator. Four groups of inefficiencies (*i.e.*, basic, information management, skill-dependent, and synchronization) were identified and discussed during the investigations.

Human beings have the ability to execute multiple and complex tasks (mental and physical) at the same time. However, although skills and expertise level can vary among people, all humans eventually reach their natural limits. The continuous pressure on IT companies to increase their competitiveness forces human operators to reach or exceed their natural limits, thus becoming susceptible to making errors. These errors can occur due to a wide variety of causes, such as: attention span, memory, situation awareness, and personal resources. Based on analysis of documents related to service incidents, it was possible to identify five common error-prone activities in ITSM – *i.e.*, action, retrieval, checking, decision, and communication errors.

Fundamental question II: *How models available in the literature can be used to assess performance improvements obtained with mashups?*

Answer: Despite its importance, ITIL only provides high-level generic guidelines to IT organizations, without proposing, for example, concrete models

and methods for capturing metrics and evaluating the quality of IT processes. Such evaluation is important for IT service providers to quantify and, most importantly, to predict the deployment impact of IT solutions. In this context, the KLM model is widely used, well corroborated by experiment (see for example (KOESTER; LEVINE, 1994)), and provides a wealth of detail at the lower level of human-computer interactions, while the Complexity model addresses both levels of inefficiencies[1]. Thus, these models are the current best starting points for creating a new model to better evaluate the performance of IT processes from a time productivity perspective.

Diverse techniques to account for human errors have been developed in the last few decades, such as THERP (SWAIN; GUTTERMAN, 1980), and HEART (WILLIAMS, 1985a), whereas is considered the most comprehensive method in the field of HRA and was used in this thesis to quantify the human reliability. HEART was employed together with Event-Tree Analysis (ETA) to evaluate the overall human error probability in ITSM, since HEART itself only quantifies the reliability of individual tasks.

Fundamental question III: *What methods could be employed in the development of mashups aiming at performance enhancement?*

Answer: By using mashup basic operators (DOS SANTOS *et al.*, 2010), a user is able to specify mashups for a variety of purposes. When mashups share similar logic, however, it is often convenient to consider the employment of *mashup patterns* (MP) (OGRINZ, 2009). In the context of this thesis, these mashup patterns describe the core of solutions to recurring problems that can be frequently found in IT service management activities. Their usage allows mashups to be instantiated even more quickly by reducing mashups design and modeling. In addition, because patterns enable previously proven mashups to be reused in new scenarios, mashups patterns provide an additional level of stability. Based on the analysis of tasks performed by IT operators, it was possible to identify a set of four candidate mashup patterns to tackle inefficiencies and error-prone activities in service management processes. These patterns are: *Alerter*, *Importer*, *Transform*, and *Displayer*.

In order to prevent human errors, or at least to reduce their frequency, a new type of interaction elements, called **Error Prevention** modules (SANTOS *et al.*, 2013), was created to be used by mashup developers. Two types of **Error Prevention** module were defined: *Buffer* module and *Forcing* module. The *Buffer* module allows developers to specify for how long a specific action should be delayed before being executed, thus providing a recovery window during which an erroneous action can be canceled. The *Forcing* module, on the other hand, allows developers to introduce confirmation points in the process workflow in order to force the human operator to consciously accept them before proceeding with the mashup execution.

[1]It is important to notice that to avoid double-counting, all the complexity metrics were discarded, except the memory and decision metrics, which capture higher level potential inefficiencies not addressed by KLM.

5.2 Contributions

The contributions of this thesis can be divided into: conceptual and specific ones. Conceptual contributions could be identified due to the investigations of the literature and the experiences gathered during the development of the case study. In contrast, specific contributions are associated with individual solutions developed in this thesis. Both contributions are listed as follows.

- Conceptual contributions are:
  - Discussion of the most widely accepted frameworks for best practices and performance improvement approaches in the context of ITSM as well as their major limitations to be employed in service industries.
  - This thesis presents a survey on mashups technology. The current research status of this area is presented along with common applications and challenges for mashups in the context of IT service delivery organizations.
  - Examination of scenarios other than ITSM, where the introduced models and techniques discussed in this thesis may be applied, thus leading to performance improvements.
  - Definition of non-numeric linguistic variables to represent the values of Error Producing Conditions (EPCs) specified in the HEART technique, thus facilitating its usage.
- Specific contributions are:
  - Identification of key performance issues that may arise during the execution of ITSM processes and that may impact negatively on productivity and reliability. The objective was to identify the areas where performance improvement is possible through automation and design.
  - Combination of the most comprehensive models available in the literature to assess performance of ITSM processes. A combined model demonstrated to be a practical solution to predict the impact of employing mashups technology in ITSM processes.
  - Identification of candidate mashup patterns to tackle recurring problems in IT service management activities. These patterns also allow mashups to be instantiated even more quickly, by reducing design and modeling, and provide an additional level of stability.
  - Definition of a new category of mashup basic interaction elements, called **Error Prevention** modules, to be used by mashup developers with the objective to prevent human errors, or at least to reduce their frequency.

5.3 Future Work

The investigation developed on this thesis leads to the identification of further opportunities for research. These opportunities are described in this section as future work.

- **Validation of mashup patterns** – This thesis introduced a set of four candidate mashup patterns, which still require corroborating evidence through additional supporting cases before they reach the status of full patterns.

- **Exploration of mental activities models** – Such models could enhance the scope and accuracy of the proposed quantitative model to predict performance improvements in ITSM processes.
- **Investigate financial models** – Low productivity and reliability have a direct implication on the resolution times established in Service Level Agreements (SLAs), and, therefore, result in financial loss to service providers. A financial model could be developed to assess the loss due to broken SLAs.
- **Automation of mashup's development** – Considering that the ITSM operators have a strong knowledge of the process they perform, but may not have expertise in mashups development, the use of an automated development system may ultimately provide a significantly improved orchestration of the process.

The list of future work presented above represents the major opportunities of research that can be directly derived from the work presented in this thesis. In addition, the experiences gained with the investigation of the literature and the case study, has revealed possible research topics not directly related to ITSM process improvement, but that somehow are related to this area. Examples of such topics are: inclusion of confidentiality mechanisms in the mashup's development and investigation of rollback and exception handling mechanisms in the mashup composition logic.

REFERENCES

ACCOT, J.; ZHAI, S. Beyond Fitts' law: models for trajectory-based hci tasks. In: ACM SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 1997, New York, NY, USA. **Proceedings...** ACM, 1997. p.295–302. (CHI '97).

AHSEN, A. **Total-quality-Management**. Frankfurt am Main [u.a.]: Lang, 1996. n.16. (Schriften zum Controlling).

ALLEN, T. T. **Introduction to engineering statistics and lean sigma**: statistical quality control and design of experiments and systems. London: Springer, 2010.

ALTINEL, M.; BROWN, P.; CLINE, S.; KARTHA, R.; LOUIE, E.; MARKL, V.; MAU, L.; NG, Y.-H.; SIMMEN, D.; SINGH, A. Damia: a data mashup fabric for intranet applications. In: VERY LARGE DATA BASES, 33., 2007. **Proceedings...** VLDB Endowment, 2007. p.1370–1373. (VLDB '07).

ANDERSON, S. W.; BAGGETT, L. S.; WIDENER, S. K. The Impact of Service Operations Failures on Customer Satisfaction: evidence on how failures and their source affect what matters to customers. **Manufacturing & Service Operations Management**, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v.11, n.1, p.52–69, Jan. 2009.

ANEROUSIS, N.; DIAO, Y.; HECHING, A. Elements of system design optimization in service quality management. In: NOMS, 2010. **Anais...** [S.l.: s.n.], 2010. p.48–55.

ANTONY, J. Six sigma for service processes. **Business Process Management Journal**, [S.l.], v.12, n.2, p.234–248, 2006.

ANTONY, J.; ANTONY, F. J.; KUMAR, M.; CHO, B. R. Six sigma in service organisations: benefits, challenges and difficulties, common myths, empirical observations and success factors. **International Journal of Quality and Reliability Management**, [S.l.], v.24, n.3, p.294–311, 2007.

APTE, U. M.; GOH, C.-H. Applying lean manufacturing principles to information intensive services. **IJSTM**, [S.l.], v.5, n.5/6, p.488–506, 2004.

AUMüLLER, D.; RAHM, E. Caravela: semantic content management with automatic information integration and categorization (system description). In: EUROPEAN CONFERENCE ON THE SEMANTIC WEB: RESEARCH AND APPLICATIONS, 4., 2007, Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2007. p.729–738. (ESWC '07).

BANERJEE, N.; DASGUPTA, K.; MUKHERJEA, S. Providing middleware support for the control and co-ordination of telecom mashups. In: MNCNA, 2007. **Anais...** ACM, 2007. p.11.

BARASH, G.; BARTOLINI, C.; WU, L. Measuring and Improving the Performance of an IT Support Organization in Managing Service Incidents. In: BDIM'07, 2007. **Anais...** [S.l.: s.n.], 2007. p.11–18.

BARESI, L.; GUINEA, S. Consumer Mashups with Mashlight. In: SERVICEWAVE'10, 2010. **Anais...** [S.l.: s.n.], 2010. p.112–123.

BARROS, A. P.; DUMAS, M.; HOFSTEDE, A. H. M. ter. Service Interaction Patterns. In: BUSINESS PROCESS MANAGEMENT, 2005. **Anais...** [S.l.: s.n.], 2005. p.302–318.

BAXTER, G.; ROOKSBY, J.; WANG, Y.; KHAJEH-HOSSEINI, A. The ironies of automation: still going strong at 30? In: EUROPEAN CONFERENCE ON COGNITIVE ERGONOMICS, 30., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p.65–71. (ECCE'12).

BENSON, T.; SAHU, S.; AKELLA, A.; SHAIKH, A. A first look at problems in the cloud. In: USENIX CONFERENCE ON HOT TOPICS IN CLOUD COMPUTING, 2., 2010, Berkeley, CA, USA. **Proceedings...** USENIX Association, 2010. p.15–15. (HotCloud'10).

BERGMAN, M. **More Structure, More Terminology and (hopefully) More Clarity**. 2007.

BEZERRA, R. S.; SANTOS, C. R. P. dos; BERTHOLDO, L. M.; GRANVILLE, L. Z.; TAROUCO, L. M. R. On the feasibility of Web 2.0 technologies for network management: a mashup-based approach. In: NOMS, 2010. **Anais...** [S.l.: s.n.], 2010. p.487–494.

BHATTARAI, S.; ZHAO, Z.; CRESPI, N. Consumer mashups: end-user perspectives and acceptance model. In: INTERNATIONAL CONFERENCE ON INFORMATION INTEGRATION AND WEB-BASED APPLICATIONS & SERVICES, 12., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p.930–933. (iiWAS '10).

BOLTON, R. N.; LEMON, K. N.; BRAMLETT, M. D. The Effect of Service Experiences over Time on a Supplier's Retention of Business Customers. **Manage. Sci.**, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v.52, n.12, p.1811–1823, Dec. 2006.

BROWN, A. B.; HELLERSTEIN, J. L. An approach to benchmarking configuration complexity. In: IN PROCEEDINGS OF THE 11TH ACM SIGOPS EUROPEAN WORKSHOP, 2004. **Anais...** [S.l.: s.n.], 2004.

BROWN, A. B.; KELLER, A. A Best Practice Approach for Automating IT Management Processes. In: NOMS, 2006. **Anais...** IEEE, 2006. p.33–44.

BROWN, A. B.; KELLER, A.; HELLERSTEIN, J. L. A Model of Configuration Complexity and its Application to a Change Management System. In: IN PROCEEDINGS OF THE 9TH INTERNATIONAL IFIP/IEEE SYMPOSIUM ON INTEGRATED MANAGEMENT, 2005. **Anais...** [S.l.: s.n.], 2005.

CANDEA, G.; KICIMAN, E.; KAWAMOTO, S.; FOX, A. Autonomous recovery in componentized Internet applications. **Cluster Computing**, Hingham, MA, USA, v.9, p.175–190, April 2006.

CANNON, D.; WHEELDON, D. **ITIL – Service Operation**. [S.l.]: TSO, 2007.

CARD, S. K.; NEWELL, A.; MORAN, T. P. **The Psychology of Human-Computer Interaction**. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 2000.

CATER-STEEL, A. **Information Technology Governance and Service Management**: frameworks and adaptations. Hershey, PA: Information Science Reference - Imprint of: IGI Publishing, 2008.

CHENG, W.; PORTS, D. R. K.; SCHULTZ, D.; POPIC, V.; BLANKSTEIN, A.; COWLING, J.; CURTIS, D.; SHRIRA, L.; LISKOV, B. Abstractions for usable information flow control in Aeolus. In: USENIX CONFERENCE ON ANNUAL TECHNICAL CONFERENCE, 2012., 2012, Berkeley, CA, USA. **Proceedings...** USENIX Association, 2012. p.12–12. (USENIX ATC'12).

CHOWDHURY, N. M. M. K.; BOUTABA, R. Network virtualization: state of the art and research challenges. **Communicatons Magazine**, Piscataway, NJ, USA, v.47, n.7, p.20–26, July 2009.

CHOWDHURY, S. R.; CHUDNOVSKYY, O.; NIEDERHAUSEN, M.; PIETSCHMANN, S.; SHARPLES, P.; DANIEL, F.; GAEDKE, M. Complementary assistance mechanisms for end user mashup composition. In: WWW (COMPANION VOLUME), 2013. **Anais...** [S.l.: s.n.], 2013. p.269–272.

DIAO, Y.; KELLER, A. Quantifying the Complexity of IT Service Management Processes. In: DSOM, 2006. **Anais...** [S.l.: s.n.], 2006. p.61–73.

DIAO, Y.; KELLER, A.; PAREKH, S. S.; MARINOV, V. V. Predicting Labor Cost through IT Management Complexity Metrics. In: INTEGRATED NETWORK MANAGEMENT, 2007. **Anais...** [S.l.: s.n.], 2007. p.274–283.

DOS SANTOS, C. R. P.; BEZERRA, R. S.; CERON, J. a. M.; GRANVILLE, L. Z.; TAROUCO, L. M. R. On using mashups for composing network management applications. **Comm. Mag.**, Piscataway, NJ, USA, v.48, p.112–122, December 2010.

DRAKE, J. **HP ITSM Reference Model**. 2000.

EDBERG, D.; IVANOVA, P.; JR., W. L. K. Methodology Mashups: an exploration of processes used to maintain software. **J. of Management Information Systems**, [S.l.], v.28, n.4, p.271–304, 2012.

ENNALS, R. J.; GAROFALAKIS, M. N. MashMaker: mashups for the masses. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2007., 2007, New York, NY, USA. **Proceedings...** ACM, 2007. p.1116–1118. (SIGMOD '07).

FENSEL, D.; HENDLER, J. A.; LIEBERMAN, H.; WAHLSTER, W. (Ed.). **Spinning the Semantic Web**. new.ed. Cambridge, MA: MIT Press, 2005.

FERNÁNDEZ, J.; MÁRQUEZ, A. **Maintenance Management in Network Utilities**: framework and practical implementation. [S.l.]: Springer, 2012. (Springer Series in Reliability Engineering).

FUNG, B.; TROJER, T.; HUNG, P. C. K.; XIONG, L.; AL-HUSSAENI, K.; DSSOULI, R. Service-Oriented Architecture for High-Dimensional Private Data Mashup. **Services Computing, IEEE Transactions on**, [S.l.], v.5, n.3, p.373–386, 2012.

GEBHARDT, H.; GAEDKE, M.; DANIEL, F.; SOI, S.; CASATI, F.; IGLESIAS, C.; WILSON, S. From Mashups to Telco Mashups: a survey. **Internet Computing, IEEE**, [S.l.], v.16, n.3, p.70–76, 2012.

GRAY, J. Why Do Computers Stop and What Can Be Done About It? In: SYMPOSIUM ON RELIABILITY IN DISTRIBUTED SOFTWARE AND DATABASE SYSTEMS, 1986. **Anais...** [S.l.: s.n.], 1986. p.3–12.

GRAY, W. D.; JOHN, B. E.; ATWOOD, M. E. Project Ernestine: validating a goms analysis for predicting and explaining real-world task performance. In: HUMAN-COMPUTER INTERACTION, 1993. **Anais...** [S.l.: s.n.], 1993.

GRIZZARD, J. B.; SHARMA, V.; NUNNERY, C.; KANG, B. B.; DAGON, D. Peer-to-peer botnets: overview and case study. In: FIRST WORKSHOP ON HOT TOPICS IN UNDERSTANDING BOTNETS, 2007, Berkeley, CA, USA. **Proceedings...** USENIX Association, 2007. p.1–1. (HotBots'07).

HARRELL, F. **Regression modeling strategies** : with applications to linear models, logistic regression, and survival analysis. Corrected.ed. [S.l.]: Springer, 2001.

HARRIS, M. D. S.; HERRON, D.; IWANICKI, S. **The Business Value of IT**: managing risks, optimizing performance and measuring results. 1st.ed. Boston, MA, USA: Auerbach Publications, 2008.

HOYER, V.; STANOESVKA-SLABEVA, K.; JANNER, T.; SCHROTH, C. Enterprise Mashups: design principles towards the long tail of user needs. In: SERVICES COMPUTING, 2008. SCC '08. IEEE INTERNATIONAL CONFERENCE ON, 2008. **Anais...** [S.l.: s.n.], 2008. v.2, p.601 –602.

HOYER, V.; STANOEVSKA-SLABEVA, K. The Changing Role of IT Departments in Enterprise Mashup Environments. In: SERVICE-ORIENTED COMPUTING — ICSOC 2008 WORKSHOPS, 2009, Berlin, Heidelberg. **Anais...** Springer-Verlag, 2009. p.148–154.

HP. **HP Service Management Reference Model**. 2008.

Human error: models and management. **BMJ**, [S.l.], v.320, n.7237, p.768–770, March 2000.

IBM. **IBM systems management solutions for System x**. 2008.

ISACA. **Control Objectives for Information and related Technologies (COBIT)**. 2008.

ISO/IEC 20000-2. [S.l.: s.n.], 2005.

JHINGRAN, A. Enterprise information mashups: integrating information, simply. In: VLDB '06: PROCEEDINGS OF THE 32ND INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 2006. **Anais...** VLDB Endowment, 2006. p.3–4.

JOHANNSEN, F.; LEIST, S.; ZELLNER, G. Six sigma as a business process management method in services: analysis of the key application problems. **Inf. Syst. E-bus. Manag.**, [S.l.], v.9, n.3, p.307–332, Sept. 2011.

JURIC, M. B. **Business Process Execution Language for Web Services BPEL and BPEL4WS 2nd Edition**. [S.l.]: Packt Publishing, 2006.

KAMINSKY, D.; MILLER, B.; SALAHSHOUR, A.; WHITMORE, J. Policy-Based Automation in the Autonomic Data Center. In: INTERNATIONAL CONFERENCE ON AUTONOMIC COMPUTING, 2008., 2008. **Proceedings...** IEEE Computer Society, 2008. p.209–210. (ICAC '08).

KELLER, A.; HELLERSTEIN, J. L.; WOLF, J. L.; WU, K. L.; KRISHNAN, V. The CHAMPS system: change management with planning and scheduling. In: NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 2004. NOMS 2004. IEEE/IFIP, 2004. **Anais...** [S.l.: s.n.], 2004. v.1, p.395–408 Vol.1.

KHAN, F. I.; AMYOTTE, P. R.; MATTIA, D. G. D. HEPI: a new tool for human error probability calculation for offshore operation. **Safety Science**, [S.l.], v.44, n.4, p.313 – 334, 2006.

KIERAS, D. Using the Keystroke-Level Model to Estimate Execution Times. **University of Michigan**, [S.l.], 2001.

KIRWAN, B. The validation of three human reliability quantification techniques THERP, HEART and JHEDI: part 1 -ttechnique descriptions and validation issues. **Applied Ergonomics**, [S.l.], v.27, n.6, p.359 – 373, 1996.

KIRWAN, B. The validation of three human reliability quantification techniques THERP, HEART and JHEDI: part 3 - practical aspects of the usage of the techniques. **Applied Ergonomics**, [S.l.], v.28, n.1, p.27 – 39, 1997.

KIRWAN, B.; KENNEDY, R.; TAYLOR-ADAMS, S.; LAMBERT, B. The validation of three Human Reliability Quantification techniques THERP, HEART and JHEDI: part 2 - results of validation exercise. **Applied Ergonomics**, [S.l.], v.28, n.1, p.17 – 25, 1997.

KOESTER, H. H.; LEVINE, S. P. Validation of a keystroke-level model for a text entry system used by people with disabilities. In: ACM CONFERENCE ON ASSISTIVE TECHNOLOGIES, 1994, New York, NY, USA. **Proceedings...** ACM, 1994. p.115–122. (Assets '94).

KOSCHMIDER, A.; HOYER, V.; GIESSMANN, A. Quality metrics for mashups. In: ANNUAL RESEARCH CONFERENCE OF THE SOUTH AFRICAN INSTITUTE OF COMPUTER SCIENTISTS AND INFORMATION TECHNOLOGISTS, 2010., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p.376–380. (SAICSIT '10).

KRAFCIK, J. Triumph of the Lean Production System. **Sloan Management Review**, [S.l.], v.30, n.1, p.44–52, 1988.

LACY, S.; MACFARLANE, I. **ITIL Service Transition Version 3.0**. London, UK: The Stationery Office, 2007.

LE-PHUOC, D.; POLLERES, A.; HAUSWIRTH, M.; TUMMARELLO, G.; MORBIDONI, C. Rapid prototyping of semantic mash-ups through semantic web pipes. In: WORLD WIDE WEB, 18., 2009, New York, NY, USA. **Proceedings...** ACM, 2009. p.581–590. (WWW '09).

LEE, S.; WONG, T.; KIM, H. S. To automate or not to automate: on the complexity of network configuration. In: PROC. IEEE ICC, 2008. **Anais...** [S.l.: s.n.], 2008.

LIKER, J. **The Toyota way**: 14 management principles from the world's greatest manufacturer. [S.l.]: McGraw-Hill, 2004.

LIU, X.; HUANG, G.; MEI, H. Towards End User Service Composition. In: COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, 2007. COMPSAC 2007. 31ST ANNUAL INTERNATIONAL, 2007. **Anais...** [S.l.: s.n.], 2007. v.1, p.676 –678.

LIU, X.; HUANG, G.; MEI, H. A User-Oriented Approach to Automated Service Composition. In: WEB SERVICES, 2008. ICWS '08. IEEE INTERNATIONAL CONFERENCE ON, 2008. **Anais...** [S.l.: s.n.], 2008. p.773 –776.

LIU, X.; HUI, Y.; SUN, W.; LIANG, H. Towards Service Composition Based on Mashup. In: SERVICES, 2007 IEEE CONGRESS ON, 2007. **Anais...** [S.l.: s.n.], 2007. p.332 – 339.

MAGUIRE, R. **Safety Cases and Safety Reports**: meaning, motivation and management. [S.l.]: Ashgate, 2006.

MAJCHRZAK, A.; MORE, P. H. B. Emergency! Web 2.0 to the rescue! **Commun. ACM**, New York, NY, USA, v.54, n.4, p.125–132, Apr. 2011.

MATTHEWS, G.; DAVIES; WESTERMAN; STAMMERS. **Human performance**: cognition, stress, and individual differences. [S.l.]: Psychology Press, 2000.

MCCLELLAN, H. **Applying Lean Enterprise Principles to Optimize Delivery of Customer Service**. [S.l.]: Massachusetts Institute of Technology, Sloan School of Management, 2008.

MERRIL, D. **Mashups**: the new breed of web app - an introduction to mashups. http://www.ibm.com/developerworks/web/library/x-mashups.html.

MILLER, G. A. The Magical Number Seven, Plus or Minus Two: some limits on our capacity for processing information. **The Psychological Review**, [S.l.], v.63, n.2, p.81–97, March 1956.

MILLIKEN, K. R.; CRUISE, A. V.; ENNIS, R. L.; FINKEL, A. J.; HELLERSTEIN, J. L.; LOEB, D. J.; KLEIN, D. A.; MASULLO, M. J.; VAN WOERKOM, H. M.; WAITE, N. B. YES/MVS and the automation of operations for large computer complexes. **IBM System Journal**, [S.l.], v.25, p.159–180, June 1986.

MOHAMMADI, S.; KHALILI, A.; ASHOORI, S. Using an Enterprise Mashup Infrastructure for Just-in-Time Management of Situational Projects. In: BUSINESS ENGINEERING, 2009. ICEBE '09. IEEE INTERNATIONAL CONFERENCE ON, 2009. **Anais...** [S.l.: s.n.], 2009. p.3 –10.

MULYAR, N.; AALST, W. M. P. van der. Patterns in Colored Petri Nets. **BETA Working Paper Series, WP**, [S.l.], v.139, 2008.

NESTLER, T. Towards a mashup-driven end-user programming of SOA-based applications. In: INTERNATIONAL CONFERENCE ON INFORMATION INTEGRATION AND WEB-BASED APPLICATIONS & SERVICES, 10., 2008, New York, NY, USA. **Proceedings...** ACM, 2008. p.551–554. (iiWAS '08).

OGC. **Information Technology Infrastructure Library v3 (ITIL v3)**. 2008.

OGRINZ, M. **Mashup Patterns**: designs and examples for the modern enterprise. 1.ed. [S.l.]: Addison-Wesley Professional, 2009.

OLSON, J. R.; OLSON, G. M. The growth of cognitive modeling in human-computer interaction since GOMS. **Hum.-Comput. Interact.**, Hillsdale, NJ, USA, v.5, n.2, p.221–265, June 1990.

OMG. **Business Process Model And Notation (BPMN) Version 2.0**. 2011.

OPPENHEIMER, D.; GANAPATHI, A.; PATTERSON, D. A. Why do internet services fail, and what can be done about it? In: USENIX SYMPOSIUM ON INTERNET TECHNOLOGIES AND SYSTEMS - VOLUME 4, 4., 2003, Berkeley, CA, USA. **Proceedings...** USENIX Association, 2003. p.1–1. (USITS'03).

O'REILLY, T. **What is Web 2.0**. 2005.

PAN, J.; PAUL, S.; JAIN, R. A survey of the research on future internet architectures. **IEEE Communications Magazine**, [S.l.], v.49, n.7, p.26–36, 2011.

PANDE, P.; NEUMAN, R.; CAVANAGH, R. **The Six Sigma Way**: how ge, motorola, and other top companies are honing their performance. [S.l.]: McGraw-Hill, 2000.

PULTORAK, D.; HENRY, C.; LEENARDS, P. **MOF 4. 0**: a pocket guide: (microsoft operations framework). [S.l.]: Van Haren Publishing, 2008. (Van Haren Series).

PYZDEK, T. **The Six Sigma handbook**: a complete guide for green belts, black belts, and managers at all levels. [S.l.]: McGraw-Hill, 2003.

RASMUSSEN, J. Skills, rules, and knowledge: signals, signs, and symbols, and other distinctions in human performance models. **IEEE Transactions on Systems, Man and Cybernetics**, [S.l.], v.13, n.3, p.257–266, 1983.

REASON, J. **Human Error**. [S.l.]: Cambridge [England] ; New York : Cambridge University Press, 1990. xv, 302 p., 1990.

RUSSELL, N.; ARTHUR; AALST, W. M. P. van der; MULYAR, N. **Workflow Control-Flow Patterns**: a revised view. [S.l.]: BPMcenter.org, 2006.

SANTOS, C. dos; BEZERRA, R.; GRANVILLE, L.; BERTHOLDO, L.; CHENG, W.; ANEROUSIS, N. A data confidentiality architecture for developing management mashups. In: INTEGRATED NETWORK MANAGEMENT (IM), 2011 IFIP/IEEE INTERNATIONAL SYMPOSIUM ON, 2011. **Anais...** [S.l.: s.n.], 2011. p.49 –56.

SANTOS, C. dos; BEZERRA, R.; GRANVILLE, W.; LOEWENSTERN, D.; SHWARTZ, L.; ANEROUSIS, N. Quality Improvement and Quantitative Modeling âĂŞ-Using Mashups for Human Error Prevention. In: INTEGRATED NETWORK MANAGEMENT (IM), 2011 IFIP/IEEE INTERNATIONAL SYMPOSIUM ON, 2013. **Anais...** [S.l.: s.n.], 2013.

SANTOS, C. R. P. dos; BEZERRA, R. S.; CERON, J. M.; GRANVILLE, L. Z.; TAROUCO, L. M. R. Identifying botnet communications using a mashup-based approach. In: LANOMS, 2011. **Anais...** [S.l.: s.n.], 2011. p.1–6.

SCHOBESBERGER, D.; CARTWRIGHT, W. The Potential of Using Web Mapping as a Tool to Support Cultural History Investigations. In: KRIZ, K.; CARTWRIGHT, W.; KINBERGER, M. (Ed.). **Understanding Different Geographies**. [S.l.]: Springer Berlin Heidelberg, 2013. p.175–192. (Lecture Notes in Geoinformation and Cartography).

SHEN, B. Support IT service management with process modeling and analysis. In: SOFTWARE PROCESS, 2008 INTERNATIONAL CONFERENCE ON MAKING GLOBALLY DISTRIBUTED SOFTWARE DEVELOPMENT A SUCCESS STORY, 2008. **Proceedings...** Springer-Verlag, 2008. p.246–256. (ICSP'08).

SHWARTZ, L.; DIAO, Y.; GRABARNIK, G. Multi-tenant solution for IT service management: a quantitative study of benefits. In: INTEGRATED NETWORK MANAGEMENT, 2009. **Anais...** [S.l.: s.n.], 2009. p.721–731.

SIMMEN, D. E.; REISS, F.; LI, Y.; THALAMATI, S. Enabling enterprise mashups over unstructured text feeds with InfoSphere MashupHub and SystemT. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2009., 2009, New York, NY, USA. **Proceedings...** ACM, 2009. p.1123–1126. (SIGMOD '09).

SIRIN, E.; HENDLER, J.; PARSIA, B. Semi-automatic Composition of Web Services using Semantic Descriptions. In: IN WEB SERVICES: MODELING, ARCHITECTURE AND INFRASTRUCTURE WORKSHOP IN ICEIS 2003, 2002. **Anais...** [S.l.: s.n.], 2002. p.17–24.

SWAIN, A.; GUTTERMAN, H. **Handbook of human reliability analysis with emphasis on nuclear power plant applications**. Albuquerque, New Mexico: Sandia Laboratories, 1980. (NUREG/CR-1278).

THOM, L.; REICHERT, M.; IOCHPE, C. Activity Patterns in Process-aware Information Systems: basic concepts and empirical evidence. **International Journal of Business Process Integration and Management (IJBPIM)**, [S.l.], v.4, n.2, p.93–110, 2009.

TRIMBLE, J.; DAYTON, T.; HORN, B. Introduction to Shareable Mashups, Widgets, Composites, and Object-Oriented GUIs–Task Effectiveness and User Composing in Collaborative Software Minitrack. In: HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 2012., 2012, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2012. p.668–. (HICSS '12).

VERMA, A.; DESAI, N.; BHAMIDIPATY, A.; JAIN, A.; NALLACHERRY, J.; ROY, S.; BARNES, S. Automated Optimal Dispatching of Service Requests. In: SRII GLOBAL CONFERENCE (SRII), 2011 ANNUAL, 2011. **Anais...** [S.l.: s.n.], 2011. p.426–429.

WHITTINGHAM, R. **The Blame Machine**: why human error causes accidents. [S.l.]: Taylor & Francis, 2012.

WILLIAMS, J. HEART - A Proposed Method for Achieving High Reliability. In: SYMPOSIUM ON THE ACHIEVEMENT OF RELIABILITY IN OPERATING PLANT, SAFETY AND RELIABILITY SOCIETY, 1985. **Anais...** [S.l.: s.n.], 1985. p.87–109.

WILLIAMS, J. Validation of human reliability assessment techniques. **Reliability Engineering**, [S.l.], v.11, n.3, p.149 – 162, 1985.

WILSON, S.; DANIEL, F.; JUGEL, U.; SOI, S. Orchestrated user interface mashups using w3c widgets. In: CURRENT TRENDS IN WEB ENGINEERING, 11., 2012, Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2012. p.49–61. (ICWE'11).

YEMINI, S.; KLIGER, S.; MOZES, E.; YEMINI, Y.; OHSIE, D. High speed and robust event correlation. **Communications Magazine, IEEE**, [S.l.], v.34, n.5, p.82–90, may 1996.

YU, J.; BENATALLAH, B.; CASATI, F.; DANIEL, F. Understanding Mashup Development. **IEEE Internet Computing**, Piscataway, NJ, USA, v.12, n.5, p.44–52, 2008.

ZOU, J.; PAVLOVSKI, C. Towards Accountable Enterprise Mashup Services. In: BUSINESS ENGINEERING, 2007. ICEBE 2007. IEEE INTERNATIONAL CONFERENCE ON, 2007. **Anais...** [S.l.: s.n.], 2007. p.205 –212.

APPENDIX A - CONCLUSÕES

A investigação conduzida durante esta tese demonstrou a aplicação da tecnologia de mashups na área de Gerência de Serviços de Tecnologia da Informação (Information Technology Service Management - ITSM). Especificamente, é feita uma análise para validar a hipótese de que **"o uso de mashups aprimora o desempenho de processos de gerência de TI centrados em humanos"**. Diferente da maioria dos trabalhos existentes na literatura, onde uma significativa parte foca no aspecto de qualidade (*i.e.,* arcabouços, processos e métricas que avaliam a eficácia do ponto de vista do recebidor de tais serviços), esta tese investiga a perspectiva do provedor de serviços. Em específico, aspectos de desempenho que possuem uma implicação direta na eficiência e custo da operação, do ponto de vista do provedor. Apesar de diferentes métricas poderem ser usadas para avaliar o desempenho, esta tese foca em produtividade e em confiabilidade, duas das métricas mais usadas por organizações provedoras de serviços de TI para avaliar o desempenho dos processos de gerência.

A análise foi realizada com base nos princípios do Six Sigma, que tem como base a compreensão das causas da variabilidade no processo e em servir como um guia para alcançar, progressivamente, níveis mais elevados de consistência. Inicialmente, foi realizada uma análise das causas das ineficiências e dos erros humanos nos processos de gerência de serviços. Através desta investigação, foi possível observar as ineficiências e erros mais comuns em ITSM. Em um segundo momento, a tecnologia mashups foi investigada para identificar métodos viáveis para resolver esses problemas. O conceito de padrões de mashup e módulos de prevenção de erros - com base na prevenção e interceptação de erros - representam abordagens eficazes para obter uma melhor orquestração do processo através de reprojeto e automação. É importante ressaltar que as soluções baseadas em mashup não pretendem substituir os modelos tradicionais de gerenciamento ou ferramentas existentes, em vez disso, os mashups têm como objetivo permitir que as tecnologias tradicionais possam ser facilmente usadas e integradas para criar novas aplicações. Finalmente, modelos quantitativos disponíveis na literatura são combinados para quantificar e prever o impacto de implantação da solução de mashups sobre os processos de ITSM .

Através de um estudo de caso real, relacionado com o processo de cumprimento de requisição, foi possível examinar os métodos introduzidos para resolver problemas de desempenho que podem surgir durante a execução de processos centrados em humano. O foco deste estudo de caso é na atividade centrada em operadores humanos com conhecimento dos procedimentos de cumprimento de requisicões e responsabilidades que incluem: monitoramento de novos pedidos, envio dos pedidos para os administradores de sistemas apropriados e verificação do cumprimento dos acordos de nível de serviço. Tradicionalmente, a coleta de informações no cumprimento de requisições tem exigido

o uso de várias ferramentas com interfaces de usuário distintas. A integração dos dados coletados tem sido um processo complexo que requer os scripts sofisticados, a manipulação de grandes quantidades de dados e um alto esforço para fornecer uma apresentação adequada dos resultados esperados. O sistema de pedidos de serviço baseado em mashups que foi desenvolvido reduziu o número de níveis de interação do usuário para apenas duas interfaces que não exigem conhecimentos técnicos avançados para serem manipuladas, além de esconder detalhes técnicos relacionados à integração dos serviços para o operador humano. As melhorias observadas na produtividade e confiabilidade dos processos de ITSM fornecidas pela aplicação de mashups demonstram a viabilidade da tecnologia como uma alternativa adequada para melhorar o gerenciamento de serviços de TI.

Resumindo, esta tese apresentou um programa de melhoria de desempenho no processo de cumprimento de requisição. Inicialmente, foram definidas métricas para avaliar o desempenho no ambiente em estudo (*i.e.*, produtividade e confiabilidade). Então, os modelos quantitativos foram combinados para observar e quantificar as questões mais relevantes no ambiente atual. Finalmente, um conjunto de métodos adequados (*i.e.*, os padrões de mashup e os módulos de prevenção de erro), foram identificados e implementados para resolver os principais problemas de desempenho.

APPENDIX B - SCIENTIFIC PRODUCTION

A.1 Patent

1. CARLOS RANIERY PAULA DOS SANTOS, Lisandro Zambenedetti Granville, Nikolaos Anerousis, David Matthew Loewenstern, Louis Jonh Percello, Winnie Cheng, Larisa Shwartz. Performance Management and Quantitative Modeling of IT Service Processes Using Mashup Patterns. US Patent Pending.

    - The patent application is related to methods and arrangements for quantitatively modeling service processes. More specifically the patent covers the idea of estimating quantitative metrics with respect to ITSM processes and determining mashup patterns to the process.

A.2 Published Papers

1. Rafael Santos Bezerra, CARLOS RANIERY PAULA DOS SANTOS, Lisandro Zambenedetti Granville, Liane Tarouco. On the Feasibility of Web 2.0 Technologies for Network Management: A Mashup-Based Approach. **IEEE/IFIP Network Operations and Management Symposium (NOMS), 19-23 April 2010, Osaka, Japan, ISBN: 978-1-4244-5366-5, pp. 487-494**.

    - *Status.* Approved and published.
    - *Description.* This paper investigated whether the characteristics usually mentioned by mashup advocators hold when mashups are employed for network management.

2. CARLOS RANIERY PAULA DOS SANTOS, Rafael Santos Bezerra, João Ceron, Lisandro Zambenedetti Granville, Liane Margarida Rockenbach Tarouco. Botnet Master Detection Using a Mashup-based Approach. **Short papers of the 6th International Conference on Network and Service Management (CNSM), 25-29 October 2010, Niagara Falls, Canada, ISBN: 978-1-4244-8908-4, pp. 390-393**.

    - *Status.* Approved and published.
    - *Description.* This paper analysed the usage of mashups as an approach for integrating and correlating information from web available binary analysis tools aiming to botmaster detection.

3. CARLOS RANIERY PAULA DOS SANTOS, Rafael Santos Bezerra, João Marcelo Ceron, Lisandro Zambenedetti Granville, Liane Margarida Rockenbach Tarouco. On Using Mashups for Composing Network Management Applications. **IEEE Communications Magazine, Vol. 48, Issue 12, December 2010, ISSN: 0163-6804, pp. 112-122**.

    - *Status.* Approved and published.
    - *Description.* This paper proposed an architecture and a system prototype that allows network administrators to design their own management applications through the composition of external resources.

4. CARLOS RANIERY PAULA DOS SANTOS, Rafael Santos Bezerra, Leandro Bertholdo, Lisandro Zambenedetti Granville, Winnie Cheng, Nikos Anerousis. A Data Confidentiality Architecture for Developing Management Mashups. **12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011), 23-27 May 2011, Dublin, Ireland, ISBN: 978-1-4244-9220-6, pp. 49-56**.

    - *Status.* Approved and published.
    - *Description.* This paper proposed a novel development methodology and system architecture, called Maestro, that allows developers to express their data privacy concerns and enforce policies during mashup executions.

5. CARLOS RANIERY PAULA DOS SANTOS, Rafael Santos Bezerra, João Marcelo Ceron, Lisandro Zambenedetti Granville, Liane Margarida Rockenbach Tarouco. Botnet Master Detection Using a Mashup-based Approach. **7th Latin American Network Operations and Management Symposium (LANOMS), 10-11 October 2011, Quito, Ecuador**.

    - *Status.* Approved and published.
    - *Description.* This paper proposed a flexible solution for creating new applications focused in detecting dynamic botnets.

6. CARLOS RANIERY PAULA DOS SANTOS, Lisandro Zambenedetti Granville, Winnie Cheng, David Loewenstern, Larisa Shwartz, Nikos Anerousis. Performance Management and Quantitative Modeling of IT Service Processes Using Mashup Patterns. **7th International Conference on Network and Services Management (CNSM 2011), 24-28 October 2011, Paris, France, ISBN: 978-1-4577-1588-4, pp. 1-9**.

    - *Status.* Approved and published.
    - *Description.* This paper provided a systematic framework for analyzing inefficiencies through a combined model to guide the use and estimate the value of improving orchestration of the process using mashup design patterns

7. CARLOS RANIERY PAULA DOS SANTOS, Lisandro Zambenedetti Granville, Larisa Shwartz, Nikos Anerousis, David Loewenstern. Quality Improvement and Quantitative Modeling – Using Mashups for Human Error Prevention. **13th IFIP/IEEE Symposium on Integrated Network and Service Management (IM 2013), 27-31 May 2013, Ghent, Belgium**.

- *Status.* Approved and published.
- *Description.* This paper analyzed the usage of mashups as an effective approach to cope with errors introduced by human operators while performing their daily activities in the context of IT Service Management (ITSM).

On the Feasibility of Web 2.0 Technologies for Network Management: A Mashup-Based Approach

Rafael Santos Bezerra,
Carlos Raniery Paula dos Santos,
Leandro Marcio Bertholdo,
Lisandro Zambenedetti Granville,
Liane Rockenbach Tarouco
Institute of Informatics
Federal University of Rio Grande do Sul, Brazil
Email: {rsbezerra, crpsantos, granville}@inf.ufrgs.br, {berthold,liane}@penta.ufrgs.br

***Abstract*—Mashups are a new breed of Web applications, created through the integration of external resources available on the Web. Recently, they have been considered a hallmark of Web 2.0 technologies, placing the end user on a developer role and encouraging both collaboration and reuse. Following the increasing efforts in investigating new approaches to network management, mashups present themselves as a technology that can bring several advantages to the field. However, to this date, the usage of mashups in network management remains unexplored. Therefore, the present paper approaches this subject, proposing a Mashup Development Tool to network management. We discuss both the architecture of such system and a proof of concept prototype. We them employ our prototype to address the case study of integrating Autonomous System routing information.**

I. Introduction

The ever increasing complexity of computer networks constantly demands network management tools with more sophisticated capabilities. Usually, however, such capabilities do not come along with adequate user interfaces presented to the human network operators. Despite the current great availability of visual programming libraries and interactive solutions on the Web, many network management sytems (*e.g.*, Nagios, OpenNMS, OpManager) persist, requiring skilled human operators to run them. Besides, such solutions cannot address, for example, situations where managers need to visualize information in a specific and meaningful way, or when such information needs to be aggregated with external data from heterogeneous sources. This scenario is more problematic in the management of large backbones, because the management information usually comes from many different systems and devices.

Motivated by the aforementioned issues, there is a recent research trend towards the use, in the network management field, of technologies originally established and successfully employed in other areas. This trend includes, for example, the investigation of Web services [1] and *peer-to-peer* (P2P) [2] for network management: Web services traffic can cross administrative boundaries easier than SNMP because they use Internet protocols (*e.g.*, HTTP, FTP, and SMTP) as the underlying transport mechanism, while P2P potentially enables human-centric collaborative management along different administrative domains. Among the new technologies available, a set of novel ones, referenced with the title of Web 2.0, has been catching the attention of both industry and academy. Web 2.0 designates a new kind of Web applications where users are motivated to actively create and organize contents available on the Web [3].

In the myriad of technologies and applications that define the Web 2.0, one is of special interest in this paper: the *mashups*. Mashups are Web applications created from the composition and reuse of pre-existing external resources. Current work and approaches on mashups propose Web-based systems that allow end users with no programming expertise to develop their own applications. This development happens through the dynamic integration of existing resources like Web services, Web pages, and RSS feeds. This integration, which is inherently complex, is usually enabled by high-level abstractions and user-friendly interfaces [4]. As far as the authors of this paper are aware of, no effort towards the investigation of mashups for network management has been carried out so far, despite the existence of both academic [5] and industry [6] efforts on general purpose mashup development systems.

Considering the potential advantages of mashups and the lack of research about Web 2.0 technologies applied to network management, we aim in this paper at investigating whether the characteristics usually mentioned by mashup advocators hold when mashups are employed for network management. In particular, the following questions will be investigated:

- Can network management information be composed dynamically with external data from heterogeneous sources?
- How those compositions are better than traditional approaches used by network managers, such as script programming?
- Can mashups address network management situational needs?

In order to answer these questions, we have designed and implemented an architecture of a mashup system that enables the composition of management information sources available on the Web. As case study, we created, using the developed

system, a mashup to aggregate and monitor routing information collected from Autonomous Systems (ASes) that operate on the Internet using the Border Gateway Protocol (BGP). The aggregated information includes the amount of traffic exchanged between ASes and the number of routes announced by one AS to another, in a scenario known as BGP peering. Monitoring BGP peering is important because ASes rely on it to better maintain the health of Internet connectivity. The analysis of the aggregated information can lead to evidences of several critical problems. For example, it is possible for an AS to detect if a Service Level Agreement (SLA) is being broken by a neighbor Internet Service Provide (ISP).

The remainder of this paper is organized as follows. In Section 2 we review the main background and related work on the mashup technology. In Section 3 we introduce our proposed architecture that aims to support the development of mashups for network management. Also in Section 3, we describe the developed mashup sytem prototype. In Section 4 we discuss the case study. Finally, we draw conclusions and propose future work in Section 5.

II. Related Work

Mashups are Web applications created from the composition of preexisting Web resources like dynamic Web pages, Web services, and RSS feeds [5]. One of the main distinctions between mashups and other traditional composition technologies (*e.g.*, BPEL) is the goal of enabling average Web users with no programming skills to create their own applications, which better address their needs. Traditional composition technologies, on the other hand, require reasonably skilled developers that have sufficient knowledge about, for example, programming languages and paradigms, communication protocols, and distributed systems.

End user created mashups are built using mashup development systems, which themselves tend to be Web applications too. Usually, mashup development systems accumulate the roles of mashup repositories and runtime environments. This leads to the mixed scenario where a single system is used to create, store, and execute mashups. That does not mean that a mashup created in one system cannot be moved to be executed in another external system, or that it cannot be stored in a third system to form a remote library of mashups. Although this is all possible, in most of the current cases mashups are created, stored, and executed in the same environment. The term mashup system is usually employed to refer to such an environment. Mahsup systems are normally implemented using traditional Web technologies such as PHP and Java Server pages (JSP), and present sophisticated graphical Web interfaces created with usability-oriented technologies such as Asynchronous Javascript and XML (AJAX) [7] and Macromedia Flash.

The general components that compose the typical architecture of a mashup system are presented in Figure 1. Two different users interact with the mashup system: developers and end users. Developers define the compositions that result in the final mashups. End users, in their turn, are those interested in using the mashups created by developers. Once accessed by end users, developed mashups are executed to materialize the compositions previously specified. In this process, the mashup system retrieves information from remotely located Web resources (using technologies like SOAP, RPC, and Syndication protocols), performs operations (*e.g.*, sorting, filtering, aggregating) on the retrieved data, and finally builds a Web page that presents the composition results.

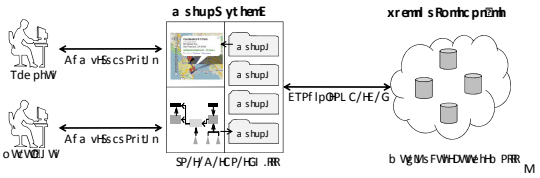

Fig. 1. Mashup system architecture

Although Figure 1 separates mashup developers and end users, both roles can be played by the same single person. In fact, the possibility of unskilled end users acting as developers is one of the key advantages advertised by mashup advocators. Whether such advantage really holds depends, among other factors, on the target problem the end user aims at solving and the mashup system used. Having that said, there are currently a set of relevant efforts on both academia and industry on researching and deploying mashup-based solutions.

One of the first investigations on mashups has been carried out by Banerjee *et al.* [8]. The authors have discussed the use of mashups in the development of telecommunication applications. They have proposed a multi-layered architecture that employs Web services as components, but with focus on easiness for inexperienced end users. The proposed architecture has been used, however, in the development of a single, very specific mashup (*i.e.*, call-a-cab) that does not allow, for example, an end user to build his/her own mashups. In addition, authors have not shown examples on how their architecture could be used in other telecommunication scenarios.

Yu *et al.* [9] have presented an overview on mashup development and the differences between mashups and traditional composition approaches. The authors have proposed a framework to classify different mashups systems in terms of what is provided by such systems, and how that is done. They also have identified some open research topics on mashups; for example, the authors observe that many of today's mashups still have a very limited audience, which may lead to possible scalability problems in future, more spread intensive usage. They also conclude that the improvement of mashup systems and the proliferation of components or modules for mashups are essential to help end users to build their own applications.

In the industry, one of the most relevant movements is related to building mashup development systems. Some examples are JackBe Presto[1], Yahoo! Pipes[2], IBM Mashup

[1]http://www.jackbe.com/presto
[2]http://pipes.yahoo.com

Center[3]. Pipes is almost exclusively related of integrating Web information through mashups. However, the other aforementioned examples are a different kind of effort, called enterprise mashups [10]. This proposal holds some similarities with SOA, proposing to develop an infrastructure of components and a system to allow dynamic and facilitated integration inside the context of an organization. Such efforts indicate that the industry believes in the potential advantages of the mashup technology.

The current industry initiatives and research efforts, as presented above, look at mashups usually considering specific scenarios. We believe that mashups can also be employed in the specific field of network management, with the potential advantage of allowing network human operators to create themselves their own, highly customized management applications without having to be trained, for example, in a complex composition technology. In the next section we introduce our mashup proposal in order to investigate this issue. We describe an architecture for a tool that aims to help network administrators to build their management mashups.

III. Proposal

To better understand the behavior of mashups when applied in network management, we defined an architecture for a mashup-based network management system. Based on this architecture, we developed a prototype tool that enabled us to create mashups and thus understand if they present advantages for the network management. It is important to note that, while this architecture describes a tool for network management, it is generic enough to describe even general purpose mashup systems.

A. Tiers

Figure 2 presents the architecture proposal and how its elements are related to each other. As aforementioned, we have two conceptual user roles, that can be, and ideally are, performed by the same user. The role of the developer relates to network operators interested in creating mashups for a specific network management need, and an administrator is one only interested in using the mashups already created by developers. With the proposed architecture, network operators acting as developers will be able to use their own knowledge to build applications more suitable to their needs. Besides, the architecture enables operators from different networks to collaborate with one another in creating more sophisticated management mashups. Another characteristic of the proposed architecture is that it allows mashup sharing between mashup systems, *i.e.*, a mashup created for a specific system can be accessed by other systems.

We designed the proposed architecture as a Web application based on the three-tier client-server model (*i.e.,* logic, presentation, data) [11]. By doing that, we could better separate the logical elements and individually handle them. For example, a change of external resources would only affect the implementation of the adaptation tier. These three tiers are:

[3]http://www-01.ibm.com/software/info/mashup-center/

- ***Adaptation Tier***: responsible for accessing and adapting external resources into a common format through elements called wrappers;
- ***Composition Tier***: orchestrates the execution of mashups, and performs operations (*e.g.*, sorting, filtering, aggregating) over the retrieved information;
- ***Presentation Tier***: where both end users, and network managers can create and/or use mashups, through the Development Environment (DE) and the Runtime Environment (RE).

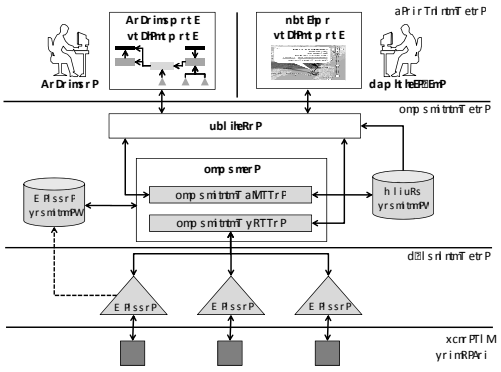

Fig. 2. Proposed architecture

B. Elements and Interactions

As mentioned before, wrappers are elements responsible for accessing external information, and making it available to the mashup system. In fact, they act as gateways to the different access methods (*e.g.*, SOAP, RPC) and data formats (*e.g.,* XML, CSV). For each wrapper, a set of meta-information describing the accessed resource is defined and stored in the Wrapper Repository. Examples of meta-information are input and output parameters, and resource description. This meta-information has two main goals: to allow the external resource to be accessed, and to describe resources to developers. The mashup system reads this meta-information and, based on it, developers can create their compositions. When a mashup is executed, each relevant wrapper for that mashup starts, retrieves external data, and forwards it to elements of the composition tier, which will integrate the retrieved data and build a Web page presenting the composition result.

On the composition tier, the Composer element is responsible for managing the compositions created by developers. When a mashup is created, a module of the Composer, called Composition Planner, receives the integration defined by developers (in the form of external resources pointers and relationships between these resources), validates such integration, transforms it into a common representation that can be processed by the mashup system, and finally stores it in the Mashup Repository. Another module of the Composer is the Composition Runner, which orchestrates calls to the

elements when executing the compositions stored by the Planner. The Composition Runner interacts directly with wrappers, processing the information received from them and outputting the mashup presentation (*i.e.*, a Web page).

Another element of the composition tier is the Publisher, whose function is to guarantee the access control of mashups. When a developer creates a mashup, he/she defines groups and roles. For example, a developer can define that some mashup cannot be edited, but it could be extended or used by the a specific group of users. The Publisher is also responsible for allowing mashups to be moved to external systems to be remotely executed. In this case, there are two mashup systems involved: the requester and the provider. The requester queries the provider's Publisher for a list of available mashups. The provider, in its turn, accesses its internal repository and replies with the list of mashups enabled to be externally executed. When the requester system asks for some specific mashup, both wrappers and mashup definition are delivered back.

The presentation tier has elements that are executed on the client-side, in a Web browser: the aforementioned Development Environment (DE) and the Runtime Environment (RE). The DE is used by the developer to create mashups by selecting which information will be integrated, setting the input data, and establishing relationships between them. The RE module, in its turn, exhibits the final results of the compositions (*i.e.*, mashups). When a network manager accesses the RE, the Publisher is called. It interacts with the Composition Runner requesting the execution of the selected mashup, receives the output (*e.g.*, HTML plus Javascript page) and sends it to the Presentation Layer, which will display the output to the user.

C. Interaction Components

In order to allow even unskilled network operators to build their own management applications, high-level abstractions should be used. These abstractions hide technical details of the external resources and of the operations executed on the retrieved data. To achieve that, we propose a set of interaction components that are displayed to network operators when acting as developers. These components allow the operators to define the business logic behind a mashup. Four types of components are proposed, although the architecture is extensible enough to support new ones. These four components are:

- **Visual**: represents the way results can be displayed, for example, in a map, table, or tree;
- **Control**: represents basic programming logic, such as loops and conditions;
- **Operation**: are functionalities available in the mashup system that perform operations on the retrieved information. Examples include: filtering, merging, aggregating, arithmetic operations, and boolean logic operations;
- **Adaptation**: created based on the meta-information defined in wrappers, it represents the external resources accessed by the system.

To allow the reuse of existing mashups and build more sophisticated compositions, we also defined a component called **Reuse**. This component represents the compositions available in the mashup repository. Through the integration of this component with the aforementioned four, developers can extend a mashup created by some other developer.

D. Mashup System Prototype

Based on the presented architecture, we developed a network management mashup system prototype. This prototype focuses on enabling network managers to create their own management tools, even if they possess little to no programming experience. Our goals with this prototype are to evaluate the core concepts of our proposed architecture and to evaluate whether such architecture is useful in aiding a the development of a mashup system. Achieving these goals is one of the reasons we decided to develop a new mashup system instead of using an already existing tool. Furthermore, during our research, we could not find a mashup system suitable to the needs of network managers we perceive, such as compatibility with network management tools (*e.g.* MRTG) and protocols (*e.g.* SNMP), motivating us to build our own mashup system.

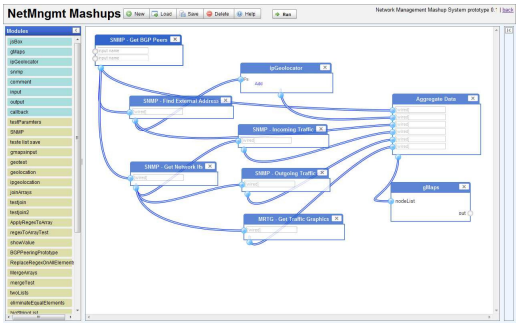

Fig. 3. Developer Environment

Our tool uses SOAP Web Services [12] as wrapper model, aiming to leverage the cross-plataform capabilities and the widespread standardization of this technology. As proposed by the architecture, these services are invoked by the composer, which is a server-side, JSP based engine. This engine comunicates with the presentation tier development environment, which is an AJAX based web system created with the WireIt library[4]. The development environment, as presented in Figure 3 presents a drag-and-drop interface for mashup development, where a user connect boxes with wires to represent the data flow. Each box represents a component, having a suitable number of inputs and, in this initial version, one single output. The mashups can also be executed in this environment. We also developed a set of components to this system, such as wrappers, operators and visual components, in order to create management tools. These components, as well as their use in the creation of a network mashup employing our prototype, are detailed in the next section.

[4]http://javascript.neyric.com/wireit/

On the next section, we also evaluate the proposed architecture and the prototype developed through the creation of a mashup whose specific goal is to integrate BGP routing information. The BGP peering issue addressed will be described and then we discuss the developed mashup.

IV. Case Study

Based on the architecture proposed in section 3, and using the developed prototype, we created a mashup that addresses BGP peering issues. Such issues are not easily tackled by administrators when using traditional network management tools. Our proposal is to employ Web 2.0 technologies in these problems to know if they present advantages over traditional management solutions. In this section we present the case study of managing BGP peering, discuss about the difficulties in doing so through conventional tools and present our mashup based solution.

A. BGP Peering

The Internet is an interconnection of independent administrative domains, most of them owned by Internet Service Providers (ISPs). Inside such domains, network administrators define policies (*e.g.,* for routing, traffic shaping, and traffic priorities) independently from other domains. For this reason, these domains are called Autonomous Systems (ASes). The connections between these ASes are maintained and controlled through the Border Gateway Protocol (BGP) [13]. When two ASes establish a connection, one announces routes to the other and vice-versa through BGP. These announcements (which are basically a set of IP address prefixes) indicate that one AS wants to receive traffic from the other. Two interconnected ASes are called *peers*, and the ASes interconnection is denominated *BGP peering* [14].

The main issues of BGP peering are related to unexpected behavior of the inter-AS relationship, for example, when a certain amount of expected traffic routed from/to a peer is suddenly replaced by a much higher or lower amount. This situation arises due to several reasons. For example, when a peer is not announcing its prefixes to the other peer, or when that peer is announcing an unconventional number of routes possibly breaking an SLA. This can happen when BGP routers are misconfigured or malfunctioning. Whichever the reason, the issue should be solved as soon as possible, given the large volume of traffic usually exchanged between ASes, which ends up leading to financial consequences too. In order to properly monitor the status of BGP peering, a network operator, located inside an AS, needs to have access, at least, to the following information:

- Local and remote BGP peers: a local BGP router is connected to remote BGP routers that belong to other ASes (*i.e.*, remote peers). Information about local and remote peers can include unique identifier (Autonomous System Number - ASN), ISP owner, and location of BGP routers;
- Number of announced routes: the number of routes announced by the local peer to remote ones, as well as the number of route announcements received from the remote peers to the local one. Usually, this number is defined by AS administrators when they establish SLAs between their ASes;
- Traffic exchanged: the amount of traffic exchanged between ASes. There is a financial cost associated to this exchange, which is also usually defined in SLAs. Network managers may define policies and strategies to split traffic between different peers in order to optimize costs.

The BGP-4 MIB [13] module, standardized by the IETF, provides both peer information and routing information via the Simple Network Management Protocol (SNMP). However, the associated analysis is not trivial. For example, in order to discover the IP address of remote ASes routers, the information retrieved from the local router's BGP-4 MIB module must be translated accessing complementary information from another MIB module: the MIB-II module. Also, to find the number of routes announced by each remote peer, the whole BGP-4 routing table must be retrieved from the local router via SNMP. That is so because this information (*i.e.*, number of routes announced per peer) is not supported in the BGP-4 MIB module [14] but can be calculated from the BGP-4 routing table. The task of discovering the amount of exchanged traffic is even more complex. This is done by checking which local network interfaces are connected to remote peers, and accessing the objects If.InOctects and If.OutOctects of the MIB-II module for each of these interfaces. This is insufficient for richer traffic analysis because these two objects do not provide per se, for example, temporal traffic averages, thus forcing the management station to pool the local BGP router to calculate the temporal traffic behavior in each interface. This is in fact the technique usually employed by traffic monitoring tools such as MRTG. However, the integration of such tools with information retrieved from the BGP-4 MIB module is difficult to be done. All this process is normally done manually by the network administrator, and at most through script programming.

B. Created Mashup

We developed the mashup to enable network administrators to manage the BGP peering issue previously presented. The final mashup aggregates routing information from local BGP routers and their remote peers on a map-based Web page. Map mashups are currently the most widely used mashup category on the Web [15] due to the increasing popularity of map APIs such as Google Maps[1], Yahoo Maps [2], and Microsoft's Bing Maps [3]. Network maps, on their turn, are classical tools used for the visualization of management topologies [16].

To build the mashup, the following components have been employed:

[1] http://code.google.com/apis/maps/
[2] http://maps.yahoo.com/
[3] http://www.microsoft.com/virtualearth/

- **SNMP**: an Adaptation component which allows the access to SNMP managed devices through a SNMP wrapper. Supports the protocol main operations (*e.g.*, GET, GET-NEXT, WALK), working as a gateway. Receives the target device IP, community string, operation to be realized and the accessed object OID as inputs. Outputs the operation results, received through SNMP;
- **Geolocation**: also an Adaptation component which interacts with *hostip.info* IP geolocation service. Receives an IP address as input and outputs the approximate geographic coordinates of the IP (*i.e.*, latitude and longitude); Its precision is limited, as are most, if not all, IP geolocation tools. This limitation is inherited by the created mashup, which depends on the service to position the routers in the map. Further discussion about IP geolocation precision is out of the scope of the present work;
- **Image extraction**: Adaptation component that extracts images from Web front-ends of external tools. It was used to pull graphics from MRTG, responsible for traffic statistics, and BGPe, responsible for announced routes statistics. This tools can be executed on the intranet or on the Internet, as the wrapper only requires a valid URL for them;
- **Graph**: a Reuse component (created using operation components) that aggregates information into a graph that represents the connections between a BGP router (*i.e.*, a root node) and its peers (*i.e.*, leaf nodes). Its output is a JSON string [17] that represents the graph;
- **Map**: a Visual component responsible for building the map. It is based on the information generated by the Graph component to draw the network topology, and display the retrieved statistics from the image component. The output is an HTML/JavaScript Web page that is rendered by users' Web browsers. For this work, we used the Google Maps API to create the maps.

In Figure 4, the design of the developed mashup is presented using the Business Process Modelling Notation (BPMN). The developer firstly inform the local BGP router address, its SNMP community string, and the MRTG and BGPe URLs on the developement environment. Based on that information, one SNMP wrapper retrieves the IPs of local BGP peers using the BGP-4 MIB module, and a second SNMP wrapper retrieves the IPs of external BGP peers. Having the set of external IPs, and using the MIB-2 module, the interfaces connected to external routers are retrieved from the local BGP router. For such interface, a SNMP request is done to retrieve the amount of traffic exchanged, and the correct statistics graphics from MRTG and BGPe. All IP addresses are so geolocated , and the graph operator assembles the representation of the network and feeds the map. Finally, The map wrapper creates the visualization of the mashup by placing visual elements representing the routers and connections on the map and annotating them with the relevant information.

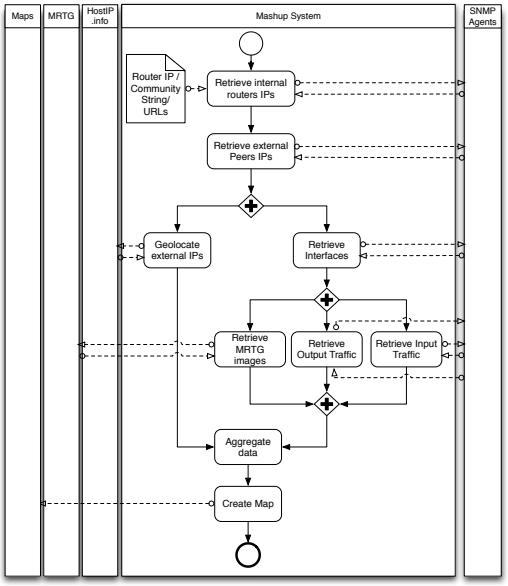

Fig. 4. Design of BGP router information visualization mashup

The wrappers (represented visually as adaptation components) have been developed as standalone applications. Thus, they can also be used by other composition systems that use a SOAP-based wrapper model or even a Java-based one, since the implementations under the SOAP interface were realized in Java. They have been designed with access interfaces that allow their internal implementation to be modified without affecting the external composition. For example, since the IP geolocation services are not accurate, the geo wrapper could be replaced by another one that access a better service.

The Runtime Environment, showed on Figure 5, presents the maps built to the user as Web pages, where the peers are represented as markers and the connections as lines linking those markers. Clicking on a visual element opens up a window where the user can access the pertinent information of the selected element, such as ASes involved in a connection, total traffic, traffic analysis graphics, and announced routes graphics. The user can also control the zoom level of the map according to his/her needs.

The information aggregated on the mashups allows an administrator to directly visualize the active BGP connections of a given set of managed ASes. Proceeding through a few intuitive interaction steps, mainly mouse clicks on elements representing BGP routers or connections, allows the manager to observe more detailed information as well as traffic and routes statistics. It is possible to insert more than one BGP router in one mashup, save it, and load the mashups again. The page itself is not saved, as it is dynamically created. Instead,

a mashup meta-description is persisted. When a mashup is loaded, this description is recovered and the composition is re-executed based on it, rebuilding the mashup with updated information. The mashup can also be programmed to update its information at runtime.

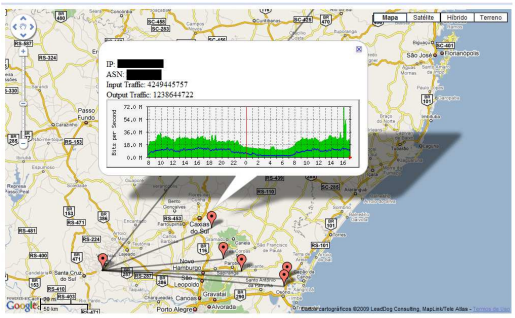

Fig. 5. *Runtime Environment* (BGP peering Mashup)

Originally, gathering BGP peering information requires the use of several tools with distinct interfaces. Besides, the integration itself is a complex process, which requires sophisticated scripts, handling large amounts of data, and effort to develop a suitable presentation to the results obtained. Our prototype reduces the interaction to the use of two user interfaces that do not require technical skills to be manipulated, keeping the complex technicalities of the integration transparent to the end user. In the development environment, the manager sets well known parameters (*e.g.,* IP addresses, URLs and SNMP community strings) so that the integration can be performed automatically by the underlying engine. The runtime environment exhibits the results of such integration into a map-based dynamic Web page. The popularity of map APIs and map-based Web pages is a strong indicative of their good usability, which the prototype inherits. The substantial decrease in the number of interaction steps and interfaces involved also contributes to the overall usability improvement of the prototype.

Having the discussion of prototype been presented, the next session will draw the conclusions obtained through its development and testing. After the presentation of the conclusions of this paper, we define our future work.

V. Conclusions and Future Work

The improvements in usability and agility provided by our prototype lead to the conclusion that the usage of mashups is both feasible and potentially advantageous to the network management field. The ever increasing growing of networks brings along more complexity to management, increasing the need for technologies that improve the network management process [16]. By enabling network managers to create their own management tools, a mashup-based approach can provide such improvement. It opens up the possibility to deal with situational issues that otherwise would be highly costly to address, such as the BGP peering issue presented. Verifying such possibilities provided by a mashup-based approach to network management is one of the main contributions of this work.

The development of the BGP peering mashup allowed us to observe that even exclusively network management information can be integrated with heterogeneous data through dynamic compositions. This can be useful for current networks, which usually employ different kinds of tools/services (*e.g.*, MRTG, Cisco Netflow, Looking Glass) in their infrastructures. We believe that with other wrappers (network management exclusively, or not), several other different management scenarios could be addressed. For example, integrating botnet related information to help network administrators to mitigate their attacks, employing tools such as online sandboxes.

The mashup-based approach is potentially better than traditional network management tools. It allows the creation of more meaningful information for the network administrators. The administrators don't need to spend time learning a specific programming language to create scripts. Situational changes in the management logic can be applied, tested, and deployed dynamically. Since mashups are essentially compositions, they can be shared with other administrators to be edited, and extended in a collaborative fashion. Due the decoupling of the building boxes of a mashup, its possible to spread them between different administrative domains, making possible a distributed management.

As future research, we aim at creating other wrappers, allowing our prototype be employed in different management situations. We will also investigate how to enable the compositions be run as daemons. They should be able to execute actions (to be represented as a new component), allowing not only the monitoring management, but also configuration management.

References

[1] J. Soldatos and D. Alexopoulos, "Web services-based network management: approaches and the wsnet system," *Int. J. Netw. Manag.*, vol. 17, no. 1, pp. 33–50, 2007.

[2] L. Z. Granville, D. M. da Rosa, A. Panisson, C. Melchiors, M. J. B. Almeida, and L. M. R. Tarouco, "Managing computer networks using peer-to-peer technologies," *Communications Magazine, IEEE*, vol. 43, no. 10, pp. 62–68, 2005. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2005.1522126

[3] T. O'Reilly, "What is web 2.0," 2005, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

[4] D. Merril, "Mashups: The new breed of web app - an introduction to mashups." 2003, http://www.ibm.com/developerworks/web/library/x-mashups.html.

[5] X. Liu, Y. Hui, W. Sun, and H. Liang, "Towards service composition based on mashup," in *Services, 2007 IEEE Congress on*, 2007, pp. 332–339. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4278815

[6] R. Ennals and D. Gay, "User-friendly functional programming for web mashups," in *ICFP '07: Proceedings of the 2007 ACM SIGPLAN international conference on Functional programming*. New York, NY, USA: ACM, 2007, pp. 223–234.

[7] J. J. Garret, "Ajax: A new approach to web applications," 2005, http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[8] N. Banerjee, K. Dasgupta, and S. Mukherjea, "Providing middleware support for the control and co-ordination of telecom mashups," in *MNCNA*. ACM, 2007, p. 11. [Online]. Available: http://doi.acm.org/10.1145/1376878.1376889

[9] J. Yu, B. Benatallah, F. Casati, and F. Daniel, "Understanding mashup development," *IEEE Internet Computing*, vol. 12, no. 5, pp. 44–52, 2008.

[10] V. Hoyer, K. Stanoevska-Slabeva, T. Janner, and C. Schroth, "Enterprise mashups: Design principles towards the long tail of user needs," in *IEEE SCC*. IEEE Computer Society, 2008, pp. 601–602. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/SCC.2008.88

[11] W. W. Eckerson, "Three tier client/server architecture: Achieving scalability, performance, and efficiency in client server applications," Open Information Systems, 1995.

[12] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web Services Web - An Introdution to SOAP, WSDL,and UDDI," *IEEE Internet Computing*, vol. 6, no. 2, pp. 86–93, 2002.

[13] Y. Rekhter and T. Li, "A border gateway protocol 4 (bgp-4)," United States, 1995.

[14] G. D. Battista, T. Refice, and M. Rimondini, "How to extract bgp peering information from the internet routing registry," in *MineNet '06: Proceedings of the 2006 SIGCOMM workshop on Mining network data*. New York, NY, USA: ACM, 2006, pp. 317–322.

[15] J. Wong and J. Hong, "What do we "mashup" when we make mashups?" in *WEUSE '08: Proceedings of the 4th international workshop on End-user software engineering*. New York, NY, USA: ACM, 2008, pp. 35–39.

[16] F. Kamoun, "Toward best maintenance practices in communications network management," *Int. J. Netw. Manag.*, vol. 15, no. 5, pp. 321–334, 2005.

[17] D. Rubio, "An introduction to json," 2007, http://dev2dev.bea.com/pub/a/2007/02/introduction-json.html.

Botnet Master Detection Using a Mashup-based Approach

Carlos Raniery P. dos Santos*, Rafael Santos Bezerra*, Joao Marcelo Ceron*,
Lisandro Zambenedetti Granville* and Liane M. R. Tarouco*
*Institute of Informatics - UFRGS
Rio Grande do Sul, Brazil
Email: {crpsantos, rsbezerra, jmceron, granville}@inf.ufrgs.br, liane@penta.ufrgs.br

***Abstract*—Botnets are considered by specialists, in both industry and academy, as one of the greatest threats to security on the Internet. These networks are composed by a large number of malware-infected hosts acting under a central command. They are usually employed to perform DDoS attacks or phishing scams. The behaviour of these botnets evolves due the adoption of new and sophisticated infection methods, changing of network protocols, and the employment of different command and control mechanisms. The security community, thus, is always dealing with such constant change. However, most botnet mitigation methods address just specific infection types or C&C protocols. We, therefore, propose a botnet mitigation approach based on the dynamic integration of pre-existing tools that can be employed together to achieve a more efficiently detection solution. To such end, we base our approach on a novel Web 2.0 technology called mashups to perform the information correlation. The proposal is extensible enough to allow even non-security information such as online mapping APIs be integrated to create more sophisticated compositions, and displaying the results in a more meaningful way.**

I. Introduction

The proliferation of personal computers and the decreasing costs of communication led the growing of computer networks, specially the Internet. Unfortunately, this growing comes associated with the increasing number of vulnerable hosts to infections. In the past, such infections were caused by viruses and worms that aimed to compromise and disable their targets. However this scenario has changed, virus creators no longer aim to just acquire recognition and attention from their community or the media. Now their main interest is making profit of their creations. Most of the infections are no longer committed to cause non functioning on the victims. Instead, infected hosts become *bots*, which can be remotely controlled by a human operator known as *master* or *botmaster*. Sets of such hosts are called *botnets* and are usually used to illegal purposes, such as large scale Distributed Denial of Service (DDoS) attacks, email *spamming* and *phishing* [1].

Botnets are considered the current largest security threat on the Internet [2]. One of the main reasons for this is the high number of infected computers. According to Vincent Cert [3], up to 25% of all computers on the Internet belong to botnets. Another reason lies on their effectively. Combining the efforts of hundreds or even thousands of machines widespread across the Internet to reach a single task such as DDoSing a server or spamming makes this approach very effective and hard to prevent. Due to such widespread nature, dismantling these networks is also a complex task. However, botnets conceptually have one single point of failure that can potentially be exploited, the botmaster. Identifying the botnet controller opens up many possible countermeasures, such as blocking communication, removing infected machines from the botnet, or even infiltrating this controller host in order to identify all participants of his botnet, as proposed by [4].

Most approaches to detect botmasters are based on detecting communication mechanisms used between the botmaster and the bots, which are known as *command and control* (C&C) channels. Centralized botnets have widely employed the Internet Relay Chat (IRC) protocol for many years [5]. Some efforts to extract information from the *malware* via IRC or disable its communication can be found, such as [1], [4] and [6]. However, more sophisticated botnets started to use other channels for communication. For instance, HTTP [7] is used to provide more stealthiness by mixing the C&C messages with regular Web traffic, also crossing firewalls more easily. Storm worm [8] employs Peer-to-Peer (P2P) communication to distribute the control of the botnet. Such approaches raise the difficulty of detecting botmasters, and nowadays there are just a reduced number of works that claim to address all of them [9].

One technique that can be used to discover information about the botmaster is the *sandboxing*. A *sandbox* is a controlled environment that allows safe execution of infected binaries and the observation of malware behavior. Such observation includes system calls, created and executed files and network activity. This technique has proven itself effective to detect the botnet controller [10]. However, its employment brings costs associated to the setup of the environment and the usage of commercial tools. Currently, there are some free, Web-based sandboxes, such as [11], [12] and [13], which employment can mitigate and even eliminate those costs, at the cost of a more limited functionality and database.

Given this scenario, we propose a mashup based approach to botmaster detection. Mashups are a recent Web 2.0 technology that proposes the agile and dynamic integration of external resources in a Web page, generating new functionality from such integration [14]. Our mashup based approach consists in integrating and correlating information from web available binary analysis tools (*i.e.*, online sandboxes and anti-viruses).

We employ other free web based tools, such as geocoding services and map APIs, to better present the results obtained. Our main goal with this proposal is analyze if the usage and integration of free, web available tools is viable and effective. We also provide to the community a new tool to both detect and visualize botmasters.

The remainder of this paper is organized as follows. In section 2, we depict our mashup-based solution. A qualitative evaluation of our proposal is presented in Section 3. In the Section 4 we lay out the obtained conclusions and present the future work.

II. New Approach

In this section, we will discuss our botnet mitigation solution. It consists on a botmaster detection tool that integrates and correlates information from online binary analysis, displaying the botmasters found on a map based on the geolocation of their IPs. It also provides an interface for binary submission and an online database that stores all the information about the binaries analyzed.

A. Created Mashup

We developed a mashup for botmaster and C&C Channel detection. This mashup, presented in Figure 1, integrates and correlates information from Web available binary analysis tools, such as sandboxes and anti-viruses. The obtained results are geolocated in a map to show the obtained results. Besides the components **Geolocation** and **Map**, already presented, we also developed two others: one to access sandbox and anti-virus online tools, and another to access an IP to ASN translation service (*e.g.*, Whois).

- **Analysis**: an Adaptation component that receives the binary file uploaded by the network administrator, and forwards it to online analysis tools. Its output is a set of information regarding of the infection kind, C&C Sever IP address, and time of the analyze;
- **Whois**: Adaptation component that retrieves the ASN based on an IP address. It was used to discover from which ISP the C&C Server is based. The information is retrieved from a Whois Web page, which receives just the IP address.

The mashup for botmaster detection receives as input infected binaries, which can be submitted by users or gathered in honeypots set by network administrators. Such binaries are then submitted to three online binary analysis tools. CWSandbox and Norman, will execute the binary and analyze its behavior, and Virus Total, will scan the binary for virus signatures. The information gathered from these three tools is then correlated. Such correlation aims to discover information about a possible C&C channel, and if sucessfully, try to identify the C&C Server.

With the C&C information, we employ our **Geolocation** and **Whois** components in order to discover both location of the C&C Server, and its responsible ISP. This information is then aggregated and sent to the **Map** component, which

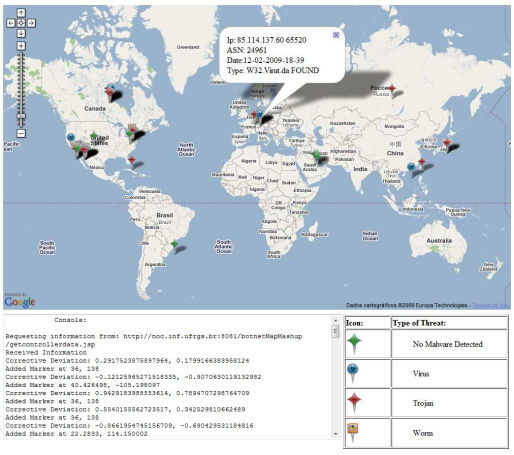

Fig. 1. Mashup for Botmaster Detection

generates an interactive map-based Web page. This Webpage allows the visualization of C&C Servers spread around the world through markers representing incidences of binary vectors of that botnet, as show in Figure 1. Furthermore, the information gathered is also available in a data feed of detected botnet vectors and C&C Servers using different data formats (*i.e.* XML and JSON). This information is also timestamped, allowing the creation of temporal representations of the infections.

The composition of results from many sandbox and anti-virus online tools is an alternative to the identification and mitigation of botnets based on a centralized approach. It replaces the usage of complex and paid tools with the integration of free, simpler ones. The adaptability obtained with such mashup-based approach is important due to the heterogeneity of technologies employed in C&C channels. Since each sandbox tool is able to analyze and detect a specific set of technologies, the usage of multiple tools allows the mashup to detect different kinds of infections. Furthermore, it allows constant self-atualization, through the addition of new tools to detect new kinds of infections. The support of new analysis mechanisms is possible by the creation of wrappers, which is a straightforward process.

Finally, mashups are built with reusability in mind. In this case, this characteristic allows the created mashup to be used as a building block for more sophisticated botnet mitigation mashups. For instance, it is possible to develop a mashup able to identify P2P botnets. Further possibilities include the implementation and addition of firewall filters to automatically isolate the C&C channel from the network potential hosts, and automatically detect infected hosts inside the network through flow analysis.

III. Evaluation and Verification

In this section, we evaluate our mashup for botnet mitigation approach. This evaluation is divided in two aspects. Firstly, we evaluate our mashup based approach qualitatively, doing an analysis of its advantages and limitations. We then make a comparative evaluation, where we compare our approach with the current botnet mitigation and information gathering techniques presented in Section 3.

A. Qualitative evaluation

In order to qualitatively evaluate our approach, we determined some characteristics that will be discussed in details following. Such characteristics, presented bellow, will be the focus of this subsection:

- **Flexibility:** relates to the capability of adaptation to new scenarios, such as dealing with new botnets or new C&C technologies;
- **Extensibility:** the possibility of reuse by different tools or different approaches;
- **Implementation effort:** the effort needed to implement the theoretical approach;
- **Usage effort:** the effort needed to use the botnet mitigation mashups once they are implemented;
- **Reliability:** how reliable are the botnet mitigation mashups, specially under critical scenarios.

The flexibility of our mashup is based on the capability of integrating new botnet information and mitigation tools. This is special interesting for botnet mitigation since such networks are always evolving. New protocols for C&C channels, infection methods, and obfuscation techniques are always being used and obligating the employment of new techniques and tools to deal with them. Wrappers can be created to these new tools [15], enabling them be integrated in our botnet mitigation mashup. Additionally, non-exclusive security tools can be employed to increase the effectiveness of the mashup approach. For instance, a module to analyse Netflow flows and find C&C channels can be developed, and a firewall module can dynamically create firewall rules to stop such communication.

The flexibility is obtained due the modularity of our mashup system. We proposed and implemented modules (*i.e.*, wrappers) to access different online sandboxes and anti-viruses systems. Such modules was composed with pre-existing modules (*i.e.*, geolocation and mapping) created originally to other context (*i.e.*, BGP peering) [15]. The result of the composition (*i.e.*, the botnet mitigation mashup presented in Section 4), can be reused and extended by other users of the mashup system, but also, by users whom don't use the system, for example, by accessing the report available in different data formats (*i.e.*, JSON, CSV, XML). This report includes information the following information: type of infection, botnet name, ASN of the infected host, and geographic and chronologic information.

The reduction of the implementation effort is obtained by the existence of wrappers to access the external systems. In fact, using a mashup-based approach reduces this effort to the creation of independent wrappers by software developers. The integration of those wrappers is responsibility of the network security administrator, who knows well about security but is not necessarily a skilled programmer. About the usage effort, the mashup approach gives the end user the possibility to easily manipulate the components and alter the composition, which adds another layer of extensibility, as previously discussed.

We are aware of the reliability limitation of our approach. This happens because the mashup depend on free online and external tools, which can fail. Problems such as offline services, overloaded services, discontinued services and API changes may arise with the usage of such online tools. Redundancy can be, and was, used to mitigate this issue. Similar services are available online, such as the sandboxes employed in our implementation. Furthermore, once the information about botnet infection is gathered, it is stored in the mashup system. In case of failure of the external services, binary analysis is compromised, but existing reports remain functional and useful. Thus, reliability of our approach is adaptable to the administrator needs, but it demands an adequate choice of components.

IV. Conclusions

In this paper, we proposed a mashup-based botnet mitigation approach. In our proposal, we cover the integration of various botnet mitigation tools and techniques with external resources in botnet mitigation mashups. Based on this proposal, a mashup prototype was developed with information about botnets detected using free online sandboxing and anti-viruses tools with geocoding, Whois and mapping online services.

Due to the possibility of integration of various techniques, including ones yet to come, to cover different kinds of botnets, we reached the conclusion that our mashup-based approach is adequate to the ever changing scenario of botnet mitigation. However, this approach does not aim to substitute other botnet mitigation efforts, but to leverage and correlate them to optimize the obtained results and create new ones from the integration. This is further reinforced by the comparison with other approaches performed in our evaluation section, where botnet mitigation mashups showed superior criteria coverage compared with current Vertical Correlation and Horizontal Correlation approaches. It is our understanding that the fluid and dynamic nature of botnets as a security threat demands extensible and flexible solutions, as is ours.

Having implemented our botnet mitigation mashups manually and aided by a mashup system, we have come to the conclusion that its usage reduces the effort needed to materialize our approach and further enhances its flexibility and extensibility. With the aid of a mashup system with adequate wrappers, the integration can be performed by a network administrator and/or a security expert, and the results better reflects their needs. Changing and extending this mashups becomes similarly easy. Furthermore, creating wrappers to the resources needes (IPSs, IDSs, anti-viruses, sandboxes) is a less demanding process than implementing the mashups manually.

Our implementation, while providing good information and coverage, used freely available components (*i.e.*, online sandbox, anti-viruses, geolocation tool, mapping API), remaining completely free. However, reliability and maintenability suffer due to the lack of guarantee of service of these resources. In more critical implementations, we recommend the usage of more reliable tools.

Our proposal sees botnet mitigation mashups as extensible tools, and our implementation reflects this view. To this end, the C&C information map code is available to be extended and the information used to create the map is available in a report. We delneated cases where this information can be further extended, For instance, the integration with a firewall to automatically create rules to block communication with botmasters and/or the integration with flow analysers to identify infected hosts within the network are both cases to be explored.

The search for online resources to be integrated and how these resources can be leveraged to mitigate botnets is also a research opportunity. Plenty of tools, be them security related (such as computer diagnosis tools) or not (such as RSS feeds) are available online to be *mashed up*, and they bring a host of possible new mashups. On the other hand, the evaluation of the integrability of current security tools such as IDSs, IPSs, firewalls, anti-viruses in a mashup based approach is also a research opportunity which we intend to chase. Evaluate if this tools offer, for instance, web interfaces, web service interfaces and computer readable reports and how those can be used in a mashup based approach.

REFERENCES

[1] M. K. Paul Bacher, Thorsten Holz and G. Wicherski, "Know your enemy:tracking botnets," Tech. Rep., 2005. [Online]. Available: http://www.honeynet.org/papers/bots

[2] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, "Peer-to-peer botnets: overview and case study," in *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. Berkeley, CA, USA: USENIX Association, 2007, p. 1. [Online]. Available: http://portal.acm.org/citation.cfm?id=1323129

[3] E. Stinson and J. C. Mitchell, "Towards systematic evaluation of the evadability of bot/botnet detection methods," in *WOOT'08: Proceedings of the 2nd Usenix Workshop on Offensive Technologies*. San Jose, CA, USA: USENIX Association, 2008.

[4] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, February 2008.

[5] J. Goebel and T. Holz, "Rishi: identify bot contaminated hosts by irc nickname evaluation," in *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. Berkeley, CA, USA: USENIX Association, 2007.

[6] D. Dagon, G. Gu, C. Lee, and W. Lee, "A taxonomy of botnet structures," in *Proceedings of the 23 Annual Computer Security Applications Conference (ACSAC'07)*, December 2007.

[7] N. Daswani and M. Stoppelman, "The anatomy of clickbot.a," in *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. Berkeley, CA, USA: USENIX Association, 2007, pp. 11–11.

[8] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling, "Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm," in *LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*. Berkeley, CA, USA: USENIX Association, 2008, pp. 1–9.

[9] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the 17th USENIX Security Symposium (Security'08)*, 2008.

[10] C. Willems, T. Holz, and F. Freiling, "Toward automated dynamic malware analysis using cwsandbox," *IEEE Security and Privacy*, vol. 5, no. 2, pp. 32–39, 2007.

[11] International Secure Systems Lab Vienna University of Technology, Eurecom France, UC Santa Barbara, "Anubis: Analyzing Unknown Binaries," available at: http://anubis.iseclab.org/. Accessed in April 2008.

[12] Ben Stock 2007-2008, Lehrstuhl fr Praktische Informatik 1, University of Mannheim, "CWSandbox - Behavior-based Malware Analysis," available at: http://cwsandbox.org/.com. Accessed in April 2008.

[13] Norman SandBox Information Center (NSIC) , "Norman Sandbox Information Center," available at: http://http://www.norman.com. Accessed in April 2008.

[14] D. Merril, "Mashups: The new breed of web app - an introduction to mashups." 2003, http://www.ibm.com/developerworks/web/library/x-mashups.html.

[15] R. S. Bezerra, C. R. P. dos Santos, L. Z. Granville, and L. Tarouco, "On the feasibility of web 2.0 technologies for network management: A mashup-based approach," in *NOMS 2010 - TechSessions*, Osaka, Japan, apr 2010.

On Using Mashups for Composing Network Management Applications

Carlos Raniery Paula dos Santos, Rafael Santos Bezerra, João Marcelo Ceron,
Lisandro Zambenedetti Granville, Liane Margarida Rockenbach Tarouco
Federal University of Rio Grande do Sul – Institute of Informatics
Av. Bento Gonçalves, 9500 – Porto Alegre, RS – Brazil
{crpsantos, rsbezerra, jmceron, granville, liane}@inf.ufrgs.br

***Abstract*—Mashups are Web applications created through the integration of external resources available on the Web. They have been considered a hallmark of Web 2.0 technologies, allowing end users to develop their own applications and encouraging cooperation and reuse. However, their usage in the network management field remains unexploited. In this context, we look at Web 2.0 as a feasible mechanism able to integrate heterogeneous management information. In this paper, we propose an architecture and a system prototype that allows network administrators to design their own management applications through the composition of external resources. The creation of mashups for two network management scenarios allowed us to both evaluate our architecture and, mainly, observe benefits, challenges, and opportunities that arise from such approach. Observed characteristics such as easy-of-use, extensibility, and context specific development candidates mashups to be an interesting area for further research in the network management field.**

***Index Terms*—Mashups, Web 2.0, Network Management, Information Composition**

I. Introduction

Experienced network administrators normally use a variety of different tools when managing modern computer networks. These tools are usually built without considering the administrators' need to use them in an integrated fashion; integration among management tools is rarely a design requirement. This situation forces network administrators to find their own ways of integrating network management tools. For example, in order to check whether neighbor Autonomous Systems (ASes) are exchanging network traffic without violating predefined Service Level Agreements (SLAs), an administrator has to manually access the ASes routers, monitor the exchanged traffic, and compare the traffic with the SLAs definitions. Although the issue of retrieving management information from network devices can be solved, for example, using the Simple Network Management Protocol (SNMP), there is no widely accepted or employed solution for combining large amounts of management information retrieved from different sources.

Today, it is not uncommon to find situations where management information ends up being integrated through scripts written by network administrators in a 'domestic' ad hoc way. This means administrators must be skilled script programmers as well as experts on the networks they manage. Novice administrators, however, are rarely experienced in both managed network and script coding, and training them is both time consuming and expensive. One way to address this situation is to improve the process of developing management tools using methods that are easier for inexperienced administrators to learn and carry out. These methods can also be used in the management of very context-specific situations where traditional methods like scripting are not viable because of time and cost constraints.

A new kind of Web 2.0 application known as *mashups* [1] is being widely advertised as an enabler for users with no programming expertise to compose new applications and services through the collection of information from different sources typically available on the Web. Mashups focus the data collected from these sources into a single user-friendly presentation. Mashups are composed on *mashup systems*, which employ high-level abstractions and user-friendlier interfaces that hide technical details (e.g., underlying network protocols and multiple data parsers). To build their mashups, inexperienced users, acting as developers, access a mashup system using a Web browser and define their compositions by interconnecting visual elements that represent the actual available sources of information. Users/developers can then share their creations on the Web(i.e., the built mashups) with other interested users.

We believe that the benefits brought by mashups can be exploited in the context of network management. There is a real opportunity for that because a number of important management tools already expose their results through Web interfaces. For example, the widely available graphs of the Multi Router Traffic Grapher (MRTG) tool could be included as part of more complex management mashup applications. In addition, information from Web systems not originally conceived for network management, such as Geolocation and Looking Glass, could be incorporated into management mashups, thus enabling the composition of even more sophisticated mashups.

In this article we examine the potential benefits of mashups as management applications created by inexperienced network administrators through the composition of heterogeneous resources. The rest of the article is organized as follows. First, we review key concepts related to mashups and mashup systems and present examples of Web-available resources that could be used to create network management mashups. We then introduce a system architecture and prototype to support the retrieval of heterogeneous information in an easy-to-use, management composition environment. Next, we use case studies to present two mashups we created in the proposed

mashup system for two relevant management tasks: detecting Border Gateway Protocol (BGP) peering problems, and identifying botnet communication channels. These mashups allowed us to better understand the management benefits obtained with them, which are summarized, along with concluding remarks and future research opportunities and directions, in the closing section of this article.

II. Mashups

Recently, the concept of Web 2.0 has been receiving much attention from both industry and academia. The term Web 2.0 was introduced by Tim O'Reilly [2] to define a set of Web applications based on the principles of interactivity and cooperation. Central to this paper are the Web 2.0 applications known as mashups. Mashups are built from the composition of pre-existing external resources available on the Web (e.g., RSS feeds, Web pages and Web Services) [1]. A mashup may be ad-hoc programmed by a developer through traditional programming techniques, but it may also be defined by users with no programming expertise through *mashup systems*.

Mashup systems are Web-based development tools that enable end-users to create their own mashups; they are what an Integrated Development Environment (IDE) is to a programming language. Usually, mashup systems perform three main tasks: provide users with visual tools to define new mashups, store created mashups in mashup repositories or libraries, and execute mashups when requested. The three tasks are often performed from a single mashup system, but it is possible for a mashup to be created in one system, executed in another, and stored in a third to form a remote mashup repository. Figure 1 depicts the basic mashup system architecture.

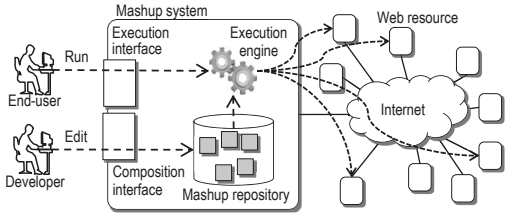

Fig. 1. Mashup System Architecture

Figure 1 shows two different users: the developer and the end-user. Developers define or edit compositions that result in the final mashups stored in a mashup repository. End-users then access the mashups that interest them. When a mashup is accessed from the repository, it runs on top of an execution engine to materialize the composition previously specified. In this process, the execution engine retrieves information from remotely located Web resources (using technologies like SOAP, RPC, and Syndication protocols), performs operations (*e.g.*, sorting, filtering, aggregating) on the retrieved information, and finally builds a Web page that presents all the combined results.

Although Figure 1 separates developers and end-users, both roles can be played by the same person. In fact, the possibility of unskilled end-users acting as developers is one of the key benefits advertised by mashup advocators. Whether such benefits really hold depends, among other factors, on the problem the end-user aims to solve and the capability of the mashup system to hide technical details without losing expressiveness in the composition definition.

The access and representation of external information by mashup systems could be further enhanced by the more recent concept of Web 3.0. This refers to a semantic Web [3], where both information and services have meta information to describe them, enabling data to be processed by both humans and computer software. The availability of meta information could allow mashup systems to use more dynamic methods of representing external information. However, Web 3.0 models and concepts are still in the early stages of definition and are not ready to be effectively employed in mashup systems as of today.

III. Web-Available Resources

Nowadays, network administrators rely on many different tools for managing their networks. Most of these tools are designed to address specific network management problems. However, there are other useful tools conceived in other contexts that could be used by network administrators. In this section, we present examples of network management tools as well as general-purpose tools that would likely work well in network management mashups.

A. Network Management Web Tools

The Multi Router Traffic Grapher (MRTG) is a Web tool widely used in network management. In its most common setup, MRTG shows network traffic histograms plotted from information polled from network routers via SNMP. The popularity of MRTG motivated the development of more sophisticated tools like RRDTool and SmokePing, although in essence these tools just advance over the quite successful MRTG's original solution. Since the graphs generated from these tools are available on the Web, they can be integrated into other systems and show compositive information of the managed networks. Examples of organizations that use such tools include the Brazilian National Education and Research Network (RNP) and the Swiss academic backbone SWITCH.

Another example of a relevant tool is Cisco's NetFlow, which is used in network routers to capture IP traffic information (*i.e.*, network flows) for periodic exporting to remote collector servers, where the flows may be analyzed by other tools. Although NetFlow is a technology originally promoted by Cisco, there are many 3rd party (and free) tools to process, manage, and visualize network flows. Two of these, FlowScan and NTop, are tools used to create graphs and visualizations from already analyzed flows. Like MRTG, FlowScan and NTop graphs can also be displayed on the Web and thus potentially integrated with other tools. The University of Houston and Cisco are examples of organizations that use FlowScan and NTop to visualize captured flows.

ISPs, telecommunication companies , and universities usually provide Looking Glass Web sites to help experienced users

conduct network diagnostics. These sites show information relative to the backbone infrastructures of each organization, in addition to offering various other tools like ping, traceroute, and BGP information retrieval on the same integrated Web page. Since the archived data is public, a remote network management system would only need to parse the HTML pages that Looking Glass generated to extract the relevant information.

B. General Purpose Web Tools

Beyond the tools specifically designed for network management, other tools created for different purposes could help network administrators in their tasks.One example is IP Geolocation services, which provide geographic location (*i.e.*, latitude and longitude coordinates) given the IP address of a network element. Geolocation services are often used by Web masters to keep track of the geographic distribution of visitors to a Web site. In network management, such services could be useful for security purposes to find, for example, spammers' location. Examples of popular IP Geolocation services include Geobytes and IP2Location. These services, however, are not always precise because they depend on the accuracy of the geographic information provided by ISPs.

Further examples of online available tools that could be employed in network management are online mapping services. Network maps are a traditional, widely employed, network management tool. They help administrators to represent and visualize the network topology, grouping together their managed devices based on defined parameters (*e.g.*, inside an enterprize, in a city, in a country). Online mapping services can be used to create highly interactive, cross-platform Web-based maps. Examples of such services include Google Maps, Yahoo Maps, and Microsoft Bing Maps. Such services also offer Application Programming Interfaces (APIs) that allow Web developers to create their own maps.

These are only a few of the many resources that could potentially be combined into network management mashups. Several other examples can be used in a production environment, where an administrator often has context-specific needs that cannot be addressed by traditional software development methodologies because of time and cost constraints. In the next sections, we show how these and other Web-available resources can be integrated to compose mashups devoted to specific management problems.

IV. An Architecture for Mashup-based Network Management Systems

To investigate the feasibility of mashups for network management, we defined in a previous work [4] an architecture for mashup-based network management systems. Based on this architecture, we developed a prototype tool that enabled us to create two mashups and thus understand the possible benefits for network management. It is important to note that, while this architecture is designed for network management systems, it is generic enough to describe even general purpose mashup systems.

A. System Architecture and Tiers

The system architecture and the relationship among its components are presented in Figure 2. We assume two user roles (developer and administrator) that can be occasionally performed by the same person. The developer represents network administrators interested in composing their own mashup-based management applications. The administrator represents users interested only in mashups already created. In addition, we designed the architecture to enable the collaboration among several developers in order to create more sophisticated management mashups.

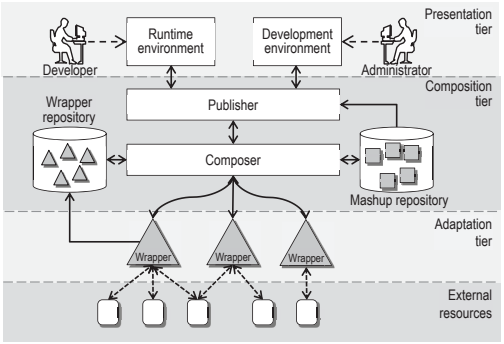

Fig. 2. Proposed Architecture

The proposed architecture is based on the three-tier client-server model (*i.e.*, logic, presentation, data). The tiers run as logically separated processes, allowing changes to be made as needed on one tier without affecting the other tiers. For example, changes in the external resources do not affect tiers other than the adaptation tier. Our three tiers are:

- **Adaptation tier**, responsible for accessing external resources and presenting, internally to the architecture, a common access interface, exposed by *wrappers*, that hides the specificities of external resources (*e.g.*, access method and information modeling);
- **Composition tier**, responsible for composing management information using the *composer* component, which orchestrates calls to external resources and performs operations (*e.g.*, sorting, filtering, aggregating) over the retrieved information, and the *publisher* component, which controls the access to the available mashups in the mashup repository;
- **Presentation tier**, provides the Web environments for developers to define their compositions (through the *development environment*), and for administrator to access and run the available mashups (through the *runtime environment*).

B. Interaction Elements

In order to allow inexperienced network administrators in programming techniques to build their own management applications, high-level abstractions of the external data sources

are used. These abstractions hide technical details such as underlying network protocols and multiple data parsers. To achieve that, we propose a set of interaction elements that are available as visual boxes to developers so that they can define the management logic behind a mashup. Five types of interaction elements are proposed, although the architecture is flexible to support new ones. These five elements are:

- **Visual**, that supports different ways of displaying (*e.g.*, map, table, tree) the results of a composition;
- **Control**, that represents basic programming logic, such as loops and conditions;
- **Operation**, that performs operations on the retrieved data, like filtering, merging, aggregating, arithmetic operations, and boolean logic operations;
- **Adaptation**, that represents the external resources accessed by the system;
- **Reuse**, that is used to extend the available mashups to build more sophisticated compositions.

C. Mashup System Prototype and Technologies

Based on the presented architecture, we developed a network management mashup system prototype. We use Simple Object Access Protocol (SOAP)-based Web services as wrappers given their cross-platform capabilities and widespread standardization. Wrappers are called by the Composer component, which is a server-side JavaServer Pages (JSP)-based engine. This engine communicates with the development environment, which is an Asynchronous Javascript And XML (AJAX)-based Web system created with the WireIt library. The development environment presents a drag-and-drop interface for mashup development, where users connect multiple visual blocks to define their compositions. Each block represents an interaction element with its inputs and outputs.

Today, there are no available mashup systems able to integrate network management Web tools (*e.g.* MRTG) and protocols (*e.g.* SNMP). Therefore, our purpose with our prototype is to prove the concepts of employing mashups for network management by using an actual system in specific management scenarios. In the next sections, we present our mashups compositions using the proposed mashup system for two relevant management tasks. We also detail how the presented interaction elements were employed to define the mashup composition logic.

V. Case Study 1 - BGP Peering

The first case study concerns communication breakdowns among the independent and interconnected administrative domains that make up the Internet. Mostly owned by Internet Service Providers (ISPs), these domains are referred to as Autonomous Systems (ASes) because their administrators define polices (*e.g.,* for routing, traffic shaping, and traffic priorities) independently from other domains. Connections between ASes are usually maintained through the use of the Border Gateway Protocol (BGP). When two ASes establish a connection, one AS announces routes to the other and vice-versa through BGP. These announcements (which are basically a set of IP address prefixes) indicate that an AS wants to receive traffic from the other neighbor AS. Two interconnected ASes are called *peers*, and the ASes interconnection is known as BGP peering [5].

The main issues of BGP peering are related to unexpected behavior of the inter-AS relationship, for example, when the amount of traffic routed from/to a peer suddenly and dramatically exceeds or falls below an expected, usually agreed upon, traffic amount. The cause is an inappropriate peer operation, for example, when a peer is not announcing its prefixes to its neighbor peer, or when the peer is announcing an unexpected number of routes, possibly violating an SLA. This can happen because of misconfigurations or malfunctioning of network devices. Whichever the reason, the issue should be solved as soon as possible, given the large volume of traffic usually exchanged between ASes and the financial costs incurred by these exchanges. In order to properly monitor the status of BGP peering, a network administrator, located inside an AS, needs access at a minimum to the following information:

- **Local and remote BGP peers** - A local BGP router is connected to remote BGP routers that belong to other ASes (*i.e.*, remote peers). Information about local and remote peers can include unique identifiers (Autonomous System Number - ASN), ISP owner, and location of BGP routers;
- **Number of announced routes** - The number of routes announced by the local peer to remote ones, as well as the number of route announcements received from the remote peers to the local one should be available. Usually, the number of announced routes is defined by AS administrators when defining SLAs;
- **Traffic exchanged** - The administrator needs to be able to check the amount of traffic exchanged between ASes. There is a financial cost associated to this exchange, which is also usually defined in SLAs. Network administrators may define policies and strategies to split traffic between different peers in order to optimize costs.

The BGP-4 Management Information Base (MIB) module, standardized by the Internet Engineering Task Force (IETF), provides both peer and routing information via SNMP. However, analysis of such information is not trivial [6]. For example, in order to discover the IP addresses of remote ASes routers, the information retrieved from the local router's BGP-4 MIB module must be complemented with information from another MIB module: the MIB-II. Also, to find the number of routes announced by each remote peer, the whole BGP-4 routing table must be retrieved from the local router via SNMP. That is so because although this information (*i.e.*, number of routes announced per peer) can be retrieved from the BGP-4 routing table, it is not supported in the BGP-4 MIB module [5].

The task of determining the total amount of exchanged traffic is even more complex. It requires checking which local network interfaces are connected to remote peers, and accessing the objects `ifInOctets` and `ifOutOctets` of the MIB-II module for each of these interfaces. This is insufficient for richer traffic analysis, however, because these two objects do not provide, for example, temporal traffic averages, thus forcing the management station to poll the local BGP router

in order to calculate the temporal traffic behavior on each interface. This is in fact the technique usually employed by traffic monitoring tools such as MRTG. However, integrating such tools with information retrieved from the BGP-4 MIB module is not easy. The whole process is usually manually executed by the network administrator or, in some cases, domestically automated through script programming.

A. A Mashup for Monitoring BGP Router Information

We developed a mashup to enable network administrators to monitor the BGP peering scenario described above. The mashup aggregates routing information from local BGP routers and their neighbor remote peers on a map-based Web page. Map mashups are currently the most widely used mashup category on the Web because of the increasing popularity of map APIs. Map-based displays also make sense here because network maps are traditional management tools used for the visualization of management topologies [7]. To build the mashup, we employed the following interaction elements:

- **SNMP** - The SNMP element enables the access to SNMP managed devices through an SNMP wrapper, which supports the main protocol operations (*e.g.*, GET, GET-NEXT, GET-BULK, SET). The element receives as input parameters the target device IP address, community string, protocol operation to be realized, and the MIB Object Identifier (OID) of interest. Its outputs are the operation results;
- **Geolocation** - This element interacts with the *hostip.info* IP geolocation service. It receives an IP address as the input parameter, and outputs the approximate location of the device through geographic coordinates (*i.e.*, latitude and longitude); The precision of the Geolocation element is limited by the precision of the hostip.info service, as in any other IP geolocation service;
- **Image Extraction** - This element extracts images from the front-ends of Web-based tools. It is used to pull graphics from MRTG (responsible for traffic statistics), and BGPe (responsible for BGP announced route statistics);
- **Graph** - The Graph element aggregates external information into a graph that represents the connections between a BGP router (*i.e.*, a root node) and its peers (*i.e.*, leaf nodes). Its output is a JSON string that represents the graph;
- **Map** - This last element builds a visual map. It uses the information generated by the Graph element to draw the network topology, and displays the retrieved statistics from the Image Extraction element. The output of the Map element is an HTML/JavaScript Web page that is rendered by users' Web browsers. In this work, we used the Google Maps API to create our maps.

Figure 3 shows the composition of our BGP monitoring mashup, rendered in Business Process Modeling Notation (BPMN). First, the developer enters the local BGP router address, its SNMP community string, and MRTG and BGPe URLs on the development environment. Based on that information, an SNMP wrapper retrieves the IP addresses of local BGP peers using the BGP-4 MIB module, and a second SNMP wrapper retrieves the IP addresses of external BGP peers. Once the set of external addresses is obtained, the network interfaces connected to external routers are retrieved from the local BGP router accessing the MIB-II module. For each interface, an SNMP request to retrieve the amount of traffic exchanged is sent, and the correct statistics graphics from MRTG and BGPe are requested. All IP addresses are geolocated, and the graph element assembles the representation of the network and feeds the map. Finally, the map wrapper creates the visualization of the mashup by placing visual elements representing routers and connections, and annotating them with the relevant information.

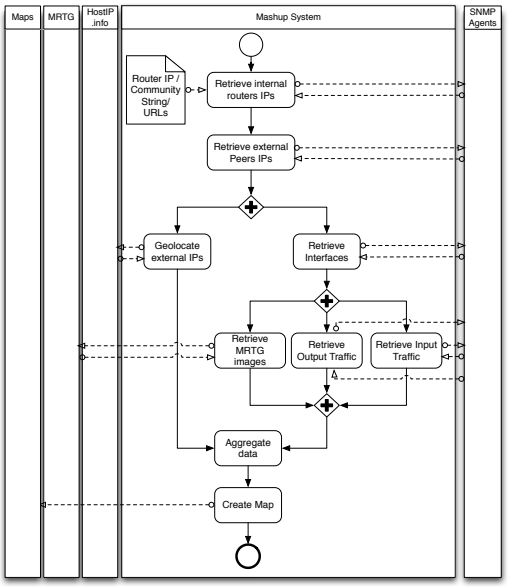

Fig. 3. Design of BGP Router Information Visualization Mashup

The wrappers (visually represented as adaptation elements in Figure 2) have been developed as standalone applications. Thus, they can also be used by other composition systems that use a SOAP-based wrapper model or even a Java-based model, since the implementation under the SOAP interface has been coded in Java. In addition, the wrappers have been designed with access interfaces that allow their internal implementation to be modified without affecting the external composition. For example, since the accuracy of IP geolocation varies, the geolocation wrapper can be replaced by another wrapper that accesses a more accurate service.

The runtime environment, presented in Figure 4, shows the built maps on Web pages, where peers are represented by symbols, and the connections among them as lines linking the symbols. Clicking on a visual element opens up a window where the user can access the pertinent information of the selected element, such as the ASes involved in a connection, the total traffic, the traffic analysis graphics, and the announced

route graphics. Users can also control the zoom level of the map according to their needs.

The information aggregated on the mashups allows an administrator to directly visualize the active BGP connections of a given set of managed ASes. Proceeding through a few interaction steps, mainly mouse clicks on elements representing BGP routers or connections, allows the user to observe more detailed information as well as traffic and route statistics. It is possible to insert more than one "seed" BGP router in a mashup, save it, and load the mashup again. The page itself is not saved, since it is dynamically created. Instead, a mashup meta-description is stored. When a mashup is loaded, the mashup description is recovered and the composition is re-executed, rebuilding the mashup with updated information.

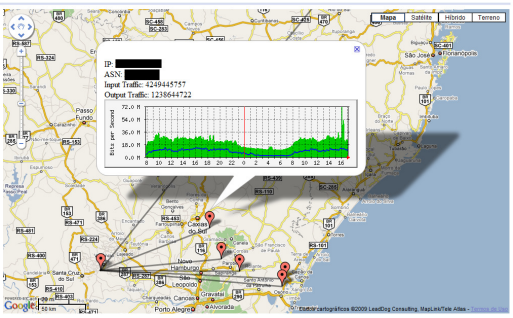

Fig. 4. Runtime Environment (BGP peering Mashup)

Traditionally, gathering BGP peering information has required the use of several tools with distinct user interfaces. Integration of the data collected has been a complex process requiring sophisticated scripts, handling of large amounts of data, and much effort to develop a suitable presentation of the results obtained. Our prototype reduces the number of interaction levels for the user down to two user interfaces that do not require technical skills to be manipulated, while also keeping the complex technical details of the integration hidden from the end-user. In the development environment, the developer sets well known parameters (*e.g.,* IP addresses, URLs, and SNMP community strings) so that the integration can be performed automatically by the underlying engine. The runtime environment displays the results of the integration on a map-based dynamic Web page. The popularity of map APIs and map-based Web pages is a strong indication of their usability, which our prototype inherits. The substantial decrease in the number of interaction steps and interfaces involved also contributes to the overall usability improvement of the prototype over traditional methods.

B. Extending the Mashup

It is possible to compose more sophisticated mashups based on our initial mashup, which can be used in management scenarios other than the one presented. For example, in a scenario where one AS provides international traffic access to regional ISPs and the communication between them is essentially UDP, errors at the AS that cause UDP packets to be discarded may not be detected by the ISPs. To solve that problem, Quality of Service (QoS) parameters can be included in the mashup to help network administrators of the local ISPs diagnose the problems in the network.

QoS-related parameters such as queue size, discard rate, and CRC errors can be employed. For example, if the network administrator of a regional ISP is informed that its international traffic via an AS is suddenly being discarded at an abnormally high rate, this may suggest a botnet infection and the ISP might want to quarantine traffic coming from the AS by blocking all the input traffic from this peer. In our mashup, such QoS parameters could be retrieved just after discovery of the network interfaces that are connected to remote peers, in parallel with the input/output traffic discovery. New interaction elements can be added to the composition to access, for example, the objects `ifOutQLen`, `ifInDiscards`, `ifOutDiscards`, `ifInErrors` and `ifOutErrors`.

VI. Case Study 2 - Botnet Detection

The proliferation of personal computers and the decreasing costs of communication have led to ever growing, large computer networks. Unfortunately, with this growth there is an ever increasing number of hosts vulnerable to infections. In the recent past, such infections were generally caused by viruses and worms designed to compromise and ultimately disable their targets. Nowadays, computer virus developers are more interested in the money they can make from their creations. One of the most widespread ways they do this is by transforming infected hosts into *bots*, creating a network of compromised machines known as a *botnet*.

Botnets are usually used for illegal purposes, such as large scale Distributed Denial of Service (DDoS) attacks, email spamming and phishing [8]. From a technical view, *botnets* are widely diffused and highly heterogeneous networks, typically crossing ISPs, countries, and continents. Bots (*i.e.*, compromised computers) can be remotely controlled by an operator known as *master* or *botmaster*. To control a botnet, botmasters employ an infrastructure of communication composed of a set of Communication and Control (C&C) channels through which they can launch commands to be executed by bots. The main propose of these channels is thus to propagate instructions from the botmaster to the compromised machines within the same infrastructure. These channels are able to operate over different network topologies and use a variety of communication mechanisms, including Internet Relay Chat (IRC), HTTP, peer-to-peer (P2P) protocols, and even quite recent Web tools such as *Twitter*.

The first botnets were based in centralized architectures using IRC or customized protocols for communication. Even today, many botnets are still designed this way. Although these centralized structures are effective, they have, by definition, one single point of failure, the *C&C server*, which is a central point where all bots connect, waiting for instructions from the botmaster. Disabling the C&C channels used by the C&C server is an effective measure to stop the malicious actions launched by a botmaster.

One of the main techniques to discover information about C&C channels is through malware analysis. A scalable way to automate the malware analysis process is with a *sandbox*. A *sandbox* is a controlled environment that allows safe execution of infected binaries and observation of malware behavior. Such observation includes tracking down system calls, created files, executed files, and network activity. Currently, there are some free, Web-based sandboxes such as Anubis, CWSandbox, and Norman.

A. Botmaster Detection Mashup

We developed a mashup for botmaster and C&C channel detection. This mashup, depicted in Figure 5, integrates and correlates information from Web available binary analysis tools, such as sandboxes and anti-viruses. The obtained results are geolocated on a map to show the locations of detected botmasters. Besides the **Geolocation** and **Map** elements previously introduced, we also developed two other elements: one to access sandbox and anti-virus online tools, and another to access an IP to Autonomous System Number (ASN) translation service (*e.g.*, Whois).

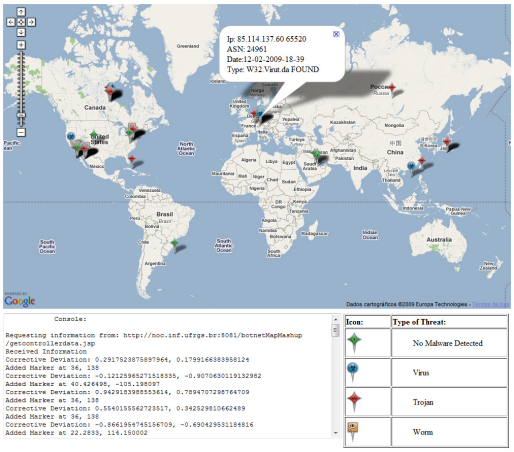

Fig. 5. Mashup for Botmaster Detection

- **Analysis** - The analysis element receives the binary file uploaded by the network administrator and forwards it to online analysis tools. Its output is a set of information regarding: infection type, C&C server IP address, and time of the analysis;
- **Whois** - This element retrieves the ASN associated to an IP address. It is used to discover in which ISP the C&C server is currently operating. The information is retrieved from a Whois Web page, which receives just the IP address as input parameter.

Figure 6 presents the mashup design for botmaster detection. In the first step, possible malicious binaries that infected a honeypot are submitted to the mashup, which sends them to three online binary analysis tools. CWSandbox and Norman execute the binary and analyze its behavior, while Virus Total scans the binary looking for virus signatures. The information gathered from these three tools is then correlated. The purpose of the correlation is to discover whether a C&C channel exists, and if so, to identify the associated C&C server.

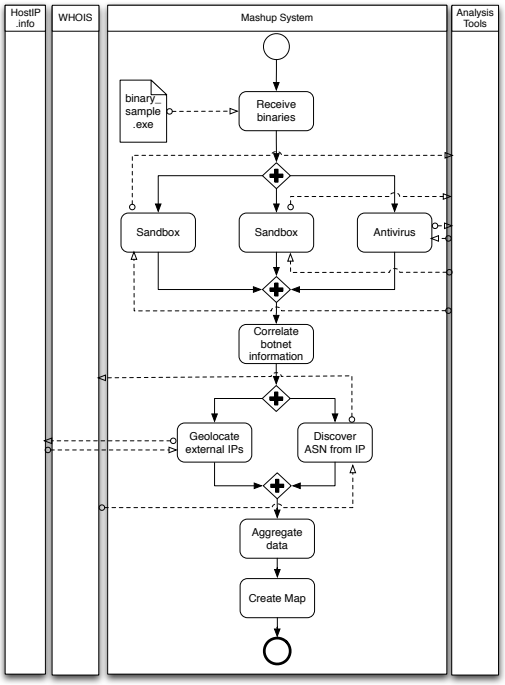

Fig. 6. Design of Botmaster Detection Mashup

With the C&C information, we use the **Geolocation** and **Whois** elements to discover both the location of the C&C server and its responsible ISP. This information is then aggregated and sent to the **Map** element, which generates an interactive map-based Web page. This Web-page allows the visualization of C&C servers spread around the globe through markers representing incidences of binary vectors of the botnet, as shown in Figure 5. In addition, the information gathered is also made available in a data feed of detected botnet vectors and C&C servers using different data formats (*i.e.* XML and JSON). Finally, the information is also time-stamped, allowing the creation of temporal representations of infections.

The composition of results from a collection of sandbox and anti-virus online tools is an alternative to the identification and mitigation of botnets based on a centralized approach. It replaces the use of complex and proprietary tools with tools that are free and simpler to use. The adaptability obtained with such a mashup-based approach is important because of

the heterogeneity of technologies employed in C&C channels. Since each sandbox tool is able to analyze and detect a specific set of technologies, the use of multiple tools allows the mashup to detect different types of infections. Furthermore, it allows constant self-updates through the addition of new tools to detect new infections. New analysis mechanisms are easily supported through the creation of new wrappers.

Finally, mashups are built with reusability in mind. This feature allows mashup compositions to be used as building blocks for more sophisticated compositions. For instance, it is possible to develop a mashup able to identify P2P botnets. Further possibilities include the detection of infected hosts inside a network through flow analysis, and the mitigation of botnets through the creation of firewall filters to automatically isolate the C&C channel.

VII. Conclusions and Future Directions

Currently, network administrators have to deal with an increasingly large amount of heterogeneous network information. Managing this information requires new and more sophisticated tools able to operate in both current and future networks. In this paper, we have introduced the use of mashups as a composition approach for network management. We have introduced a mashup system to compose new network management applications.

Through mashups, network administrators can create their own specialized management tools. Such mashups can also be reused by different administrators to build more sophisticated applications in a cooperative fashion. The development of mashups occurs through mashup systems, which employ high-level abstractions and usability-oriented interfaces to hide technical details from administrators and allow them to integrate management information. It is important to observe that mashup-based management does not replace traditional management models and tools; rather it allows traditional technologies to be more easily used to create new applications, pushing such technologies beyond their original purposes.

A mashup-based network management architecture and a mashup system prototype have been presented. With them, it was possible to rapidly develop mashups for two context specific management scenarios. The process required integrating Web-based management tools (*e.g.*, MRTG) with tools not originally conceived for network management (*e.g.*, geolocation, mapping). Some of the challenges observed include handling multiple protocols and data formats, definition and implementation of both high-level semantic and usability-oriented interface, and the support of cooperative content creation.

Our work is intended to be an initial step toward the introduction of mashups in the network management area. As Web 2.0 develops, there will likely be other interesting research opportunities in the context of network management. For example, semantic Web technologies can be used to improve the process of mashup development. Based on requirements defined by administrators, and on meta-information provided by the services, a mashup system could discover the most appropriate services to be employed in a given scenario. Mashups can also improve the management of large and geographically distributed networks [9] . Network administrators with complementary roles in different domains can team up and cooperatively create mashups for specific management tasks. Finally, mashups can also provide network administrators a means to leverage future internet services while keeping track of the traditional concerns such as devices and protocols [10].

References

[1] J. Yu, B. Benatallah, F. Casati, and F. Daniel, "Understanding Mashup Development," *IEEE Internet Computing*, vol. 12, no. 5, pp. 44–52, 2008.

[2] S. Murugesan, "Understanding Web 2.0," *IT Professional*, vol. 9, no. 4, pp. 34 –41, Jul.-Aug. 2007.

[3] D. Fensel, J. A. Hendler, H. Lieberman, and W. Wahlster, Eds., *Spinning the Semantic Web*, new ed. Cambridge, MA: MIT Press, 2005.

[4] R. S. Bezerra, C. R. P. Santos, L. M. Bertholdo, L. Z. Granville, and L. R. Tarouco, "On the Feasibility of Web 2.0 Technologies for Network Management: A Mashup-Based Approach," in *12th IEEE/IFIP Network Operations and Management Symposium*, Apr. 2010, pp. 487–494.

[5] G. D. Battista, T. Refice, and M. Rimondini, "How to extract BGP peering information from the internet routing registry," in *SIGCOMM Workshop on Mining Network Data (MineNet)*. New York, NY, USA: ACM, 2006, pp. 317–322.

[6] V. Cridlig, H. J. Abdelnur, R. State, and O. Festor, "Xbgp-man: an xml management architecture for bgp," *Int. Journal of Network Management*.

[7] F. Kamoun, "Toward best maintenance practices in communications network management," *International Journal of Network Management*, vol. 15, no. 5, pp. 321–334, 2005.

[8] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda, "Automatically Generating Models for Botnet Detection," in *14th European Conference on Research in Computer Security*, 2009, pp. 232–249.

[9] J.-P. Martin-Flatin, S. Znaty, and J.-P. Hubaux, "A survey of distributed enterprise network andsystems management paradigms," *J. Netw. Syst. Manage.*, vol. 7, no. 1, pp. 9–26, 1999.

[10] J. Schonwalder, M. Fouquet, G. Rodosek, and I. Hochstatter, "Future Internet = content + services + management," *IEEE Communications Magazine*, vol. 47, no. 7, pp. 27–33, 2009.

Carlos Raniery Paula dos Santos (crpsantos@inf.ufrgs.br) is a Ph.D. candidate in Computer Science at the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS), Brazil where he also hold an M.Sc. degree in Computer Science (2008). In 2005, he received degree in Telematics from the Federal Center of Technological Education of Ceará (CEFET-CE). His topics of interest include network management, Web services-based management, P2P-based systems, and Web 2.0/3.0 technologies.

Rafael Santos Bezerra (rsbezerra@inf.ufrgs.br) received a B.Sc. (2008) in Computer Science from the Federal University of Piaui. He is a M.Sc. candidate in Computer Science at the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS), Brazil. His topics of interest include network and service management, Web 2.0, Web-oriented architectures, software engineering, software testing, and systems security.

João Marcelo Ceron (jmceron@inf.ufrgs.br) holds a B.Sc. degree (2007) in Computer Science from UNISINOS and M.Sc. (2010) from the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS). He is SSP-CNSA certificate by Sans Institute. Currently, works in the RNP point of presence in the Rio Grande do Sul state, and is member of the CERT-RS (Computer Emergency Response Team - Rio Grande do Sul), responsible for the state academic network. His topics of interest are network management and network security.

Lisandro Zambenedetti Granville (granville@inf.ufrgs.br) is an associate professor at the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS), Brazil. He received his M.Sc. and Ph.D. degrees, both in Computer Science, from UFRGS in 1998 and 2001, respectively. From September 2007 to August 2008 he was a visiting researcher at the University of Twente, The Netherlands, with the Design and Analysis of Communication Systems group. He has served as a TPC Co-Chair for IFIP/IEEE DSOM 2007, IFIP/IEEE NOMS 2010, and TPC Vice-Chair for CNSM 2010. He is also Technical Program Chair of IEEE Communications Society Committee on Network Operations and Management (CNOM). His areas of interest include policy-based network management, P2P-based services and applications, and management of virtualization for the Future Internet.

Liane Margarida Rockenbach Tarouco (liane@penta.ufrgs.br) is a full professor at the Center for Innovation and Use of New Technologies in Education of the Federal University of Rio Grande do Sul (UFRGS), Brazil. She holds a M.Sc. degree (1976) from UFRGS and a Ph.D. degree (1990) from the University of So Paulo (USP), both in Computer Science. She chaired the 7th Brazilian Symposium on Computer Networks (SBC/LARC SBRC 1989) and has been serving as a TPC member since 1983. From 1983 to 1992 she was a representative of Brazil at IFIP TC6 (Communication Systems). Her areas of interest include distributed network management and expert systems to support fault correlation.

A Data Confidentiality Architecture for Developing Management Mashups

Carlos R. P. dos Santos and Rafael S. Bezerra
and Lisandro Z. Granville and Leandro M. Bertholdo
Institute of Informatics - UFRGS - Porto Alegre, RS, Brazil
Email: {crpsantos, rsbezerra, berthold, granville}@inf.ufrgs.br

Winnie Cheng and Nikos Anerousis
IBM Watson Research
Hawthorne, NY 10532, USA
Email: {wcheng, nikos}@us.ibm.com

***Abstract*—Mashups are powerful applications created from accessing and composing multiple and distributed information sources. Their ease-of-use and modularity allow users at any skill level to construct, share and integrate their own applications. However, data security concerns remain a hindering factor in its widespread adoption, in particular, for network management. In this paper, we propose a novel development methodology and system architecture called *Maestro* that allows developers to express their data privacy concerns and enforce policies during mashup executions. We evaluated Maestro by building two mashup applications for managing live networks and by running performance tests that show that our runtime has negligible overhead.**

I. Introduction

In recent years we have observed a quick proliferation of a new class of Internet applications better known as "mashups" or "situational applications". A mashup can be characterized as the product of combining two or more complimentary web-based applications and/or data sources. The increasing availability of APIs for popular web applications and services makes it possible now for even the most inexperienced users to compose mashups to solve some very specific problems (hence the term "situational"). It is very common nowadays to encounter applications that overlay data objects on top of a mapping service and thereby providing an enhanced view of the data. It was not long before systems and network management practitioners took advantage of the mashup paradigm to enable a new class of management applications on top of existing repositories of management information.

In our previous work [1], we conducted a set of experiments and analysis of the mashups approach and we concluded that mashups are a promising and appropriate tool for monitoring, diagnosing and managing networks. They provide an excellent visualization medium for composed information, collecting and aggregating data from multiple network monitoring and performance tools. Furthermore, the modular nature of mashups encourages re-use and sharing of modules between different service providers and network administration staff, which in turn, allows composing even more sophisticated applications.

As the spectrum of possible applications widens, a number of challenges intrinsic to mashups become apparent. Data security in particular proves to be extremely challenging in an open mashup environment, and is the main topic of study in this paper. Especially in network management applications, mashups may be required to handle and display sensitive data that require well-defined disclosure policies. For example, in a common application that shares traffic information between Internet service providers, a certain provider may engage in proprietary peering relationships with competing providers and the inadvertent publishing of such detailed information through a mashup may compromise their market position. On the other hand, the disclosure of *aggregate* traffic can be very useful to all parties to monitor quality of service levels, as long as the details are not being disclosed. Such examples motivated us to study further the requirements for data protection in shared mashup environments; particularly what mechanisms must be put in place to satisfy the confidentiality concerns of different parties.

Protecting sensitive data in a highly collaborative and dynamic mashup environment poses several non-trivial questions. Our research indicates that the solution should be centered on an enhanced composition programming model that specifically captures and enforces disclosure policies. In this way, data protection mechanisms are built directly into these applications. The model should be easy to understand, sufficiently expressive, and able to represent broad categories of security concerns among a wide range of mashup applications.

To satisfy the above-mentioned requirements we adopt the concept of Information Flow Control (IFC) in our architecture. IFC is a security model that tracks the flow of sensitive data, enables the specification of disclosure policies and enforces them. IFC is a promising direction in enhancing the security of mashups, particularly in the context where data is interchanged with other mashup modules. This paper introduces a new mashup development system, named *Maestro*, that extends a state-of-the-art IFC programming model, Aeolus [2], with a highly modular mashup composition architecture. The key contributions of our research work are to:

1) Propose a novel secure mashup development environment;
2) Show a proof-of-concept prototype with negligible overhead and little compromise on usability;
3) Demonstrate realistic scenarios for privacy-preserving network management.

We designed and implemented a secure mashup develop-

ment and deployment platform that enables mashup developers to respect various data confidentiality concerns during the composition process. We created two management mashups for administrating Autonomous Systems (AS): one focuses on network quality management and the other on network traffic monitoring. These scenarios are derived from interviewing real-world experts, particularly network administrators, and from our understanding of their concerns. The network quality management mashup aggregates information such as network traffic, number of announced routes, input and output queue sizes, number of discarded packets, and CRC errors from BGP-enabled routers connecting AS peers. Such management capability is important because it allows network administrators in each AS to verify whether their service providers have violated Service Level Agreements (SLAs), requiring the enforcement of financial provisions in the contract. The second mashup is designed for network traffic monitoring providing different views of the backbone network traffic with different access levels of the information for each group.

The remainder of this paper is organized as follows. In Section 2, we provide a background of our research. In Section 3, we introduce our architecture, Maestro, which supports strong confidentiality mechanisms for mashup development. In Section 4, we describe two network management scenarios where the enforcement of such mechanisms is necessary. In Section 5, we study two implementations of Maestro for these scenarios by creating the corresponding mashup applications and evaluating their implementation. Section 6 discusses related work. The paper concludes in Section 7.

II. Background

A. Web 2.0 and Mashups

In recent years, web applications have taken on renewed interested both in industry and academia, termed as Web 2.0. These trends propose web applications that focus on the end-users and more importantly, encourage them to be organizers of content in addition to being consumers of information. As a result, there is an increasing demand for principles that support re-use of content, collaborative sharing and compilation of content among users [3]. Mashups have emerged as a class of web application that composes content from a variety of web resources such as web pages, RSS feeds, web services and online APIs [4].

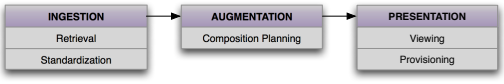

Fig. 1. Mashup Development Methodology

Typically, mashups are built following the methodology proposed by Jhingran *et. al* [5], presented in Figure 1. There are three main steps: ingestion, augmentation, and presentation. The ingestion step gathers data from disparate sources, such as RSS Feeds, Web Services, and online APIs. Since these sources tend to be heterogeneous, wrappers are often used to standardize and retrieve the interested data. During the augmentation step, the user composes the data by defining modules that operate over the data, transform it, and generate more meaningful information from the aggregate. The result of this process is presented to the mashup creator and viewers in the presentation step. The presentation can range from a simple text file containing the results of the composition, to a complex and highly interactive map display.

Mashups can also be built in an *ad-hoc* manner, using standard programming tools and integrated development environments (IDE) to develop one specific composition. This approach relies heavily on the skill of web developers. Mashup systems make the creation of mashups accessible to a larger community, with no programming expertise needed to create custom mashup applications. Such systems are environments for mashup creation, similar to an IDE for traditional development, adhering to the development methodology described earlier. Mashups systems are also runtime platforms for executing and storing created compositions.

In the context of the network management area, we conducted a set of experiments and analysis in our previous work [1] that has shown mashups as a promising and an appropriate tool for monitoring, diagnosing and managing networks. This work also provides depth details of the mashup system used and extended in this paper. Compared to the current network management solutions, easier and faster development are the main advantages of solutions based on mashups, allowing the creation of larger applications, and reducing development costs and time.

B. Data Protection

Mashups enable new capabilities in data visualization. However, the ability to protect such data is essential. While data protection techniques have been proposed since the 1970s [6] [7], the majority of the work in this area centered on providing data protection for static data such as databases and files [8]. Mashups present a very different operating environment, in which data may be generated dynamically as the output of one mashup module and fed through a variety of intermediate ones before the final result is displayed to the end-user.

Conventional techniques for ensuring the confidentiality of data can be broadly classified into two bodies of work: access control and cryptographic key management. Access control provides frameworks for identifying the users of a system and seeks to restrict the access different users have on the data. For example, role-based access control [9] allows the specification of a task-based role such as "Accounting Department Personnel", the types of data (*e.g.*, revenue and expense report files) that is accessible to users of this role, and the operations (e.g., read, and/or write) that can be done on the data. Access control protects data in storage rather than in flight, the latter is common to mashups. It deals poorly with data that are output of computations and also with data that have been transformed, in many cases iteratively by several mashup modules. Cryptographic key management

[10] uses encryption algorithms to hide data and relies on the distribution of the encryption/decryption keys to restricted users of the system to protect the confidentiality of the data. It can better hide in-flight data than access control techniques, at the expense of significantly complicating the key generation and distribution schemes as it has to cover all possible data transfer points between mashups. Encryption protocols must be agreed upon between two interacting mashups and new keys must be generated whenever sharing relationships change. The key strength of encryption is in ensuring confidentiality of data content during transfers rather than in expressing confidentiality policies between myriad of network endpoints.

An area of security research that has recently regained interests is information flow control [11]. Information flow control focuses on controlling the propagation of sensitive data. It does this by labeling the data and by providing rules on how labeled data can travel through a system. Different information flow control techniques have been proposed and applied to a variety of settings: at the programming language level, at distributed application programming model, and for operating system processes monitoring. In this work, we extend the Aeolus programming model in [2] for mashup development. Comparisons with other IFC works were also done in [2]. Compared with other IFC models, Aeolus is easy-to-use with a simple set of rules yet retaining expressiveness of confidentiality policies. Compared with other IFC implementations, Aeolus is particularly suited to describing functions and outputs of computations.

III. Maestro

The mashup development methodology described in Section II-A, and the Network Management Mashup System (NMMS) [1] provide a good basis for mashup developments but does not address data confidentiality concerns. In particular, the augmentation step must be modified to capture the flow of sensitive data. Such problem is not specific to a particular architecture like the NMMS, but it is a gap in mashup-related research and is an opportunity of innovation. In this section, we describe our secure mashup development system, Maestro, which bridges this gap. Maestro demonstrates how an IFC model like Aeolus can be applied to a composition framework such as NMMS.

A. Secure Development Methodology and System Architecture

The methodology for Maestro, like Aeolus, centers around three key concepts: principals, tags, and labels. Principals represent entities with an interest in security, such as individuals or network providers. Tags provide a way for principals to categorize their information. Labels are sets of tags and are used to enforce information flow. We propose the use of one type of label, the Secrecy Label (L_s), for expressing the confidentiality concern of the information in mashups. When data flows from one module (A) to another one (B), it must satisfy the secrecy rule presented below.

- Secrecy Rule: $A.L_s \subseteq B.L_s$

The secrecy rule says that the destination's secrecy label ($B.L_s$) must be a superset of the source's secrecy label ($A.L_s$) to ensure that the sensitivity of all sources contributing to the destination is captured by the destination's secrecy label.

To support our secrecy mechanism, we designed the components highlighted in Figure 2. These components are responsible for tracking the propagation of data and preventing accidental sensitive data leakage. They do the former by enforcing the secrecy rule and the latter by disallowing any data with non-empty secrecy label to leave the system. For example, a text field that has the *Financial* tag in its secrecy label cannot be published on a public web portal. The Secrecy component provides labeling support and propagation of data in accordance with the secrecy rule. The User Authentication component in Figure 2, besides providing user login validation, uses the user credentials to relate it to a principal. The runtime engine in the Composition Runner, runs the mashup as a particular principal that has very specific authority on what sensitive data it can disclose.

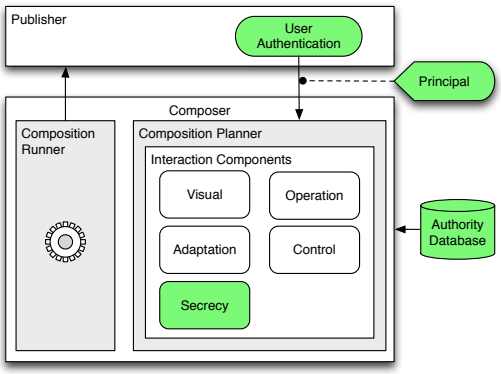

Fig. 2. Maestro Architecture

Since our mashup runtime engine prevents any non-empty secrecy label, the security of the model reduces to when tags can be removed from the secrecy label. Hence, removing a tag from the secrecy label is considered a privileged operation that requires authority. Authority is defined with respect to a tag and is given to principals. When a principal creates a tag, it has authority for that tag and therefore, can permit the removal of that tag. A principal can also delegate this authority to another principal or to a specific snippet of code. We refer to the latter case as an authority closure since the principal can tightly couple authority with how it will be used. For example, the principal *NetworkProvider X* which has authority for the tag *NetworkInfo X* may grant an authority closure *ComputeAggregateTrafficStatistics* the authority for this tag since the computation outputs only summarized statistics and does not disclose specific link traffic information that are deemed sensitive by *NetworkProvider X*.

Maestro has an *Authority Database* that maintains information on what tags different principals have authority for and metadata on authority closures. It also contains mappings of user credentials to principals so that we can execute with the intended principal at the User Authentication phase. When a mashup is executed, the *Authority Database* is consulted when a principal requests a removal of a tag from the secrecy label of the mashup module and our system validates that such an authority exists.

B. Development Environment

The original NMMS provides interaction elements that a mashup developer can customize to construct a new mashup application. These interaction elements include adaptation, operation and control modules. Adaptation modules provide interfaces for information retrieval from customized data sources. Operation modules implement data transformation logic such as data merge operations. Control modules represent basic programming logic, such as loops and conditions. These elements are able to read and modify the data and hence, can divert potentially sensitive data to unintended modules or recipients.

In the Maestro architecture, we have created a new type of interaction elements, called **Secrecy** modules, to provide developers a way to identify sensitive data as well as specify when they can be disclosed. This new type of element provides new features to the augmentation step for controlling information flow.

There are two types of secrecy interaction elements: *Add Tag* module and *Remove Tag* module. The *Add Tag* module allows developers to specify the ingress of sensitive data into a mashup application, by adding a developer-defined tag to the secrecy label of the module. The Maestro runtime engine checks that only modules with non-empty secrecy label can display data, in this way, providing control on data disclosure. The *Remove Tag* module allows developers to specify when a tag can be removed from the secrecy label and hence, essentially, declassifying the data and permitting its disclosure. Unlike *Add Tag*, *Remove Tag* requires authority and our runtime is augmented to consult the authority database before proceeding with the removal. The *Remove Tag* module can have specific code associated with it, implementing an *authority closure*. An *authority closure* is authoring bound to specific code. For example, a *Remove Tag* module may have code that anonymizes server names and IP address before returning ingress/egress network queue sizes.

To support the secrecy modules, the message format exchanged between the modules in the composition flow includes a label parameter composed of a set of tags. When the mashup creator defines no tags, the Maestro runtime engine assumes an empty label and continues to track the propagation of sensitive data by conforming to the secrecy rule.

C. Example of Mashup Composition with Data Secrecy Protection

We provide an example illustrating how the secrecy interaction elements are used in the creation of a mashup application. In Figure 3, secrecy modules are added to protect the data gathered and transformed in a mashup application. In Step (1), two adaptation elements are instantiated to retrieve external information from data sources. Depending on the data source, the *Add Tag* secrecy modules are used in Step (2) to appropriately tag the sensitivity of the data entering the mashup application. For example, one data source may be tagged ”A” and the other ”B”. All these modules are made available to the creator through a Development Environment similar to the one originally developed in NMMS, and are included as visual toolboxes that can be dragged-and-dropped into the composition flowchart.

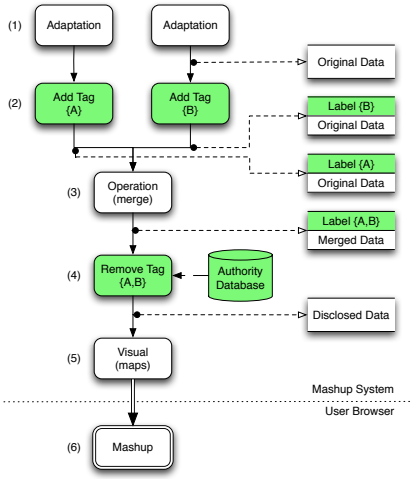

Fig. 3. Secrecy Usage Example

At runtime, the labels are propagated from module to module. For example, the operation module in Step (3) will have a secrecy label that includes all the tags from Step (2). The result of this operation module cannot be passed to the presentation tier since it has a non-empty secrecy label and hence is protected from accidental and unintended leakage of private information. To allow data that has been tagged sensitive to display, the creator must also specify the *Remove Tag* module (Step (4)) that permits the removal of the tag from the secrecy label by defining the necessary redaction or anonymization procedures before mashup application results can be displayed through visual elements to the users (Step (5) and Step (6)). The Maestro runtime engine will ensure that such code is executed and that the *Remove Tag* module runs with a principal that has authority to remove the specified tag.

IV. Applications

In this section, we describe two use cases to showcase Maestro: network quality management and network traffic monitoring. The Internet is formed by an interconnection of numerous independent administrative domains called Autonomous Systems (ASes), most of them owned by Internet Service Providers (ISPs). As the name implies, these systems are autonomous because their network administrators define policies (*e.g.*, traffic shaping and traffic priorities) that are independent of those from the other domains. When two ASes establish a connection, one announces a set of IP address prefixes to the other and vice-versa using the Border Gateway Protocol (BGP). The participants of this connection are called peers and this scenario is called BGP peering.

A. Network Quality Management

Proper management of BGP peering is very important to network providers. Burstiness in Internet traffic and unexpected network behavior can cost providers millions of dollars. Abrupt changes in the amount of IP prefixes being announced and sudden drop in network traffic can happen due to misconfigurations or malfunctioning of network devices resulting in Service Level Agreement (SLA) violations. Such behavior should be detected and reported as soon as possible. Also, Quality of Service (QoS) parameters need to be monitored by both sides of a connection to help the network administrators diagnose problems in the network. For example, in Figure 4, UDP traffic is exchanged between the ASes. If an error at AS 1 causes UDP packets to be discarded, AS 2 and AS 3 will not be aware of the problem. AS 1 must detect such problems and notify the peers.

Our network quality management mashup application provides information on QoS related parameters such as queue size, discard rate, CRC errors, and various connection information (*i.e.*, number of announced routes and amount of traffic exchanged), to help peers of a connection detect unexpected behaviour and act accordingly. For example, if a network administrator in AS 2 is informed that its international traffic via AS 1 is suddenly being discarded at an abnormally high rate, this may suggest a botnet infection and AS 2 might want to quarantine traffic emanating from AS 1 by blocking all the input traffic from this peer.

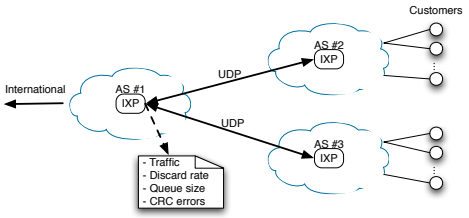

Fig. 4. Network Quality Management Application

Making all connection information available to all peers of a network will allow network administrators in each AS to diagnose network quality problems. However, such transparent and undifferentiated sharing of information undermines competitive advantages of different AS administrators. In our example, AS 1 provide services to both AS 2 and AS 3, but these two ASes may be competitors in providing local Internet access; therefore, they should not be able to see each other's network quality statistics with AS 1. Network quality indicators are very sensitive information and should be made visible exclusively to the peers participating in a specific connection. Hence, it is necessary to have a secrecy mechanism to help mashup creators express such concerns and enforce them.

B. Network Traffic Monitoring

The Network Traffic Monitoring application is an essential network management tool that provides real-time as well as historical network traffic information to many users. Different types of users have different interests in the data and need-to-know. Secrecy mechanisms are needed to disclose the appropriate level of traffic information to each user group.

A large network service provider responsible for an AS may have multiple Internet eXchange Points (IXPs) that provide services across a country or a region of the world. Each IXP may be internally managed by a sub-contractor or a standalone sub-department of the larger organization. The users involved are:

- **Chief Operations Officer (COO)**: responsible for making decisions on how and where capacity improvements should be made;
- **IXP administrator**: responsible for managing the network traffic within his/her own IXP;
- **Customers**: peers with whom an AS exchange traffic;
- **General users**: other non-privileged users interested in seeing traffic information from the AS.

The types of data that our Network Traffic Monitoring application provides can be broadly categorized into four classes:

- **IXP detailed traffic info ($\mathbf{IXP}_d$)**: traffic generated to/from a specific peer (*i.e.*, customers) inside each IXP;
- **IXP traffic ($\mathbf{IXP}_s$)**: aggregated traffic generated to/from all peers with a connection to a specific IXP;
- **AS detailed traffic ($\mathbf{AS}_d$)**: traffic generated to/from a specific peer over all IXPs;
- **AS traffic ($\mathbf{AS}_s$)**: aggregated traffic generated by the entire AS (*i.e.*, sum of all IXP traffics).

Figure 5 shows the types of users that may be involved and the appropriate level of traffic details that should be exposed to them:

Based on the different categorizations of the network traffic, we depict the correct access levels for the different user types in Table I:

The COO should be able to see any information about his/her network, including both detailed and summarized network traffic. With this information, he/she can understand

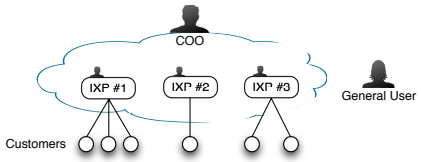

Fig. 5. Network Traffic Monitoring Application

TABLE I
INFORMATION DISCLOSURE LEVELS

	AS_a	AS_d	IXP_s	IXP_d
General Users	X			
Customers	X	X		
IXP admins	X	X	X	
COO	X	X	X	X

the usage pattern and capacity constraints of the network backbone, and invest in additional resources if necessary. The network administrator should be able to see detailed traffic information within his/her own IXP and not the details of other IXPs. This respects the principle-of-least-privilege and minimizes probability of sensitive information leak. The customers (*i.e.*, peers) need to visualize information regarding all traffic generated by each IXP. This information helps them to decide with which IXP to maintain a peering relation. Detailed extended AS and IXP traffic information should be hidden to the customers. Finally, the total traffic generated/received by the AS can be displayed to any user. This is usually done to show the growth of the backbone to the general public as a marketing mechanism.

V. CREATED MASHUPS AND EVALUATION

Based on the presented network management scenarios, we developed two mashups using Maestro to enable network administrators to manage their networks while respecting their privacy concerns. Both mashups display aggregated network information on a map-based web page. Network maps are a classical tool for visualizing network topologies and the health of different regions of the network, and especially for understanding the complex networks maintained by AS service providers.

A. Network Quality Management Mashup

To create the first mashup, we generated three different types of adaptation modules to capture the different data sources for each network interface: *SNMP*, *geolocation*, and *image extraction*. Each *SNMP* module retrieves from a local BGP enabled router, the IPs of local BGP peers using the BGP-4 MIB module, the IPs of external BGP peers, for the network interface on which it is connected. For each network interface, a SNMP request is done to retrieve the amount of traffic exchanged, discard rate, CRC errors, and queue size using the MIB-2 module. The Image Extraction module retrieves the appropriate network graphs generated by the Multi Router Traffic Grapher (MRTG) tool that accesses the local BGP router.

To present different views to different users running the created mashup, we added several Secrecy modules to the application. After retrieving data from the network interfaces of the managed BGP router, a *Split* operation module and several *Filter* operation modules separate the information into different flows depending on which partner is connected to each interface. For each of those flows, we used the *Add tag* secrecy module to add a specific tag to its secrecy label. Next a Merge component integrates all separated flows into one. At this point, depending on the runtime filter conditions, the information flow may be composed of a set of tags (*i.e.*, any subset of the above tags) in its secrecy labels. This logic is presented in Figure 6-A. Any data manipulations downstream will contain these tags and the Maestro runtime will prevent its disclosure. In this way it protects against composition errors and logical errors in other modules from mistakenly publishing private information.

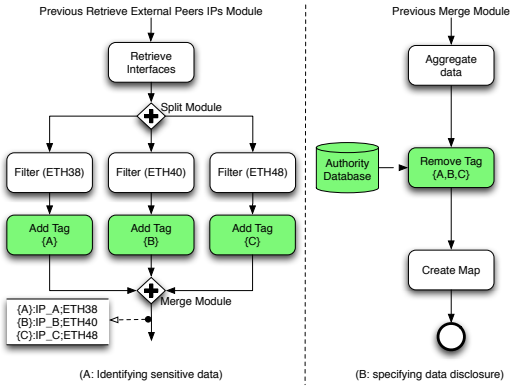

Fig. 6. Composition Logic

In the last step, before reporting the mashup output to a visual component (the *Map*), we used the Secrecy component, *Remove tag*, to specify the tags to be removed depending on the user executing the mashup. This is shown in Figure 6-B. At runtime, the Authority Database is consulted to see whether the running principal has authority to perform the tag removal. The mashup output can only be displayed if all tags are removed from the secrecy label.

Figure 7 presents the resulting map generated from this mashup. Peers are represented as markers and the connections as lines linking those markers. Clicking on a visual element opens up a window where the user can access the pertinent information of the selected element, such as ASes involved in a connection, total traffic, CRC errors, packet discard rate, and announced routes. The user can also control the zoom level of the map according to his/her needs.

Referring back to the scenario description in Figure 4,

Fig. 7. Network Quality Management Mashup

for a mashup displaying AS1's network quality information, users from competing AS2 and AS3 will only see information for their shared links with AS1 (*i.e.*, users from AS2 will not see link utilization data between AS1 and AS3). This is enforceable since each group will run with a different principal that only has authority for removing tags associated with their link data.

Since we maintain the modularity of the composition methodology, adding secrecy modules to the composition flow is easy-to-grasp and retains the easy-of-use benefit of mashup systems.

B. Network Traffic Monitoring Mashup

The second mashup is devoted to present different granularity of traffic information to different types of users. We used four *Add tag* modules in different steps of the composition, each related to one user access level. The first module retrieves information from each IXP. This module adds two tags: $IXP_{detailed}$ and $AS_{detailed}$. The second *Add tag* module adds the tag IXP_{sum} after the module that sums the amount of traffic exchanged inside one specific IXP. The final module sums all traffic of the backbone and adds the tag AS_{sum} .

Figure 8 shows the generated maps for the 3 different user groups. The green marks represent the IXPs, and the connections are represented as dark lines linking those markers. The blue marks represent peers and the blue lines represent their connections to the AS. Clicking over a green mark shows the sum of all traffic exchanged over a specific IXP, and clicking on a blue line shows only the specific traffic of this connection. Figure 8-A is the view of the COO, where he/she can see all IXPs of his/her backbone, and their connection with external peers. Figure 8-B, presents the view of the IXP administrator, where he/she can see only the specific traffic of peers connected directly to his/her own IXPs and also the aggregated traffic of the other IXP of the backbone. The last Figure 8-C presents the view of the Customer, where he/she is able to see the traffic of other IXPs of the AS provider and only the amount of traffic his/her AS have generated. The last view regarding the general user is not presented in the paper due to space constraints, but it presents only information containing the amount of input/output traffic of the entire AS.

C. Quantitative Evaluation

Maestro provides important security features for the development of mashups. These features require additions to the tool chain and can impose overhead on the overall mashup execution. Therefore, we ran a set of performance tests to measure the overhead of our security features. Our evaluations were carried on a Tomcat 7.0.0 application server running on a machine with 2.13GHz Intel Core 2 Duo processor, 6MB cache, and 2 GB of RAM.

Maestro provides two types of secrecy modules: *Add Tag* and *Remove Tag*. These include additional tag and label manipulations and tracking on top of the basic functionality of a module. We measured the message propagation delay times for an original basic module, an *Add Tag* module and a *Remove Tag* module. We took 30 measurements for each module type. In Table II, we show the 95% confidence level on the observed average message delay time (in milliseconds) and the standard deviation σ.

TABLE II
OVERHEAD RESULTS

	Average (ms)	σ
Original	10	2.63
Add tag	13	2.80
Remove tag	15	3.40

The overhead is insignificant if compared to other external time-consuming factors, such as processing time of router and the network round trip time of these requests, which are often in the orders of hundreds-of-milliseconds to seconds. Furthermore, a mashup application typically consists of many modules and only a small portion of these will be *Add tag*/*Remove tag*. For instance, to create the first mashup presented in this paper, we used 10 adaptation modules to access the external information, plus 14 modules to implement operations such as filter and merge. Only three *Add tag* modules and one *Remove tag* were used. All these modules are executed when a mashup application runs and hence, we estimate that the overall overhead of the application is less than 0.05%.

VI. RELATED WORK

Across the web, there are many mashup efforts. ProgrammableWeb.com, for example, as of August 2010 listed more than 5000 different mashups. Most of them, however, are not created using a development platform but rather through ad-hoc and custom HTML/JavaScript coding. Yahoo! Pipes is an example of a successful mashup system with thousands of user-created mashups. It provides a user-friendly drag-and-drop interface, dynamic web resource access, adapters for popular data formats such as RSS and other popular formats. Intel MashMaker is a mashup system that aims at extracting structured data exclusively from Web pages. Their design is based on extractors interfacing with web pages presented as visual widgets and on allowing users to link their data to other widgets.

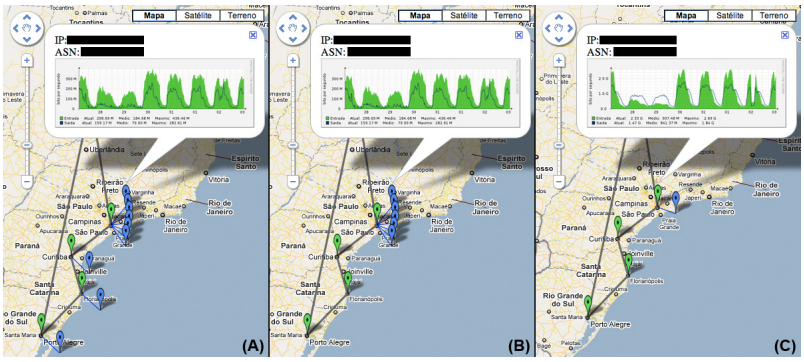

Fig. 8. Network Traffic Monitoring Mashup

In academia, one of the first investigations on mashups was carried out by Banerjee *et al*. [12]. The authors discussed the usage of mashups in the development of telecommunication applications. Their proposed architecture, however, is tailored for the development of a very specific mashup (*i.e*., call-a-cab application) and does not allow an end user to build his/her own mashups.

The work presented by Andreas Thor *et al*. [13] proposed the integration of heterogeneous data through wrappers designed to retrieve common information from various data sources. For example, the authors presented a mashup that correlates data from DBLP and Google Scholar. The employed wrappers retrieve only common information from both sources, such as author's names, publication title, and year of publication. To specify the composition, the authors proposed a script language to query and compose the common information. While these related projects have further the programmability of mashups, to our knowledge, Maestro is the first to incorporate confidentiality constraints in mashup development.

VII. Conclusions & Future Work

In this paper, we introduced a new architecture for secure mashups development, called Maestro. This architecture uses information flow control as a confidentiality enforcement mechanism and empowers mashup creators with the important ability to specify well-defined disclosure policies in handling sensitive data. We also presented two realistic network management scenarios where multiple information sources are aggregated to create new applications for specific purposes: quality management and traffic monitoring.

These scenarios have specific constraints on data disclosure and Maestro was able to capture and enforce these constraints, validating the value of our secure mashup development methodology. In addition, the usage of visual interaction elements during the augmentation step provides an easy-to-use development environment with little compromise on usability. The new visual elements, *Add tag* and *Remove tag* allow developers to control information flow, for example, in providing different views of the data to different users. Through a quantitative evaluation, we observed a negligible overhead in mashup application execution time using secrecy modules. As future research, we plan to extend Maestro with tools to allow visualization of sensitive information flows and reasoning of data validity as information are composed.

References

[1] R. Santos Bezerra, C. Raniery Paula dos Santos, L. Marcio Bertholdo, L. Z. Granville, and L. R. Tarouco, "On the feasibility of web 2.0 technologies for network management: A mashup-based approach," in *NOMS'2010: Proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium*, apr. 2010, pp. 487 –494.

[2] W. W.-Y. Cheng, "Information flow for secure distributed applications," Ph.D., MIT, Cambridge, MA, USA, Aug. 2009, also as Technical Report MIT-CSAIL-TR-2009-040.

[3] T. O'Reilly, "What is web 2.0," 2005, http://oreilly.com/web2/archive/what-is-web-20.html.

[4] D. Merril, "Mashups: The new breed of web app - an introduction to mashups." 2006, http://www.ibm.com/developerworks/web/library/x-mashups.html.

[5] A. Jhingran, "Enterprise information mashups: integrating information, simply," in *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment, 2006, pp. 3–4.

[6] D. E. Bell and L. J. LaPadula, "Secure computer systems: Mathematical foundations," MITRE Corporation, Tech. Rep., March 1973.

[7] K. J. Biba, "Integrity considerations for secure computer systems," MITRE Corp., Tech. Rep., 04 1977.

[8] R. Bhatti, D. Gao, and W.-S. Li, "Enabling policy-based access control in bi applications," *Data & Knowledge Engineering*, vol. 66, no. 2, pp. 199 – 222, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYX-4S62RG0-1/2/27510ec305e091bbe4cdf6a80209b1d8

[9] D. Ferraiolo and R. Kuhn, "Role-based access control," in *15th NIST-NCSC National Computer Security Conference*, 1992, pp. 554–563.

[10] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 309–329, 2003.

[11] A. C. Myers, "Jflow: Practical mostly-static information flow control," in *Proc. 26th ACM Symp. on Principles of Programming Languages (POPL)*, 1999, pp. 228–241.

[12] N. Banerjee, K. Dasgupta, and S. Mukherjea, "Providing middleware support for the control and co-ordination of telecom mashups," in *MNCNA*. ACM, 2007, p. 11. [Online]. Available: http://doi.acm.org/10.1145/1376878.1376889

[13] A. Thor, D. Aumueller, and E. Rahm, "Data integration support for mashups," *AAAI Workshop on Information Integration on the Web*, 2007.

Botnet Detection Using a Mashup-based Approach

Rafael Santos Bezerra, Carlos Raniery P. dos Santos, Joao Marcelo Ceron,
Lisandro Zambenedetti Granville, Liane M. R. Tarouco
Institute of Informatics - Federal University of Rio Grande do Sul, Brazil
Email: {rsbezerra, crpsantos, jmceron, granville}@inf.ufrgs.br, liane@penta.ufrgs.br

***Abstract*—Botnets are considered by specialists, in both industry and academia, as one of the greatest threats to security on the Internet. These networks are composed by a large number of malware-infected hosts acting under a central command. They are usually employed to perform DDoS attacks or phishing scams. The behaviour of these botnets evolves due the adoption of new and sophisticated infection methods, changing of network protocols, and the employment of different command and control mechanisms. The security community, thus, is always dealing with such constant change. However, most botnet detection methods address just specific infection types or C&C protocols. We, therefore, propose a new approach based on the dynamic integration of pre-existing tools to achieve a more efficiently detection solution. To such end, we base our approach on a novel Web 2.0 technology called mashups to perform the information correlation. The proposal is extensible enough to allow even non-security information such as online mapping APIs be integrated to create more sophisticated compositions, and displaying the results in more meaningful ways.**

I. Introduction

The proliferation of personal computers and the decreasing costs of communication led the growing of computer networks, specially the Internet. Unfortunately, this growing comes associated with the increasing number of vulnerable hosts to infections. In the past, such infections were caused by viruses and worms that aimed to compromise and disable their targets. However this scenario has changed, virus creators no longer aim to just acquire recognition and attention from their community or the media. Now their main interest is making profit of their creations. Most of the infections are no longer committed to cause non functioning on the victims. Instead, infected hosts become *bots*, which can be remotely controlled by a human operator known as *master* or *botmaster*. Sets of such hosts are called *botnets* and are usually used to illegal purposes, such as large scale Distributed Denial of Service (DDoS) attacks, email *spamming* and *phishing* [1].

Botnets are considered the current largest security threat on the Internet [2]. One of the main reasons for this is the high number of infected computers. According to Vincent Cert [3], up to 25% of all computers on the Internet belong to botnets. Another reason lies on their effectively. Combining the efforts of hundreds or even thousands of machines widespread across the Internet to reach a single task such as DDoSing a server or spamming makes this approach very effective and hard to prevent. Due to such widespread nature, dismantling these networks is also a complex task. However, botnets conceptually have one single point of failure that can potentially be exploited, the botmaster. Identifying the botnet controller opens up many possible countermeasures, such as blocking communication, removing infected machines from the botnet, or even infiltrating this controller host in order to identify all participants of his botnet, as proposed by [4].

Most approaches to detect botmasters are based on detecting communication mechanisms used between the botmaster and the bots, which are known as *command and control* (C&C) channels. Centralized botnets have widely employed the Internet Relay Chat (IRC) protocol for many years [5]. Some efforts to extract information from the *malware* via IRC or disable its communication can be found, such as [1], [4] and [6]. However, more sophisticated botnets started to use other channels for communication. For instance, HTTP [7] is used to provide more stealthiness by mixing the C&C messages with regular Web traffic, also crossing firewalls more easily. Storm worm [8] employs Peer-to-Peer (P2P) communication to distribute the control of the botnet. Such approaches raise the difficulty of detecting botmasters, and nowadays there are just a reduced number of works that claim to address all of them [9].

In this scenario, we believe that a new technology, called mashups, can be used to detect such dynamic botnets. Mashups are a recent Web 2.0 technology that proposes the agile and dynamic integration of external resources in a Web page, providing new functionalities from such integration [10]. In particular, we believe that mashups can be extended and rapidly adapted to detect new and evolving botnets through the integration of multiple tools (*e.g.*, anti-viruses, online sandboxes, traffic analysis tools). In this paper, we created a mashup that integrates information from disparate web available binary analysis tools such as sandboxes and anti-viruses [11] [12] [13]. We also employ other free web-based tools, such as geocoding services and map APIs to present the results obtained in a meaningful way. Our main goal with this work is not address fundamental questions related to botnet detection, but instead, we want propose a flexible solution for creating new applications focused in detecting dynamic botnets. We also want to investigate if the usage and integration of free, web available tools is viable and effective.

The remainder of this paper is organized as follows. Section 2 contains the theoretical background of our proposal, which includes more in depth discussions of the botnet problem and the mashup technology. In section 3, we depict our mashup-based solution. Section 4 we evaluate our proposal qualitatively. In the Section 5 we lay out the obtained conclusions and present the future work.

II. Background

A. Botnets

A botnet is a set of compromised machines controlled by a botmaster. To maintain such control, the botmaster employs communication channels to send instructions to each bot. Usually those instructions will be executed simultaneously by a large number of bots. Previous analysis has showed that most botnets Communication and Control (C&C) channels are based on the IRC protocol, however any other communication channel can be used. For example, the worm W32.Spybot.ABDO uses port 53/TCP, typically used by DNS, for its IRC communication with the controller [symantec]. Each botnet, thus, can implement its own communication channel to control the entire network, hence, protocol-based detection is limited.

An effective way to identify the communication channel is to analyse the behavior of bot-binaries and try to gather information about the botnet controller. Most bot-binaries have information about the botnet controller, such as a pair IP-port, which is employed to instruct the affected system to connect to the master and become a member of the botnet. However, extracting this kind of information is not a trivial task due to factors such as techniques employed to obfuscate the binary code and polymorphism methods. *Sandboxing* proposes a way around those techniques. It consists in understanding the behavior of a malware and collecting information about the botnet controller by actually executing the botnet-binary in a controlled environment. Sandboxes are usually virtual machines prepared to execute suspected binaries and produce reports that summarize the program behaviour. The analysis of this behaviour allows one to trace and identify the network access launched by the bot-binary and makes it possible to identify the botnet controller. The identification of the network controller is an important step to inhibit a botnet attack. The isolation of the botnet controller prevents infected machines to receive instructions, thus removing them from the botnet.

B. Mashups

In the past few years, new trends in Web applications reunited under the title of Web 2.0 gained focus from both industry and academy. These trends propose Web applications that focus on the end-users, encouraging them to assume a role as producers and organizers of content, instead of just consumers. They also include principles such as reuse of content and cooperation between users [14]. In this context, Mashups are Web applications created from the composition of existing Web resources, such as Web pages, RSS feeds, Web Services and online APIs [10].

The composition of these resources can be made in an *ad-hoc* manner. This approach consists on the usage of Web development technologies such as AJAX and DHTML to leverage content and functionality available online into a new, value-added application. Such task is facilitated by new technologies such as Feeds, microformats and Web Services. However, the *ad-hoc* integration is still a prerogative of skilled web developers [15]. However, the key advantage of the Mashup proposal is the objective of enabling an end-user with no programming expertise to perform resource composition and ultimately create their own applications through applications specially designed, called mashup systems.

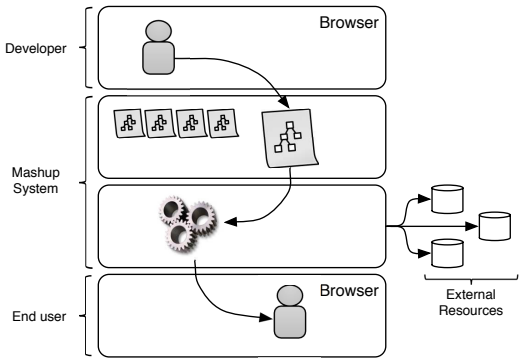

Fig. 1. Mashup system architecture

A mashup system, as depicted in the figure 1 is an environment for mashup creation, similar to an Integrated Development Environment (IDE), which also performs the role of a runtime environment that will execute the mashups, and a repository for such mashups. Mahsup systems are normally Web applications implemented using popular technologies such as PHP and Java Server pages (JSP), and present sophisticated graphical Web interfaces created with usability-oriented technologies such as Asynchronous Javascript and XML (AJAX) [16] and Macromedia Flash. These technologies are employed to make the system as easy as possible for the end-user, abstracting the technical details inherent to the composition of resources from heterogeneous sources, which are dealt with by the mashup system itself.

III. New Approach

In this section, we present our mashup-based solution for botmaster detection. Such solution was created using a highly modular mashup composition architecture, called Network Management Mashup System (NMMS) [17], which provides a good basis for mashup developments. The created mashup, integrates information from online binary analysis tools, and displays the found botmasters in a map-based Web page using their IP addresses. The mashup also provides an interface for binary submission and an online database that stores all information about already analysed binaries.

A. Created Mashup

We used the NMMS due its flexibility in adding new functionalities. The NMMS relies on visual modules to provide a way to users with no programming expertise to define their own applications. In a development environment, one user can through dragging-and-dropping those visual modules define

the composition logic. This logic is executed by a runtime engine, also available in the NMMS. The easiness of using this system is achieved through wrappers, that are components acting as gateways that retrieve and standardize external information, abstracting technical details of such access. The task of creating such wrappers is responsibility o skilled programmers, who follows a special defined methodology to create those them.

Many modules were created during the NMMS development, such as **Geolocation** and **Map**. The first interacts with *hostip.info* IP geolocation service, receiving an IP address as input and outputting the approximate geographic coordinates of the IP address. The second builds a visual Map using the Google Maps API. To this work, we developed other modules to perform specific operations necessary to our purposes. Those modules are:

- **Analysis**: receives the binary file uploaded by the network administrator, and forwards it to online analysis tools. Its output is a set of information regarding of the infection kind, C&C Sever IP address, and time of the analyze;
- **Whois**: retrieves the ASN based on an IP address. It was used to discover from which ISP the C&C Server is based. The information is retrieved from a Whois Web page, which receives just the IP address.

The overall logic of our mashup for botmaster detection is presented in Figure 2. The mashup receives as input infected binaries, which can be submitted by users or gathered in honeypots set by network administrators. Such binaries are then submitted to three online binary analysis tools. CWSandbox and Norman, will execute the binary and analyse its behaviour, and Virus Total will scan the binary for virus signatures.

With the C&C information, we employ our **Geolocation** and **Whois** components in order to discover both location of the C&C Server, and its responsible ISP. This information is then aggregated and sent to the **Map** component, which generates an interactive map-based Web page. This Web-page, present in Figure 3, allows the visualization of C&C Servers spread around the world through markers representing incidences of binary vectors of that botnet, as show in Figure 3. Furthermore, the information gathered is also available in a data feed of detected botnet vectors and C&C Servers using different data formats (*i.e.* XML and JSON). This information is also timestamped, allowing the creation of temporal representations of the infections.

Besides the existence of some available tools to detect botnets, usually they are paid and complex to use. We believe that retrieving and correlating information available in the reports generated by disparate sandbox and anti-virus online tools is a better alternative. The adaptability obtained with the mashup-based approach is important due to the heterogeneity of technologies employed in C&C channels. Since each sandbox tool is able to analyse and detect a specific set of technologies, the usage of multiple tools allows the mashup to detect different kinds of infections. Furthermore, it allows constant self-actualization, through the addition of new tools to detect new kinds of infections. The support of new analysis mechanisms is possible by the creation of wrappers, which is a straightforward process.

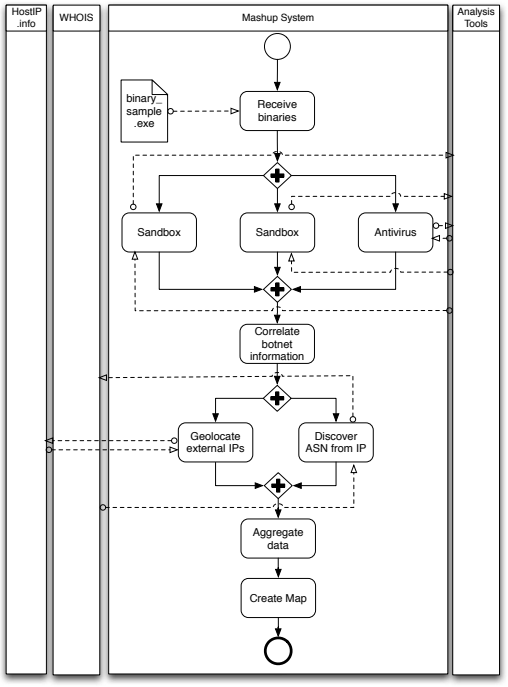

Fig. 2. Design of Botmaster Detection Mashup

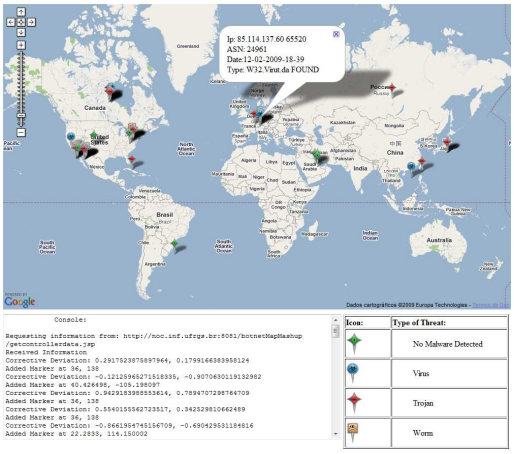

Fig. 3. Mashup for Botmaster Detection

Finally, mashups are built with reusability in mind. In this case, this characteristic allows the created mashup to be used as a building block for more sophisticated botnet detection mashups. For instance, it is possible to develop a mashup able to identify P2P botnets. Further possibilities include the implementation and addition of firewall filters to automatically isolate the C&C channel from the network potential hosts, and automatically detect infected hosts inside the network through flow analysis. Other techniques based on traffic analysis can also be used to detect whose botnets.

B. Correlation

The main function of our mashup is its ability to detect the botmasters based on the information provided by online binary analysis tools. In that way, we developed, and added to NMMS, one component to collect and analyse reports generated by those tools. This component, presented in Figure 4, is based in three steps, to know: *recovery reports* (1), *process reports* (2), and *create signatures* (3).

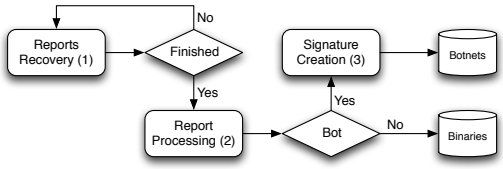

Fig. 4. Correlate Botnet Information

The *reports recovery* step verifies if the submitted files were already been processed by the online analysis tools. Such tools are queried periodically to verify if the report is already available in a provided URL. When the report is available, all of its information is stored, and sent to the *report processing* step. The reports generated by those tools contains different information: report of the malware functionalities in XML format (CWSandbox, Anubis); network traffic generated by the infected machine in *pcap* format (Anubis).

With those information, we developed a simple heuristic that extracts and correlates the desired data from the reports generated. The reports generated by CWSandbox already provide information from possible botnets. It is based on such report that we look for the same behaviour (network activity) in the reports generated by Anubis. If the same behaviour is found in both reports, a signature is created and stored. The definition of a more sophisticated heuristic can significantly improve the results obtained, however, it is beyond the scope of this work, which focuses on the mashup approach as a whole. In follow, we present one example of a generated signature. Its possible to observe a set of information related to the bot actions, such as remote address and port, and used protocols.

```
<xml>
<botnet_signature>
<id>0234</id>
<remoteaddr>85.11.143.208</remoteaddr>
<remoteport>65520</remoteport>
<protocol>TCP</protocol>
<date>Ter Jul 27 15:57:43 UTC 2010</date>
<comment>automatically generated</comment>
</botnet_signature>
</xml>
```

IV. Qualitative evaluation

In this section, we evaluate our mashup for botnet mitigation approach. This evaluation is divided in two aspects. Firstly, we evaluate our mashup based approach qualitatively, doing an analysis of its advantages and limitations. We then make a comparative evaluation, where we compare our approach with the current botnet mitigation and information gathering techniques presented in Section 3.

In order to qualitatively evaluate our approach, we determined some characteristics that will be discussed in details following. Such characteristics, presented bellow, will be the focus of this subsection:

- **Flexibility:** relates to the capability of adaptation to new scenarios, such as dealing with new botnets or new C&C technologies;
- **Extensibility:** the possibility of reuse by different tools or different approaches;
- **Implementation effort:** the effort needed to implement the theoretical approach;
- **Usage effort:** the effort needed to use the botnet mitigation mashups once they are implemented;
- **Reliability:** how reliable are the botnet mitigation mashups, specially under critical scenarios.

The flexibility of our mashup is based on the capability of integrating new botnet information and mitigation tools. This is special interesting for botnet mitigation since such networks are always evolving. New protocols for C&C channels, infection methods, and obfuscation techniques are always being used and obligating the employment of new techniques and tools to deal with them. Wrappers can be created to these new tools [17], enabling them be integrated in our botnet mitigation mashup. Additionally, non-exclusive security tools can be employed to increase the effectiveness of the mashup approach. For instance, a module to analyse Netflow flows and find C&C channels can be developed, and a firewall module can dynamically create firewall rules to stop such communication.

The flexibility is obtained due the modularity of our mashup system. We proposed and implemented modules (*i.e.*, wrappers) to access different online sandboxes and anti-viruses systems. Such modules was composed with pre-existing modules (*i.e.*, geolocation and mapping) created originally to other context (*i.e.*, BGP peering) [17]. The result of the composition (*i.e.*, the botnet mitigation mashup presented in Section 4), can be reused and extended by other users of the mashup system, but also, by users whom don't use the system, for example, by accessing the report available in different data formats (*i.e.*, JSON, CSV, XML). This report includes information the following information: type of infection, botnet name, ASN of the infected host, and geographic and chronologic information.

The reduction of the implementation effort is obtained by the existence of wrappers to access the external systems. In fact, using a mashup-based approach reduces this effort to the creation of independent wrappers by software developers. The integration of those wrappers is responsibility of the network security administrator, who knows well about security but is not necessarily a skilled programmer. About the usage effort, the mashup approach gives the end user the possibility to easily manipulate the components and alter the composition, which adds another layer of extensibility, as previously discussed.

We are aware of the reliability limitation of our approach. This happens because the mashup depend on free online and external tools, which can fail. Problems such as offline services, overloaded services, discontinued services and API changes may arise with the usage of such online tools. Redundancy can be, and was, used to mitigate this issue. Similar services are available online, such as the sandboxes employed in our implementation. Furthermore, once the information about botnet infection is gathered, it is stored in the mashup system. In case of failure of the external services, binary analysis is compromised, but existing reports remain functional and useful. Thus, reliability of our approach is adaptable to the administrator needs, but it demands an adequate choice of components.

V. Conclusions

In this paper, we proposed a mashup-based botnet detection approach. In our proposal, we cover the integration of various online available analysis tools with external resources. Based on this proposal, a mashup prototype was developed with information about botnets detected using free online sandboxing and anti-viruses tools with geocoding, Whois and mapping online services.

Due to the possibility of integration of various techniques, including ones yet to come, to cover different kinds of botnets, we reached the conclusion that the mashup-based approach is adequate to the ever changing scenario of botnet detection. However, this approach does not aim to substitute other botnet detection efforts, but to leverage and correlate them to optimize the obtained results and create new ones from the integration. It is our understanding that the fluid and dynamic nature of botnets as a security threat demands extensible and flexible solutions, as is mashups.

Having implemented our botnet detection mashup using the NMMS, we conclude that a mashup-based approach reduces the necessary effort to create new applications. With adequate wrappers, the integration can be performed/re-used by network administrators and/or security experts even without programming expertise. Our implementation, while providing good information and coverage, uses free available components (*i.e.*, online sandbox, anti-viruses, geolocation tool, mapping API).

As future works, we aim at extending the NMMS architecture to support the execution of actions based on the result of a composition. With such mechanism, we be possible to, for example, configure Intrusion Detection Systems (IDS) or firewalls when botnet infections are detected on the network allowing the mitigation of such botnets. We also want to investigate other interesting techniques, for example the ones based on traffic analysis. We believe that the integration of even more information sources can bring better results in botnets detection.

References

[1] M. K. Paul Bacher, Thorsten Holz and G. Wicherski, "Know your enemy:tracking botnets," Tech. Rep., 2005. [Online]. Available: http://www.honeynet.org/papers/bots

[2] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, "Peer-to-peer botnets: overview and case study," in *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. Berkeley, CA, USA: USENIX Association, 2007, p. 1. [Online]. Available: http://portal.acm.org/citation.cfm?id=1323129

[3] E. Stinson and J. C. Mitchell, "Towards systematic evaluation of the evadability of bot/botnet detection methods," in *WOOT'08: Proceedings of the 2nd Usenix Workshop on Offensive Technologies*. San Jose, CA, USA: USENIX Association, 2008.

[4] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, February 2008.

[5] J. Goebel and T. Holz, "Rishi: identify bot contaminated hosts by irc nickname evaluation," in *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. Berkeley, CA, USA: USENIX Association, 2007.

[6] D. Dagon, G. Gu, C. Lee, and W. Lee, "A taxonomy of botnet structures," in *Proceedings of the 23 Annual Computer Security Applications Conference (ACSAC'07)*, December 2007.

[7] N. Daswani and M. Stoppelman, "The anatomy of clickbot.a," in *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. Berkeley, CA, USA: USENIX Association, 2007, pp. 11–11.

[8] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling, "Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm," in *LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*. Berkeley, CA, USA: USENIX Association, 2008, pp. 1–9.

[9] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the 17th USENIX Security Symposium (Security'08)*, 2008.

[10] D. Merril, "Mashups: The new breed of web app - an introduction to mashups." 2003, http://www.ibm.com/developerworks/web/library/x-mashups.html.

[11] International Secure Systems Lab Vienna University of Technology, Eurecom France, UC Santa Barbara, "Anubis: Analyzing Unknown Binaries," available at: http://anubis.iseclab.org/. Accessed in April 2008.

[12] Ben Stock 2007-2008, Lehrstuhl fr Praktische Informatik 1, University of Mannheim, "CWSandbox - Behavior-based Malware Analysis," available at: http://cwsandbox.org/.com. Accessed in April 2008.

[13] Norman SandBox Information Center (NSIC) , "Norman Sandbox Information Center," available at: http://http://www.norman.com. Accessed in April 2008.

[14] T. O'Reilly, "What is web 2.0," 2005, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

[15] J. Yu, B. Benatallah, F. Casati, and F. Daniel, "Understanding mashup development," *IEEE Internet Computing*, vol. 12, no. 5, pp. 44–52, 2008.

[16] J. J. Garret, "Ajax: A new approach to web applications," 2005, http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[17] R. S. Bezerra, C. R. P. dos Santos, L. Z. Granville, and L. Tarouco, "On the feasibility of web 2.0 technologies for network management: A mashup-based approach," in *NOMS 2010 - TechSessions*, Osaka, Japan, apr 2010.

Performance Management and Quantitative Modeling of IT Service Processes Using Mashup Patterns

Carlos Raniery P. dos Santos,
Lisandro Zambenedetti Granville
Institute of Informatics - UFRGS
Porto Alegre, RS, Brazil
Email: {crpsantos, granville}@inf.ufrgs.br

Winnie Cheng, David Loewenstern,
Larisa Shwartz, Nikos Anerousis
IBM Watson Research
Hawthorne, NY 10532, USA
Email: {wcheng, davidloe, lshwart, nikos}@us.ibm.com

***Abstract*—IT Service Management (ITSM) encompasses the practices for managing information technology systems. ITSM processes can be laden with segments where the human becomes a bottleneck and slows down the entire process. These inefficiencies are usually caused by insufficient design of the process itself, or defects in the tools being used. Our work provides a systematic framework for analyzing inefficiencies through a combined model to guide the use and estimate the value of improving orchestration of the process using mashup design patterns.**

I. Introduction

IT Service Management (ITSM) encompasses the practices for managing information technology systems. A significant body of work in this field addresses the issue of quality, *i.e.*, the frameworks, processes and metrics that measure effectiveness from the point of view of the receiver of such services. In this paper we study the service provider's perspective, particularly aspects of performance that have direct implications on the efficiency and cost of the operation from the provider's point of view.

In ITSM, a very large percentage of the work is performed by humans, rather than machines. Due to its unpredictable nature, human behavior and performance are much harder to model, and consequently, to optimize. Consider the example of a modern data network that receives packets at an entry point and needs to transfer them to a destination. The data packet in its path will be processed by a variety of system elements, each programmed to perform a specific task with a high amount of accuracy and predictability. The number of events (exceptions) that can interrupt a normal processing path can be large, but are always finite, and in many cases can be accounted for in the design itself through redundancy and error handling programs. By contrast, consider a service management operation organized according to the Information Technology Infrastructure Library (ITIL) standards. The presence of humans in the critical path for performing work introduces significant variability in the final outcome. Even if the nature of work is exactly the same, a human operator may execute it in a different way each time: she may use a different process; or different tools; or a different sequence of steps; or be interrupted a number of times by external factors such as a telephone call or email. Enforcing and obtaining tight performance bounds in a human-staffed organization is far more difficult than in a process executed by a machine.

The competitive nature of IT service provider organizations calls for a continuous improvement process: IT operators need to find ways to increase performance in terms of effectiveness, productivity and quality. We focus on productivity and define it as units of work performed per unit of time; but even with rigorous definitions, performance can be evaluated in many ways and at different levels of granularity. In this paper, we attempt to provide a comprehensive model for evaluating and optimizing productivity in human-centered ITSM processes. In our analysis we do not address exogenous events, such as answering telephone calls or other interruptions. Rather, we focus on individual steps in the process that can be measured through instrumentation or observation, and can be improved through design and automation. In particular, we study the request fulfillment process, one of the operational service management processes defined by ITIL. The analysis covers the steps that a human follows to execute the process, and identifies the areas where productivity improvement is possible. As we will see in the next section, ITSM processes can be laden with segments of the process where the human becomes a bottleneck and slows down the entire process. These inefficiencies are usually caused by insufficient design of the process itself, or defects in the tools being used.

Our work provides a systematic framework for analyzing inefficiencies, and addressing them through a set of design patterns that ultimately provide a significantly improved orchestration of the process. Again, our framework does not address the rather unpredictable and chaotic nature of external events that affect human behavior. After all, we cannot control when humans decide to interrupt or slow down the process – but we can measure and control the environment where productivity is limited due to lack of tools or inefficient orchestration.

The paper is organized as follows: in Section II, we discuss the technical background related to this paper. Section III

defines inefficiencies in the ITSM context and proposes a model for measuring them. Section IV introduces the concept of mashup patterns as an effective approach for eliminating inefficiencies in an ITSM process. Section V demonstrates the application of the proposed methodology on the request fulfillment process and presents results of our case study. The paper concludes in Section VI with a brief summary of our findings and an outline of future work.

II. Background

A. Mashups

Mashups are Web applications created through the composition of pre-existing Web resources (*e.g.*, interactive maps, Web services, traditional HTML pages, or even Flash presentations) [1]. A key difference between mashups and traditional composition technologies like BPEL [2] and WSCI [3] is that mashups have the explicit goal of enabling users with limited or no programming skills to create their own tailored Web applications; traditional technologies, in turn, usually demand from developers a significant knowledge of programming languages, communication methods, and service description. Because mashups can be quickly created, they present the additional benefit of being appropriate for composing *situational applications*, *i.e.*, applications that tackle very particular, short-lived problems and so would be otherwise expensive to be coded by specialized personnel.

Mashups are composed through the use of *basic operators*. These operators hide from mashup end users how the original Web resources are orchestrated, so that users are exposed to the resulting mashup without being aware of the internal details of the composition. The coupling of such mashup operators can be guided in several ways [4]. Using metadata available in the operator's definition, the mashup system engine can select *default bindings* between the operators that are being used during the composition step. Compatibility rules and quality criteria can be used to suggest the most appropriate operators for a given one. In [5], we have defined the architectural components employed in this paper, and presented different categories of mashup basic operators, listed below.

- **Visual** operators deal with the visual presentation of relevant information through, for example, tables, graphs, and maps;
- **Control** operators relate to basic programming logics including loops and conditions;
- **Transform** operators manipulate data employing, for example, sorting and filtering;
- **Adaptation** operators translate original data from Web resources into formats more easily handled inside mashups;
- **Input** operators allow end users to feed mashups with their particular information, for example, through text fields in Web forms or by uploading files;
- **Execute** operators trigger the asynchronous background execution of actions without the explicit request of the end user, which is typical in background monitoring systems and similar applications;
- **Reuse** operators allow users to extend the available mashups to build more sophisticated compositions.

By using mashup basic operators, a user is able to specify mashups for a variety of different purposes. Where several problems share similar structure, however, it is often convenient to consider the employment of *mashup patterns* [6], where mashups with similar logic can be instantiated even more quickly from the same common pattern. In addition, because patterns enable previously proven mashups to be reused in new scenarios, mashups patterns provide an additional level of stability. Considering these benefits, in Section IV a set of patterns will be defined to tackle the inefficiencies discussed in Section III. In Section IV, we also present a quantitative evaluation of the potential impact of employing such patterns in ITSM using a combined model created based on the further-coming ones.

B. Quantitative Modeling for Performance Assessment

Despite its importance, by design ITIL only provides high-level generic guidelines to IT organizations, without proposing, for example, concrete models and methods for capturing metrics and evaluating the quality of IT processes. Such evaluation is important for the IT service providers to quantify, measure, and most importantly to predict the deployment impact of IT solutions. The scientific community has worked to propose models, methodologies, and metrics to fill this gap. We present two of such models bellow, and then, in the next section, we propose a combined model to account for inefficiencies throughout the IT process.

1) Keystroke-Level Model: The Keystroke-Level Model (KLM) was proposed to predict the time an expert user takes to perform a given task on a given computer system [7] [8]. It is based on the sequence of keystroke-level actions the user must perform to accomplish a task. This sequence is taken from a set of gestures, presented in Table I, where the total task execution time is the sum of the time for each of the gestures in the sequence. The model also provides the average time for each gesture as presented below.

TABLE I
KEYSTROKE-LEVEL MODEL

	Gesture	Time
K	Keying	0.2 sec
B	Holding/Releasing key	0.1 sec
P	Pointing	1.1 sec
H	Homing	0.4 sec
M	Mentally Preparing	1.35 sec

As an example of the use of KLM to predict interaction time, we can consider the following scenario: a file deletion by a human operator. In this simple case, we consider that the procedure is to drag the file icon to the trash can icon. For this, the action sequence can be represented as follows:

1) Initiate the deletion (decide to do the task) **M**
2) Point to file icon **P**
3) Press and hold mouse button **B**
4) Drag file icon to trash can icon **P**

5) Release mouse button **B**

$$T_{total} = 2P+2B+M = 2*1.1+2*0.1+1.35 = 3.75 \text{ sec} \quad (1)$$

2) Complexity Model: Brown and Hellerstein [9] introduced a methodology for quantitative benchmarking of configuration complexity of initial system setup, which has been extended in subsequent works. Brown *et al.* [10] proposed a model of configuration activity based on three concepts: configuration goals, procedures and actions. This model, along with the earlier methodology, allows an analyst to measure the complexity of different systems through proposed metrics classified into: execution, parameter, and memory complexities. Further, Diao *et al.* [11] extend these techniques to propose new complexity metrics and measure business-level performance indicators (*e.g.*, labor cost, productivity, quality). Finally, Diao and Keller [12] extended the metrics proposed in [10] to quantify the complexity of overall IT processes.

The approach used by Diao *et al.* can be summarized in three steps: assessing the complexity and timing a baseline scenario, construction of the regression model and evaluation of the model quality, and finally employing the model to predict labor costs, such as time. The relationship between time and complexity metrics are investigated using the multiple linear regression technique, with equation presented below:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \quad (2)$$

In this equation, the $x_i$represent the IT management complexity metrics, and the least squares approach is employed to discover the β_i value. Further explanation of this methodology is beyond the scope of this paper and can be found in [11].

A simpler version of this model is presented in [13]. In this version, the complexity metrics are grouped by subtask, and the coefficients of Equation 2 are assumed to be proportional to the time spent on each subtask and equally weighted among metrics associated with the same subtask. This simplifies the model to the point that it can be applied with nearly the same ease and generality as the KLM model of the previous section, but addressing a much larger array of metrics. This work also extended the model to cover more complex processes which include forks, merges, and joins.

III. Identifying Inefficiencies in Service Management Processes

Inefficiencies are portions of a service management process characterized by suboptimal execution of activities. In this paper, we concentrate on inefficiencies characterized as segments of the process where suboptimal human productivity reduces overall throughput for the process. Inefficiencies can appear at fundamentally different levels of analysis. For the purpose of analyzing inefficiencies within ITSM we focus on two levels: higher level inefficiencies due to the complexity of the activity itself, and lower level inefficiencies due to the mechanical execution involved in performing the activity. As an example of a potential complexity inefficiency, in a process with many decision points, the operator needs to spend time determining the correct choice. In ITSM the lower level takes into consideration the interaction of human operators with the available software tools. For example, a web application created with poor usability can impose a significant amount of wasted time for the operator due to added mouse-clicks and keystrokes required to retrieve, create or update information.

In order to discover common inefficiencies, we initially collected descriptions of the tasks performed by a group of operators involved in a common activity, as will be discussed in more detail in Section V. Based on such descriptions, we recreated their processes in a form of sequence of tasks (*i.e.*, workflows), which were later validated by the operators. We then drilled down on these workflows to subtasks representing individual actions performed by the operators when interacting with the systems. With this detailed process, we could analyze both levels of inefficiencies. The non-exhaustive list below represents several types of inefficiencies we have found during our investigations:

Basic Inefficiencies

- **Context-Switching**: the operator needs to switch to different application from the one he/she is currently working on;
- **Locating Data**: after reaching the place where the information is available, the operator needs to search for the specific data across the screen;
- **Entering Data**: the operator has to input data manually in the screen he/she is working on;

Information Management Inefficiencies

- **Copy/Paste**: manual copying of data from one system to another;
- **Consistency checks**: the operator needs to guarantee that information is consistent in different places;
- **Information Lookups**: navigate between multiple screens to assemble information;

Skill-dependent Inefficiencies

- **Retaining Information**: remembering information for a subsequent step;
- **Combining Information**: all the data is in one screen and the operator needs to extract their meaning;
- **Data transformation**: the data require some simple manual processing (*e.g.*, reformatting dates to a local format) when transferring from one screen to another.

Synchronization Inefficiencies

- **Contacting a Person**: the operator needs to talk to someone by e-mail, instant messenger or in person;
- **Becoming aware**: the operator needs to access a tool repetitively to be aware of new service requests;

We classify the inefficiencies into four categories: basic, information management, skill-dependent, and synchronization. The first refers to the most simple and low-level inefficiencies, occurring independently from the others. Information-management inefficiencies are formed by the combination of several basic inefficiencies. Skill-dependent inefficiencies relate to the reasoning capabilities or training of the human

operator. Finally, synchronization inefficiencies are those incurring delays due to factors such as waiting for an external input.

The identified inefficiencies can be mitigated by the use of a new generation of human-centric software tools aimed at decreasing both the actions required to complete a task and also the complexity encountered in carrying out the tasks.

A. Combined Model

We combine the models of Section II-B to take advantage of their relative merits. The KLM model is more widely used, is well corroborated by experiment (see for example [14]) and provides a wealth of detail at the lower level of human-computer interactions, while the Complexity model addresses both levels of inefficiencies. Thus, the presented models are the current best starting points for creating a new model to better evaluate the performance of IT processes from a time productivity perspective. To avoid double-counting, we discard all the complexity metrics except the memory and decision metrics, which capture higher level potential inefficiencies not addressed by KLM.

An analyst constructs the combined model in the following stages:

1) The analyst works with a domain expert to determine the tasks and subtasks of the process.
2) The analyst works with a domain expert to determine the complexity metrics for the Complexity model.
3) The analyst determines the KLM model through observation of user interactions.
4) The analyst measures the time to perform each subtask.
5) The analyst derives the Complexity model coefficients from the time measurements using the method of [13].
6) Since β_0 of Equation 2 represents the expected time for all factors not explained by the complexity model, the time predicted by the KLM model can be subtracted from it.

The value of the combined model is in allowing us to predict the expected change in time due to modifications in the process. If we create a standardized set of modification templates, we have the potential to search through these templates to find an optimally modified process, along with the expected time savings. We have identified a particular set of modification templates that apply specifically to subtasks involving interactions with a user interface in processing information. These *mashup patterns* form the building blocks for quantitatively motivated process improvement in human-computer interactions within ITSM.

IV. Mashup Patterns for ITSM Inefficiencies

As previously discussed, IT processes can present inefficiencies at various levels of analysis. Some of these inefficiencies are related to the interaction of users with the available tools, while other issues involve the complexity of performing a particular activity or providing mechanisms to decrease the failure risk of a specific action. We argue that these inefficiencies can be tackled by the adoption of mashups and, more specifically, by using mashup patterns. The mashup patterns described below are context-independent and can be used to address different ITSM scenarios. For each proposed mashup pattern, we discuss a relevant ITSM problem and the associated solution that employs that pattern.

Alerter pattern

Problem – In ITSM, it is common to find scenarios where a user needs to be aware of events in the managed environment. The simplest method to support this involves periodically accessing the management system to manually look for new events. For example, in the service dispatching scenario, service tickets are created at no specific time, and a dispatcher responsible for assigning those tickets needs to constantly access the ticketing system to check for new requests. That can become a problem if the dispatcher does not access the system sufficiently often, or if the time spent in unnecessary repeated accesses degrades the dispatcher's productivity. It can be even worse when the amount of monitored information is very large, or when the dispatcher needs to promptly react to time-sensitive events.

Solution – Mashups are not restricted to constantly interacting with users to perform some action. A mashup alerter pattern periodically monitors a system of interest on behalf of the user and, based on previously established conditions, sends notifications only when events of interest take place. For example, alerts can take the form of visual elements on the user's console, e-mail messages, or SMS (text) messages. Another advantage of using an alerter mashup pattern relates to situations where multiple systems must be monitored at the same time, eventually overloading the human operator with too much information. In that case, correlated events from different systems can be summarized to decrease the number of notifications. Figure 1-A presents how an alerter pattern can be created through the combination of mashup basic operators.

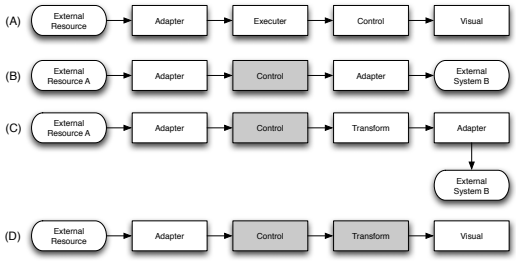

Fig. 1. Examples of mashup patterns. (A) Alerter. (B) Importer. (C) Transform. (D) Displayer.

Importer pattern

Problem – In ITSM, it is not uncommon to find scenarios where customers and service providers require the use of common data, although they use their own, particular database systems. To maintain data consistency across such systems, diverse methods can be used. For example, data adapters

can grant one party access to the system of another's. When adapters are not available, screen scrapers can be used to access the Web interface of the remote system and retrieve the common data. Finally, users can access one another's system and manually copy and paste the common data into their own system's interface. In all these cases, maintaining data consistency is not transparent for the users because they need to consciously switch the integration method when accessing multiple systems.

Solution – If external resources natively expose an application programming interface (API), then leveraging their information is just a matter of basic software programming. However, it is often the case that the most valuable content is locked away in closed or proprietary formats. In these cases, an importer mashup pattern abstracts the different methods used to access the external data so that data consistency maintenance becomes transparent to the user. Figure 1-B presents how such an importer pattern can be implemented.

Transform pattern

Problem – While interacting with different systems, it is common to find cases where data needs to undergo some simple processing while being transferred from one screen to another. For example, while copying a field, a user needs to apply rules to filter out confidential information, or the data needs to be reformatted before it could be used by a different system (*e.g.*, US and UK date formats). These data transformations are usually manually performed because ITSM systems are often created without having integration in mind.

Solution – During the process of importing data, transform operators can be inserted into the mashup logic to enable the processing of certain types of data and thus both materializing the compatibility between systems and satisfying the requirements of the IT process. It is thus possible to reduce the number of manual interventions performed by the human through the automation of these adaptations. Figure 1-C highlights some elements to show that they are not required when transforming external data.

Displayer pattern

Problem – In order to make better decisions, humans involved in ITSM activities use information from multiple systems. This information is often memorized or recorded for future use during the decision making process. For example, the configuration database process can automatically generate a port number that needs to be remembered when installing another application. If the port number is forgotten or misremembered, errors in the process may occur.

Solution – By definition, mashups combine data from multiple sources and present the results of this combination in a Web page. However, this integration tends to occur only at the presentation level; it rarely occurs at the data level. This means that information from multiple systems can be presented alone in the same Web page as independent widgets. The employment of many displayer patterns in one page enforces the concept of a "single pane of glass". This concept reduces the risks of having a poorly executed process, which would generate errors and impose costs to the company. Figure 1-D presents the operators composing the displayer pattern.

A. Quantitative Perspective

The methodology presented in Section III-A allows us to estimate time savings for our mashup patterns. By doing this, we aim to help users to predict the performance improvements quantitatively before deploying mashups over their current ITSM processes. We will use the scenario described in the alerter pattern as an example.

The scenario described in the alerter pattern is a task composed of several subtasks. The first subtask is for the operator to notice that it is time to check for new events. We will label the time spent "becoming aware" of the need to start the task T_a. Once the operator decides to look for new requests, the next subtask is to interact with tools to examine the new events. We will label the time spent on this subtask T_k. This second subtask can be modeled by KLM, while the first subtask demands investigation beyond the scope of this paper. It is important to observe that this scenario does not include a subtask associated with memory complexity. This is because the human operator does not need to retain any information to look for new requests. We consider all requests to be processed independently of the others. The alerter pattern can decrease time spent on the task by reducing the awareness time T_a to zero. Once the requests arise, notifications are sent to the human operator automatically. In addition, a well-designed implementation of the alerter pattern could reduce T_k, for example by allowing the dispatcher to access the ticket associated with an alert with just one mouse click.

The scenarios where the importer and transformer patterns can be applied present both kinds of time inefficiencies we focus on this paper: mechanical execution (T_k) and task complexity (T_c). Since both operations can be completely automated by employing mashup patterns, the time reduction in those scenarios is 100%. The same applies to the displayer pattern. Since the necessary information to process a request is provided in one single screen to the human operator, the time spent looking for the information is decreased to zero and the time associated with complexity is significantly decreased due to eliminating the need to remember one specific piece of information. Table II summarizes the time reduction estimation for each of the presented patterns.

TABLE II
TIME SAVINGS

Pattern	Current	New
Alerter	$T_a + T_k$	T'_k
Importer	$N_{fields}.(T_k + T_c)$	0 sec
Transformer	$T_k + T_c$	0 sec
Displayer	$T_k + T_c$	T'_c

The next section will provide a case study of the application of the quantitative methodology presented here to an existing ITSM activity.

V. Case Study

Amalgamation of multiple systems in a single pane of glass is critical in the area of IT Service Management, in particular in the area of IT Operations. Our case study considers the Request Fulfillment process, which is one of the operational processes in IT Management. Request Fulfillment is the process that deals with service requests, and it is defined in ITIL terminology as "management of customer or user requests that are not generated as an incident from an unexpected service delay or disruption." [15].

Request Fulfillment interfaces primarily with Service Desk and Incident Management, and supports two functions: it provides a point of communication for users and serves as a point of coordination between several groups and activities. In our study we focus on the latter function of Request Fulfillment. The process for this case study breaks coordination into two main activities: support the requests made by the customers, and solve those requests. Requests are solved by system administrators (SAs) with technical knowledge to resolve specific requests. Requests are supported by human operators, called dispatchers, with responsibilities that include: monitoring for new requests, dispatching the requests to the appropriate SA, and monitoring compliance with Service Level Agreements (SLAs).

A. Dispatch Process for Service Management

We study the case where the dispatcher has knowledge of standard fulfillment procedures and responsibility for generating requests and assigning them to a system administrator (SA). The Service Desk receives requests and creates a ticket, which may be any of the following types: incidents, problems and changes. Tickets are routed to a dispatcher, who is responsible for analyzing the request and determining the appropriate SA to assign it to for a resolution. The SA has the required skills and knowledge to solve specific requests and is responsible for taking the appropriate actions and closing the ticket.

Usually, each dispatcher is responsible for a team of system administrators in a specialized technical background. This setting has some advantages, providing high-quality and agile support, and allowing system administrators to work more efficiently.

Customers create new requests (*i.e.*, tickets) in Service Desk systems, and include all information used by system administrators to solve the request. Once the dispatcher receives the ticket and determines that his team can resolve the ticket, he would use his knowledge of his team's schedules and workloads as well as the expertise of each system administrator to finally make the assignment. Figure 2 presents the elements of this scenario.

Several time-consuming issues can arise in the dispatching process, making it infeasible to resolve tickets within the times established in SLAs and therefore resulting in financial loss to service providers. For example, it is common for customers and service providers to use their own ticketing systems, making it necessary to import data and maintain consistency between the systems. Dispatchers need to deal with data consistency and redundancy without violating any customer policies, such as data compliance for dealing with confidential information. In addition, the dispatcher and his team of SAs may be responsible for multiple customers, where each customer has a different ticketing system. This adds additional overhead in switching between multiple systems. Finally, information required for finding the most appropriate SA for one specific ticket could reside in various locations and require different tools to access it. For example, schedules tend to be managed by calendar-based systems, while the SA's actual workloads would be most accurately represented in Request Fulfillment systems and finally it is common to have SA skills associated with their user profile in the service provider's directory.

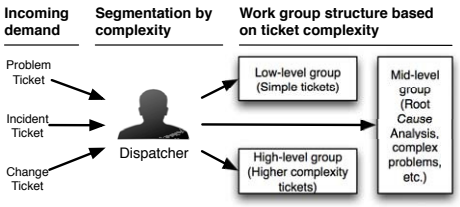

Fig. 2. E-Ticketing system to Service Desk

Automated dispatching solutions may be complex due to the variability of the environment and therefore it is not always feasible or the best alternative for this scenario. For example, in some situations a dispatcher may want to train a new administrator, and so may intentionally assign a request to a less skilled SA than is available. In this context, mashups are an interesting technology allowing the creation of dispatching systems to focus on the process of each dispatcher and helping him improve the efficiency of the assignment.

B. Performance Footprint

In order to discover bottlenecks in the dispatch process, we performed a series of time measurements among four dispatchers in a service delivery center. This information also allowed us to predict improvements obtained by the usage of mashups in the existing process. Using a stopwatch, we took 10 measurements for each assignment process and its individual tasks. This process is represented in Figure 3 as a workflow, which was obtained following the methodology in Section III-A.

The results of the measurements showed a significant time variation according to the ticket's complexity and the dispatcher's familiarity with the reported issue. For simple tickets (TS), usually repetitive tasks that the dispatcher is accustomed to assigning, the time average was 159 seconds (90% confidence level, 23.65 standard deviation), while for high complexity tickets (TC) this time was 357 seconds (90% confidence interval, 41.58 standard deviation). This difference can be justified by the need to spend more time reasoning

about all the information related to the ticket, and also by the need to gather and provide detailed information to the system administrators.

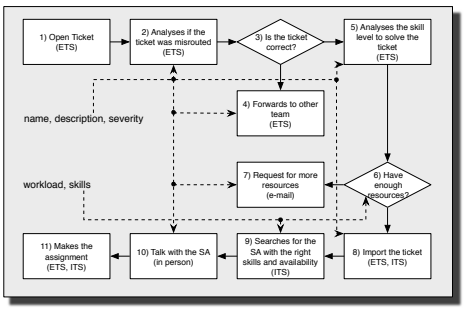

Fig. 3. Workflow of activities performed by dispatchers in Service Desk

Looking at the individual tasks, 35 (TS) and 58 (TC) seconds of the time was spend analyzing if the ticket was misrouted or not. To accomplish this task, dispatchers need to look for the right information (*e.g.*, keywords on tickets description) and decide if their teams have the right knowledge (*e.g.*, database, Unix) to solve the ticket. We also observed that most of the time was spend importing manually the tickets. This task consumed 41% and 50% of time respectively for simple and complex tickets. Finally, 58 (TS) and 94 (TC) seconds of the time were spent making the assignment, activity which involves updating the SA assignment information in both internal (ITS) and external ticketing systems (ETS).

C. Applying Mashup Patterns

All the proposed patterns can be used to create a mashup-based solution for the above-mentioned dispatching scenario. By using them, we aim to improve dispatchers assignment performance by automating some tasks, and by implementing the single pane of glass concept. This concept states that all the necessary information a human operator may need to achieve a goal should be presented in a single screen with the data already filtered and transformed.

Figure 4 shows how the proposed patterns relate with the dispatching tasks. Considering that name, description, and severity are the basic information from a ticket that a dispatcher uses most frequently to make the assignments, the *displayer pattern* can be used to show all the needed information in a single screen. This pattern can also be used to display the system administrator's workload and skills. With this pattern, it is possible to eliminate both *information lookup*, and *retaining information* inefficiencies of this task.

Since the dispatchers need to constantly monitor for new tickets, the *alerter pattern* can be used to notify them about new tickets as soon as they are created, and eliminate the *becoming aware* inefficiency. The *importer pattern* can be used to automate the task of importing tickets to the internal database, and the *transformer pattern* can be applied when the dispatchers need to modify (*e.g.,* augment, exclude) some information, for example, filtering confidential data (*e.g.*, phone numbers) on the ticket's description.

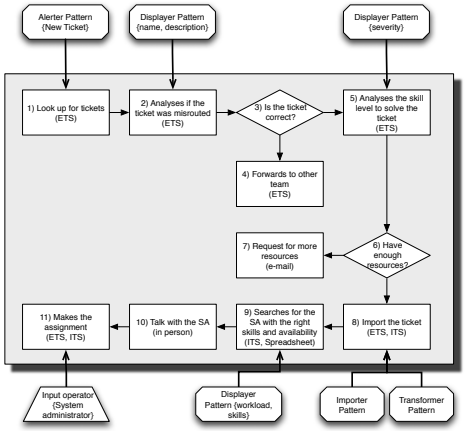

Fig. 4. Relating mashup patterns with dispatching tasks

The final Graphical User Interface (GUI) for the dispatching mashup, constructed based on mashup patterns, is presented in Figure 5. It is important to observe that to create complete solutions, the patterns need to be composed of basic operators. As presented, the **input operator** was used to allow the dispatcher to specify the system administrator who will be responsible for solving the ticket.

Fig. 5. Graphical User Interface (GUI) of the mashup for the dispatching scenario

D. Predictions on Mashups Usage

Analyzing the tasks of the dispatching scenario and using the methodology described in Section 3.C, we estimated the times presented in Figure 6. The bars indicate the labor cost

(*i.e.*, time) for each task of the dispatching process. The different colors indicate the obtained and estimated (with and without mashups) times for the process. As can be observed, the real measurements and the predicted times present a close fit, with a R^2 error of 0.82, which means the model can explain 82% of the variability in the time data, and a Root Mean Square Error (RMSE) of 13.34. This shows the accuracy of our model.

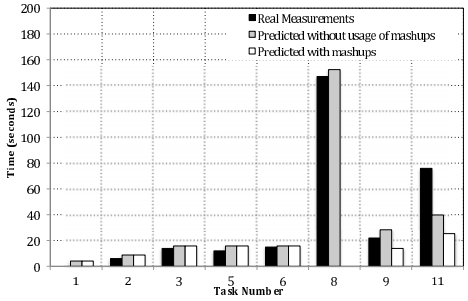

Fig. 6. Quantitative model validation with predictions of mashup cost

We do not present tasks 4 and 7 in Figure 6 because in our scenario, the tickets were not misrouted to the wrong dispatcher and we also considered that he/she always had enough resources (*i.e.*, System Administrators) to solve them. Task 10 is not showed because it was not observed during our evaluations. It is important to observe that this graph does not represent the awareness time since the KLM or the Complexity Model cannot predict it.

The obtained results show the significant time reduction of 64.42%, which mostly was due the automation of step 8 by using the importer pattern. The *displayer pattern* allows the reduction of the tasks 9 and 11, since the dispatcher does not need to look up any information in a different system, thus eliminating all the possible keystrokes, and also does not need to remember any information from a previous task, helping to decrease the memory complexity.

VI. Conclusions and Future Work

We believe that this paper represents a significant contribution: the use of a combined model to guide the use and estimate the value of improving IT processes using mashup design patterns. We have introduced a methodology for analyzing activities performed by human operators involved in IT Service Management. This methodology combines the Keystroke-Level and Complexity models to account for inefficiencies through the IT process. The combined model allowed us to tackle lower level inefficiencies of human-computer interactions, and higher level inefficiencies of performing IT tasks and subtasks, both from a quantitative perspective. As a solution for these inefficiencies, we have considered the employment of mashups and mashup patterns. The use of mashup patterns enabled us to make quantitative predictions of performance improvements due to the use of mashups. The improvements in productivity, usability and agility provided by the application of mashups demonstrate the viability of mashup technology as a means for improving IT Service Management.

Analyzis of the tasks performed by a group of human operators allowed us to find a set of common inefficiencies in their implementation of the ITIL Request Fulfillment process. Since these inefficiencies can be found in many different ITSM scenarios, mashup patterns can be applied as proven and reusable solutions. The analysis of one mashup developed from mashup patterns for a case study allowed us to observe a good fit between the times predicted by our combined model with real measurements. This indicates the feasibility of our methodology in predicting quantitatively labor costs savings due to the use of mashup technology. Finally, this paper demonstrates the significant improvement of 64.42% in the performance of the dispatching process in the case study.

As future research, we will expand our exploration of models of mental activities in business processes. By doing this, we hope to enhance the scope and accuracy of our model in predicting labor costs in the ITSM process. We also plan to extend the analysis of the ITSM process beyond inefficiencies, considering other aspects such as reliability. In this context, a plan to resolve failures in problematic tasks could be identified and instantiated for the final mashup solution. Finally, we will explore the automation of mashup development through the use of the combined model to guide selection of mashup patterns.

References

[1] J. Yu, B. Benatallah, F. Casati, and F. Daniel, "Understanding mashup development," *IEEE Internet Computing*, vol. 12, no. 5, pp. 44–52, 2008.

[2] OASIS, "Business process execution language, version 2.0," Organization for the Advancement of Structured Information Standards, May 2007. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/

[3] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy, I. Trickovic, and S. Zimek, "Web service choreography interface (WSCI) 1.0," World Wide Web Consortium, Note NOTE-wsci10-20020808, Aug. 2002.

[4] C. Cappiello, M. Matera, M. Picozzi, G. Sprega, D. Barbagallo, and C. Francalanci, "Dashmash: A mashup environment for end user development," in *ICWE*, ser. Lecture Notes in Computer Science, vol. 6757. Springer, 2011, pp. 152–166.

[5] C. R. P. dos Santos, R. S. Bezerra, J. a. M. Ceron, L. Z. Granville, and L. M. R. Tarouco, "On using mashups for composing network management applications," *Comm. Mag.*, vol. 48, pp. 112–122, December 2010.

[6] M. Ogrinz, *Mashup Patterns: Designs and Examples for the Modern Enterprise*, 1st ed. Addison-Wesley Professional, 2009.

[7] S. K. Card, A. Newell, and T. P. Moran, *The Psychology of Human-Computer Interaction*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 2000.

[8] D. Kieras, "Using the keystroke-level model to estimate execution times," *University of Michigan*, 2001.

[9] A. B. Brown and J. L. Hellerstein, "An approach to benchmarking configuration complexity," in *In Proceedings of the 11th ACM SIGOPS European Workshop*, 2004.

[10] A. B. Brown, A. Keller, and J. L. Hellerstein, "A model of configuration complexity and its application to a change management system," in *In Proceedings of the 9th IFIP/IEEE Symposium on Integrated Management*, 2005, pp. 631–644.

[11] Y. Diao, A. Keller, S. S. Parekh, and V. V. Marinov, "Predicting labor cost through it management complexity metrics," in *In Proceedings of the 10th IFIP/IEEE Symposium on Integrated Management*, 2007, pp. 274–283.

[12] Y. Diao and A. Keller, "Quantifying the complexity of it service management processes," in *DSOM*, 2006, pp. 61–73.

[13] L. Shwartz, Y. Diao, and G. Grabarnik, "Multi-tenant solution for it service management: A quantitative study of benefits," in *Integrated Network Management*, 2009, pp. 721–731.

[14] W. D. Gray, B. E. John, and M. E. Atwood, "Project ernestine: Validating a goms analysis for predicting and explaining real-world task performance," in *Human-Computer Interaction*, 1993.

[15] OGC, "Information technology infrastructure library v3 (itil v3)," Office of Government Commerce, May 2008. [Online]. Available: http://www.itil-officialsite.com/

Quality Improvement and Quantitative Modeling – Using Mashups for Human Error Prevention

Carlos Raniery P. dos Santos,
Lisandro Zambenedetti Granville
Institute of Informatics - UFRGS
Porto Alegre, RS, Brazil
Email: {crpsantos, granville}@inf.ufrgs.br

Larisa Shwartz, Nikos Anerousis
IBM T.J. Watson Research Center
Hawthorne, NY 10532, USA
Email: {lshwart, nikos}@us.ibm.com

David Loewenstern
WhitePages
New York, NY 10018, USA
Email: dloewenstern@whitepages.com

***Abstract*—Modern IT service provider organizations are under a continuous pressure to increase their competitiveness. Ways to reduce costs while improving performance – in terms of effectiveness, productivity, and quality – of services are a key focus area for companies in the IT industry. The existence of human operators in this industry, although required, may introduce defects in the process. In IT service provider organizations, preventing human errors from affecting the system is critical because of the strict requirements for quality. Our work in this paper is inspired by Six Sigma and is based on partial automation and process redesign to prevent human errors from occurring, or at least to reduce their frequency. In particular, we analyze the usage of mashups as an effective approach to cope with errors introduced by human operators while performing their daily activities in the context of IT Service Management (ITSM).**

I. Introduction

Modern IT service provider organizations are under a continuous pressure to increase their competitiveness. Ways to reduce costs while improving performance – in terms of effectiveness, productivity, and quality – of services are a key focus area for companies in the IT industry. However, despite all the solutions that have been proposed, modeling and optimizing human-centered processes remains a burdensome task. The human operator may be influenced by multiple factors and execute the process in a different way each time, thus introducing a significant variability in the final process outcome.

Automation is often used by the companies to obtain tight performance bounds [1]. However, that may not be feasible in some cases because of additional effort to deploy and maintain the automation infrastructure [2]. This is specially true when processes are constantly changing or are too complex to be automated. In such cases, the existence of human operators, although required, may introduce defects in the process. Previous researches have showed that human error is the largest contributor to reduced dependability in IT systems, and occur despite experience [3] [4] [5]. Even additional training and familiarity with systems cannot eliminate all the errors (*i.e.*, mistakes, slips, or lapses) made by the human operator. For example, a database administrator can accidentally delete a semicolon when executing consecutive commands, which may lead to a system outage or to a permanently damaged data.

In IT service provide organizations, preventing human errors from affecting the system – by avoidance or interception – is critical because of the strict requirements for quality. Errors in such environment introduce a considerable variance in quality, thus reflecting in severe penalties from Service Level Objective (SLO) violations. Variability is measured as the number of defective units at the output of the process, and is usually addressed by IT service quality engineers using techniques from the manufacturing domain. For example, SixSigma [6] and Lean [7] are two popular methodologies used in conjunction (*i.e.*, Lean Sigma [8]) to identify and remove the causes of defects and minimize the variability in both manufacturing and business processes.

In this paper we address the problem of reducing the occurrence of human errors in the Request Fulfillment process by using mashups technology. In the context of ITSM, mashups can be used to improve performance in human-centered processes through partial automation and process redesign. We propose a methodology inspired by SixSigma and based mashups to improve performance of ITSM processes by eliminating defects and reducing variability. In a previous work [9], we have showed that mashups are a feasible solution to tackle inefficiencies in ITSM processes, thus increasing the human operator productivity. Once more, we do not address exogenous elements, such as answering telephone calls. After all, we cannot change the human condition, but we can change the conditions under which humans work. In particular, we focus our work on error-prone activities performed by human operators that can be identified and quantified. The goal of our research is to analyze the usage of mashups as an effective approach to cope with errors introduced by human operators while performing their daily activities in the context of IT Service Management (ITSM).

The key contributions of our research work are:

- Introduce a comprehensive methodology to analyze error-prone activities in human-centered ITSM processes;
- Present a set of mashup patterns and interaction elements to cope with human errors;
- Provide a quantitative model to evaluate and predict the impact of employing such patterns in ITSM processes.

In order to reach these objectives, we focus on the Request

Fulfillment process, which is one of the operational processes in IT management and presents a high risk of human error. The proposed methodology uses the Human Error Assessment and Reduction Technique (HEART), which is widely used in the field of Human Reliability Assessment (HRA) for the purposes of evaluating the probability of a human error occurring throughout the completion of a specific task. Event Tree Analysis (ETA) is used for identifying and evaluating the overall probability of human error in the sequence of events performed during the Request Fulfillment process.

The remainder of this paper is organized as follows. In Section II we present the technical background related to this paper. In Section III we identify error prone activities in the context of ITSM and propose a methodology to account for those errors in Section IV. In Section V we introduce a set of mashup patterns and interaction elements for coping with human errors in ITSM processes. In Section VI we demonstrate the application of the proposed methodology on the request fulfillment process and present results of our case study. The paper concludes in Section VII, with a brief summary of our findings and an outline of future work.

II. Background

The objective of this section is to review the technical background of our proposal, which includes more in depth discussions of human error assessment and the mashup technology.

A. Mashups

Over the past few years, a new set of Web 2.0 applications have gained interest from both industry and academia. These applications, termed as mashups, are Web applications created through the integration of external resources available on the Web (*e.g.*, RSS feeds, Web services, and online APIs) [10]. Essentially, the main objective of mashups is to graphically combine and integrate disparate assets (*e.g.*, presentation, data, and functionality) in new ways. Unlikely traditional composition technologies (*e.g.*, BPEL and WSCI), mashups are focused on the end-users, allowing them to create their own applications and encouraging cooperation and reuse.

Mashups can be created following the methodology proposed by Jhingran et. al [11]. In this methodology, three main steps are defined: ingestion, augmentation, and presentation. The ingestion is the process of gathering data from heterogeneous sources (*e.g.*, Web services, RSS feeds, online APIs). During this process wrappers are often used to access and standardize the interested data. In the augmentation step, the retrieved data is integrated and transformed in order to create more meaningful information with purposes different from the original ones. During the presentation step, the result of the augmentation process is displayed in a Web browser to the human user. The presentation can range from a simple text file containing the results of the composition, to a complex and highly interactive map display.

Besides mashups can be created through traditional programming techniques, specific systems (*i.e.*, mashup systems) can be employed by users with no programming expertise during the creation process. Such systems are Web-based development tools that enable end-users to create their own mashups; they are what an Integrated Development Environment (IDE) is to a programming language. Usually, mashup systems perform three main tasks: provide users with visual tools to define new mashups, store created mashups in mashup repositories or libraries, and execute mashups when requested. The three tasks are often performed from a single mashup system, but it is possible for a mashup to be created in one system, executed in another, and stored in a third to form a remote mashup repository.

In the context of ITSM, mashups can be used to improve performance in human-centered processes. We have ourselves [9] introduced a set of mashup patterns to address suboptimal execution of activities, providing a significantly improved orchestration of the process, thus improving the productivity. Further productivity, mashups can also be applied to decrease the number of errors introduced by human operators. The main concepts under the human error area and the technical background related are presented in the next subsection.

B. Human Errors

Human beings have the ability to execute multiple and complex tasks (mental and physical) at the same time. However, although skills and expertise level can vary among people, all humans reach their natural limits. When such limits are reached or exceeded, humans become susceptible to making errors.

Several works are found in the literature aiming to define ”human error”. However, up to now, there is no agreement on a unique definition for the term. In this paper, we assume the definition proposed by Reason [12], who considers that erroneous actions include all situations in which a planned sequence of physical or mental activities failed to obtain a result and whose errors cannot be attributed to interventions of external causes. Reason’s work is based on the Skill, Rule, Knowledge (SRK) based approach [13], which introduces a classification for errors common to all professions (*e.g.*, aviation, military, nuclear power, health care). These errors are of the following types:

1) **Knowledge-based errors** - Occur because of a deficit of knowledge. For example, a person may want to perform some plan or action, but the plan or action is implemented partially or incorrectly because of a lack of knowledge, thus not achieving the proper result;
2) **Rule-based errors** - Occur when a well-known rule is wrongly used, or a situation is misinterpreted. For example, when a person using a particular operating system is forced to use a different one, and then tries to perform the same actions even if the Graphical User Interface (GUI) is different;
3) **Skill-based errors** - Occur when actions are divergent to the original intentions. For example, when an user performs an automatic action, with little attention, and forgets to execute a specific task;

Reason [12] also introduced the Generic Error Modeling System (GEMS), which is an extension of the SRK approach. In that work, Reason observed that the type of errors can be

either involuntary or intentional. Involuntary actions (*i.e.*, slips and lapses) are those that deviate from planned intentions and, thus, do not reach their goals. Intentional actions (*i.e.*, mistakes and violations), on the other hand, are performed consciously but the desired result is not achieved. Figure 1 presents GEMS human error taxonomy.

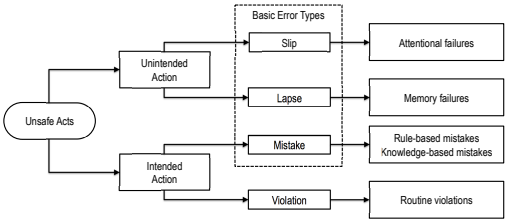

Fig. 1. Generic Error Modeling System (GEMS)

Previous investigations found that 61% of the human errors are skill-based, which means that humans are prone to slip and lapse with familiar tasks. When such tasks become harder, rule-based (28%) and knowledge-based (11%) become the most common failures made by humans. In the context of ITSM, human operators are well trained in specific areas of expertise and mostly of the work include routine tasks (*e.g.*, dispatching service requests, configure database server, restore password). In this paper, we focus on such skill-based errors, which can be tackled through proper instrumentation.

C. Quantitative Modeling for Human-Error Assessment

Diverse techniques to account for human errors have been developed in the last few decades, such as Technique for Human Error Rate Prediction (THERP) [14], Justification of Human Error Data Information (JHEDI), and Human Error Assessment and Reduction Technique (HEART) [15]. Although such techniques might appear to be outdated, enhanced methods were not developed in the field of Human Reliability Assessment (HRA) [16]. These techniques are still used in diverse high-risk professions such as nuclear power, healthcare, and aviation. Previous investigations [17] [18] [19] have shown that all these techniques present a very significant correlation between their predictions and the corresponding real measurements of human errors.

In this paper, we quantify human reliability using HEART, which is considered the most comprehensive method in the field of HRA. HEART method does not require the use of real human error measurements to make the predictions; instead, it specifies 9 Generic Task Types (GTTs) and provides the nominal Human Error Probability (HEP) for each one. HEART also specifies 38 Error Producing Conditions (EPCs), which are factors that can affect human performance making it less reliable (*e.g.*, distractions, overload). HEART methodology is based on the following four steps:

1) Identify all sub-tasks a system operator would be required to execute in order to complete an specific activity;
2) Identify the generic task type corresponding to the activity and its HEP;
3) Select the perceptible EPCs in the activity and that have a negative effect on the outcome;
4) Calculate the final HEP using the equation 1.

$$HEP[final] = HEP * \left\{ \prod_{i=1}^{n} [(EPC_i - 1)\, Ap_i + 1] \right\} \quad (1)$$

In this equation 1, $HEP[final]$ is the final probability of human error for an specific activity, HEP is the nominal human unreliability accordingly to the task type, EPC_i is the ith error-promoting condition, and Ap_i is the proportion assessment factor (from 0 to 1) defined by the analyst. Because of space constraints, we suggest the interested reader to refer to the work of Williams [20] in order to observe HEP and EPC values in more details.

III. Human Errors in Service Management Processes

Continuous pressure on IT companies to increase their competitiveness forces human operators to reach or exceed their natural limits. Such constant effort, poorly designed systems, and lack of experience are examples why the human operators make errors during their daily activities [21]. In particular, we address in this paper error-prone activities, *i.e.*, portions of an IT Service Management (ITSM) process characterized by high risk of human error, that can be measured through instrumentation or observation, and can be improved through design and automation. Errors in ITSM processes can occur due to a wide variety of causes, for example:

- *Attention Span*: As the number and complexity of tasks increase, our ability to focus and maintain our attention decreases;
- *Memory*: Our short term (or working) memory is limited to the equivalent of retaining around seven distinct items at a time. This may not be sufficient for active processing of some complex tasks;
- *Situation Awareness*: A person's awareness (or not) and perception of different elements in their environment affect their information processing and actions;
- *Personal Resources*: the ability to process information appropriately diminishes as stress and fatigue increase.

In order to discover common error-prone activities in ITSM, we have analyzed tasks performed by a group of Subject Matter Experts (SMEs) in a very large IT Service Support and Delivery organization. Based on documents related to service incidents, it was possible to identify the non-exhaustive list of human errors presented below, presented in rough order from lowest to highest level. We illustrate these errors using two scenarios: a change management scenario in which a dispatcher assigns change requests to the appropriate technician, who is responsible for resolving them; and a continuity management scenario in which an analyst designs a disaster-recovery plan which will then be tested manually.

- *Action Errors* are associated with actions performed by an operator on an object (*i.e.*, element of interest) and that change the state of the system. For example: the change request contains correct information, but the technician implements the change on the wrong server; the change request requires two technicians to operate in order (first reinstall the operating system, then install the data base) but the actions are performed out of order; the business recovery plan is complete but the tester fails to implement some part of the plan, causing the test to fail.
- *Retrieval Errors* are failures to retrieve correct information, either from memory or from a visual display, to be used in a further step. For example: the technician retrieves and acts upon a change request that had been assigned to a different technician; a business continuity analyst does not notice a list of required static IP addresses in the client configuration and so fails to include them in the business recovery plan.
- *Checking Errors* are errors that occur when the operator fails to check some information. It can be the combination of action and retrieval errors. For example: the technician marks the wrong change request as having been resolved.
- *Decision Errors* relate to the reasoning capabilities or training of the human operator and occur when the operator has to make an explicit choice between multiple alternatives. For example, a dispatcher assigns a change request to a technician who does not have the appropriate skill level for the task; an analyst suggests an unsuitable hardware substitution in a recovery plan.
- *Communication Errors* occur when the operator fails to pass information to another person, either in person or through a system (*e.g.*, instant messenger, e-mail). For example: a dispatcher fails to inform a technician of a change request that had been assigned to him or her;

Where it is feasible, companies can reduce human errors through complete automation of the process. However, even where this is not possible, it is very often the case that the process may be partially automated, using software to alert the operator to potential errors or provide support to reduce task complexity [22] [23]. The next section presents a methodology for locating error-prone components of ITSM processes and applying partial automation in the form of mashups to reduce the potential for human error.

IV. Methodology for Quality Assessment

We combine Six Sigma with various techniques to improve the performance of ITSM processes by eliminating defects and reducing variation. The HEART technique, for example, was extensively validated by experiment in other fields and represent one of the most widely used technique for Human Reliability Assessment (HRA) [17] [18] [19]. Event Tree analysis, in turn, is mainly used in the field of safety engineering and reliability engineering and can be used combined HEART to determine the probability of failure in a sequence of events. Finally, mashups represent an effective approach for eliminating root cause problems that degrade the quality in ITSM processes. Our methodology is based in the following stages:

1) The analyst works with a Subject Matter Expert (SME) to determine the process workflow and root causes of problems;
2) Using HEART technique the analyst determines the Human Error Probability (HEP) of each activity in the workflow;
3) Event Tree analysis is used to evaluate the performance of overall workflow;
4) The analyst identifies the most appropriate Mashup Patterns and interaction elements to solve problems;
5) The analyst evaluates the process once more to ensure improvements become embedded to the process.

Based on the obtained process workflow (stage 1) and on the individual HEP values for each activity (stage 2), the analyst builds an Event Tree (stage 3) to identify and evaluate the sequence of events in a failure scenario. Once the most error-prone activities are identified, the analyst selects the most appropriate mashup patterns to improve the process (stage 4). Finally, the analyst run through the methodology again (stage 5), from stage 2 to stage 4, in order to ensure the improvements become embedded, and to detect new improvement areas in the process.

Besides its simplicity and easy of application, the HEART technique presents several problems. For example, the Error Producing Conditions (EPCs) are not independent of each other; moreover, the use of the method is extremely subjective and relies heavily on the experience of the analyst; and finally, from a qualitative point of view, the EPCs are a useful list of factors to guide quality analysts, but the numerical values (*i.e.*, Assessed Proportion of Affect) are subjective and imprecise. In order to take into consideration such weaknesses, we have analyzed the values of 2024 EPCs chosen by Subject Matter Experts (SMEs) in 8 common ITSM activities.

TABLE I. Assessed Proportion of Affect

Class	Average	σ
Low	0.1833	0.086157
Medium	0.5032	0.106578
High	0.8316	0.077288

A clustering algorithm, called k-means, was employed to group these EPCs' values in clusters. k-means is a partitioning technique that groups a dataset into k clusters. The criterion used to assess the number of clusters is the average silhouette of the data. Three significant clusters – *i.e.*, low (L), medium (M), and high (H) – have been identified. For each cluster we have measured the EPC value average, which will be used as input to expression 1. In Table I, we show these averaged results and the respective standard deviation σ. We employed "linguistic variable" to represent each cluster because it is very useful when situations are too complex of ill-defined to reasonably described in conventional quantitative expressions.

V. Copying Human-Errors in ITSM Processes Through Mashups

As previously discussed, although required, the existence of human operators in ITSM processes can introduce a significant variability in the final process outcome. These operators execute multiple and complex tasks, usually employing different systems, and them becoming susceptible to making errors. In our previous work [9], we have introduced a set of mashup patterns for eliminating inefficiencies in ITSM processes, thus improving the human operator's productivity. Such inefficiencies are caused by insufficient design of the process itself, or defects in the tools being used. We argue that these mashups patterns can be used to cope with human errors too. In special interest of this paper, we analyse involuntary actions (*i.e.*, slips and lapses) performed by human operators and that can be tackled through instrumentation and redesign. The list bellow summarizes the mashup patterns we have introduced before.

- **Alerter pattern**: periodically monitors a system of interest on behalf of the user and, based on previously established conditions, sends notifications only when events of interest take place;
- **Importer pattern**: abstracts the different methods used to access the external data so that data consistency maintenance becomes transparent to the user;
- **Transformer pattern**: enable the processing of certain types of data and thus both materializing the compatibility between systems and satisfying the requirements of the IT process;
- **Displayer pattern**: enforces the concept of a "single pane of glass", allowing the combination of data from multiple sources and present the results of this combination in a Web page.

In order to prevent human errors from occurring, or at least to reduce their frequency, we have created a new type of interaction elements, called **Error Prevention** modules, to be used by mashup developers. These interaction elements are used to hide technical details from the mashup developer [24]. We define two types of this **Error Prevention** module: *Buffer* module and *Forcing* module. The *Buffer* module allows developers to specify for how long time a specific action should be delayed before being executed, thus providing a recovery window during which an erroneous action can be cancelled. According to Reason [12], roughly 78% of human errors can be detected immediately after they are executed. The *Forcing* module, on the other hand, allows developers to introduce confirmation points in the process workflow in order to force the human operator to consciously accept them before proceeding with the mashup execution.

In Figure 2, we provide an example illustrating how the **Error Prevention** modules are used in the creation of a mashup application. In Step (1) two adaptation modules are used to retrieve information from external data sources, which is merged In Step (2). The result of this composition is sent to a third system using another adaptation module (5). *Buffer* and *Forcing* modules are used (Step (3) and Step (4)) to first, force the human operator to confirm if the mashup should continue its execution, and second to wait for an amount of time before sending the merged data to the external system.

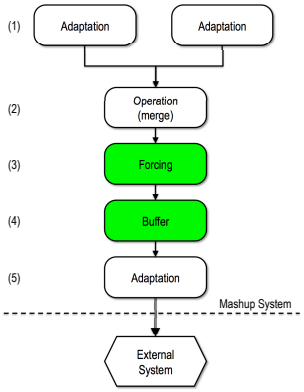

Fig. 2. Usage Example of Error Prevention Modules

VI. Case Study and Evaluation

We concentrate on the Request Fulfillment process, one of the operational processes in IT Management, responsible for carrying out service requests, and that interfaces with Service Desk, Incident Management, and Change Management. ITIL v3 describes it as "management of customer or user requests that are not generated as an incident from an unexpected service delay or disruption" [25]. This process includes two main activities: support the requests (*i.e.*, tickets) made by the customers, and solve those requests. The first activity is performed by system administrators (SAs) responsible for solving specific requests; the second activity is executed by dispatchers, who are human operators responsible for monitoring new requests, dispatching requests to the right SAs, and monitoring compliance with Service Level Agreements (SLAs).

A. Dispatch Activity

In this paper we focus our analysis on dispatch in Request Fulfillment, an activity centered on humans with knowledge of standard fulfillment procedures. Once the customer creates a new request (*i.e.*, incident, problem or change request) in Service Desk, a ticket is routed to a dispatcher who is responsible for analyzing the ticket and determining the appropriate SA to solve it. Usually, each dispatcher is responsible for teams of system administrators, each with a specialized technical background. Once the dispatcher receives the ticket, he analyses if it was misrouted or not. To accomplish this task, dispatchers determine if his team is able to resolve it (using his knowledge of his team's schedule, workload and expertise). If his team is not able to solve the ticket, he forwards it to another dispatcher who is responsible for a different team of SAs.

Although fully automated solutions offer many advantages, they may not be feasible for the dispatch activity. Dispatch is

constantly changing and is too complex to be automated. For example, in some cases a dispatcher may want to train a new administrator, and so may intentionally assign a request to a less skilled SA than is available. Although required, the use of human dispatchers in service delivery centers may introduce defects in the process, thus reflecting in severe penalties from Service Level Objective (SLO) violations. In this context, mashups are a new technology that simplifies and makes more predictable the creation and execution of dispatching systems, focusing on the activities performed by each dispatcher, thus decreasing the possibility of errors.

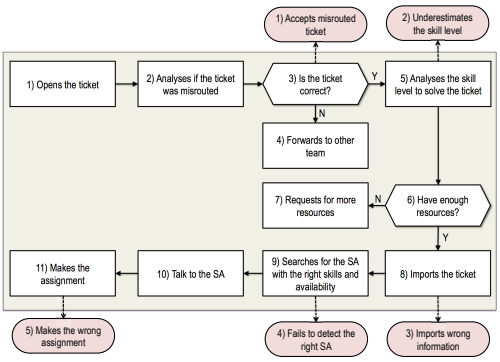

Fig. 3. Workflow of activities performed by dispatchers in Service Desk. Ovals represent points of failure.

In order to detect problems and perform a root cause analysis in dispatch process, we have shadowed dispatchers in their daily activities. Based on interviews, was possible to determine the workflow of activities presented in Figure 3. Once a new request is received from a customer, the dispatcher determines if his team has the proper expertise to solve it, *i.e.*, he analyses if the ticket was misrouted. If the ticket was not misrouted, the dispatcher analyses the required skill level to solve it. Next, the dispatcher imports data from the customer's ticketing system. This activity is usually performed manually because data usually need to undergo some processing (*e.g.*, anonymising confidential information). Once the ticket is imported, the dispatcher uses SA availability, workload and skills to determine the proper SA to solve it.

In order to accomplish his tasks, dispatchers interact with multiple tools (*e.g.*, multiple ticketing systems, calendar-based systems, spreadsheets). Furthermore, dispatchers may also be overloaded with customer requests, which can vary from just a few dozen to hundreds in a busy day. Due this complex nature and continuous pressure, dispatchers are vulnerable to making errors. Based on the interviews and on documents related to service incidents, it was possible to identify the most common failures in dispatch, presented in Figure 3. These failures may occur due the combination of the error types we have depicted in section III. The dispatcher may, for example, accept a misrouted ticket. This can happen due to a decision error, when he fails to decide if the ticket was misrouted; or due an action error, when he mistakenly clicks on the accept button instead of rejecting it. The skill level to solve the ticket can also be underestimated, characterized as a decision error. Retrieval errors may also be introduced during the importing step. Another example of a decision error occurs when the dispatcher fails to detect the right SA to solve the ticket. Finally, even if the dispatcher knows the proper SA, he may fail to make the assignment because selects the wrong SA in the ticketing system (*i.e.*, an action error).

B. Baseline Performance

In order to discover the human error probability (*i.e.*, $HEP[final]$) of each task performed by dispatchers, we have classified them as type G. According to HEART, the generic task type G involves "completely familiar, well designed, highly practised, routine task occurring several times per hour, performed at the highest possible standards by highly motivated, highly trained and experienced person, totally aware of the implications of failure, with time to correct potential error, but without the benefit of significant job aids". This generic task type has a nominal human error probability of 0.0004. For each task, we also have selected the EPCs that present a negative effect on the outcome and them calculated the $HEP[final]$ according to equation 1. We illustrate in Table II the HEART calculations for task 3 (*i.e.*, import wrong information about the ticket).

TABLE II. HEART CALCULATIONS FOR TASK 3

EPC	HEART effect	Assessed proportion of effect	Assessed effect
Suppressing information	x 9	High	7.48
Channel overload	x 6	High	4.99
Little checking	x 3	Medium	1.51

Based on these values, the total assessed EPC effect is $(7.48 * 4.99 * 1.51) = 56.36$ and the assessed human error probability is $(0.0004) * 56.36 = 0.022$. Therefore the probability of occurring a human error in Task 3 is 2.2%.

TABLE III. ORIGINAL FAILURE PROBABILITIES

i	Task	$HEP[final_i]$
1	3	0.002391857
2	5	0.019094864
3	8	0.022549899
4	9	0.019094864
5	11	0.002391857

By following the HEART methodology for the remaining tasks, we were able to get the failure probabilities presented in Table III. We do not present tasks 4 and 7 because in our scenario, the tickets were not routed to the wrong dispatcher and we also determined that he always had enough resources (*i.e.*, System Administrators) to solve them. Task 10 is not shown because it was not observed during our evaluations. With these values it was possible to employ the Fault Tree Analysis (FTA) method in order to estimate the overall workflow failure rate. This failure rate was obtained using the equation 2, presented bellow.

$$P[failure] = 1 - \prod_{i=1}^{n}(1 - HEP[final_i]) \quad (2)$$

Let $P[failure]$ be the probability of failure in a sequence of events and $HEP[final_i]$ be the final probability of human error for a specific task, which is obtained from equation 1. By taking the equation 2, we obtain $P[failure] = 0.064015658$, which means that there is a chance of 6.5% for the dispatcher to make at least one error during the execution of the dispatch process.

C. Applying Mashup Patterns and Error Prevention Modules to Dispatch

All the mashups patterns proposed in our previous work and the error prevention modules introduced in this paper can be used to create a mashup-based solution for the above-mentioned dispatching scenario. By using these patterns and modules, we aim to eliminate, or at least reduce, the probability of dispatchers making errors during their daily activities. Figure 4 shows how the mashup patterns and proposed error prevention modules relate with each specific task in the dispatching scenario.

The displayer pattern is used to show in a single screen all the useful information for the dispatcher (*e.g.*, workload, skills, ticket description). This pattern aids to reduce the occurrence of retrieval errors, either from memory or from a visual display. The combined use of both importer and transformer patterns is useful to avoid action and retrieval errors. Even with the introduction of these patterns, the dispatcher may still make some error. In order to reduce this probability even more, the two error prevention modules introduced in this paper are used after the assignment task (11). If the dispatcher mistakenly selects the wrong SA to assign the ticket, the system will require a confirmation to execute the assignment task. Even if the dispatcher confirms that, he still has a few seconds to cancel the operation before it is executed.

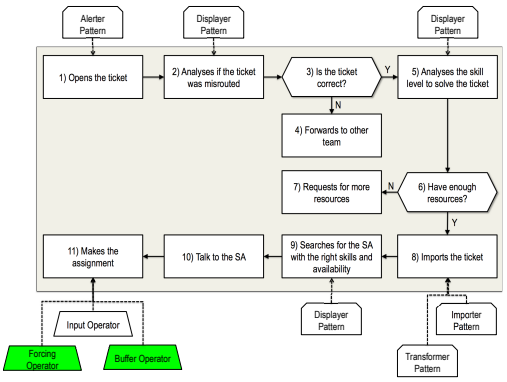

Fig. 4. Relating mashup patterns and operators with dispatching tasks

D. Predictions on Mashups Usage

Analyzing the tasks of the dispatching scenario and using the methodology described in section IV we estimated the human error probabilities presented in Table IV. Each line of this table represents a specific task of the dispatching process. Task 8 was completely automated by employing mashup patterns, thus the human error probability is 0%.

TABLE IV. FAILURE

i	Task	$HEP[final_i]$
1	3	0.00091324
2	5	0.00445674
3	8	0.00000000
4	9	0.00476030
5	11	0.00091324

From equation 2, we obtain $P[failure] = 0.011004691$, which means that the use of mashups can reduce to 1.1% the chance of dispatchers to make at least one error during the execution of the dispatch process. The obtained result show the significant reduction of 83.07% in the probability of human error, which mostly was due to the automation of step 8 by using the importer pattern. The usage of a displayer pattern contributed to reducing the chance of failure in tasks 5 and 9. In both tasks, the dispatcher no longer needs to look up for any information in a different system, thus eliminating all the possible retrieval errors.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we extended our previous research on applying mashups to improve the performance of ITSM processes by eliminating defects and reducing variability. We have introduced a methodology inspired on Six Sigma and based on partial automation and process redesign to tackle error-prone activities in human-centered ITSM processes. HEART and Event Tree Analysis were combined to evaluate and predict the impact of employing mashups in ITSM processes. We have focused our analysis on dispatch in Request Fulfillment, an activity centered on humans with knowledge of standard fulfillment procedures. Based on interviews with a group of SMEs in a very large IT Service Support and Delivery organization was possible to determine the workflow of activities performed and the most common failures in dispatch.

The new interaction elements, *Buffer* and *Forcing*, in conjunction with a set of mashup patterns enable mashup developers to create new applications that ultimately prevent human errors from occurring, or at least reduce their frequency. Two ways to prevent human errors were observed: avoidance and interception. Error avoidance (*i.e.*, keeping people from making errors) was possible through the use of mashup patterns and the *Forcing* module. Error interception (*i.e.*, stopping the errors from reaching the system) was possible through the *Buffer* module. Altough was not observed the reduction of Knowledge-based errors (*i.e.*, decision errors), the presented patterns and interaction elements avoided the occurrence of Rule-based and Skill-based errors (*i.e.*, action and retrieval errors), which represent 89% of human errors according to previous investigations. We also demonstrated the significant

reduction of 83.07% in the probability of dispatchers making errors during their daily activities.

As future research, we aim to expand our exploration of performance in business processes by evaluating the correlation of productivity and quality. We also plan to extend the analysis of the ITSM process beyond productivity and quality, considering other aspects such as effectiveness. Finally, we will explore the influence of these three aspects in ITSM process costs.

REFERENCES

[1] A. Keller, J. L. Hellerstein, J. L. Wolf, K. L. Wu, and V. Krishnan, in *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, vol. 1, 2004, pp. 395–408.

[2] A. B. Brown and J. L. Hellerstein, "Reducing the cost of it operations: is automation always the answer?" in *Proceedings of the 10th conference on Hot Topics in Operating Systems - Volume 10*, ser. HOTOS'05. Berkeley, CA, USA: USENIX Association, 2005, pp. 12–12.

[3] T. Benson, S. Sahu, A. Akella, and A. Shaikh, "A first look at problems in the cloud," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, ser. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 15–15.

[4] J. Gray, "Why do computers stop and what can be done about it?" in *Symposium on Reliability in Distributed Software and Database Systems*, 1986, pp. 3–12.

[5] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, "Why do internet services fail, and what can be done about it?" in *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems - Volume 4*, ser. USITS'03. Berkeley, CA, USA: USENIX Association, 2003, pp. 1–1.

[6] P. Pande, R. Neuman, and R. Cavanagh, *The Six Sigma Way: How GE, Motorola, and Other Top Companies are Honing Their Performance*. McGraw-Hill, 2000.

[7] J. Krafcik, "Triumph of the lean production system," *Sloan Management Review*, vol. 30, no. 1, pp. 44–52, 1988.

[8] T. T. Allen, *Introduction to engineering statistics and lean sigma: statistical quality control and design of experiments and systems*. London: Springer, 2010.

[9] C. dos Santos, L. Granville, W. Cheng, D. Loewenstern, L. Shwartz, and N. Anerousis, "Performance management and quantitative modeling of it service processes using mashup patterns," in *Network and Service Management (CNSM), 2011 7th International Conference on*, oct. 2011.

[10] D. Merril, "Mashups: The new breed of web app - an introduction to mashups." 2003, http://www.ibm.com/developerworks/web/library/x-mashups.html.

[11] A. Jhingran, "Enterprise information mashups: integrating information, simply," in *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment, 2006, pp. 3–4.

[12] J. Reason, *Human Error*. Cambridge [England] ; New York : Cambridge University Press, 1990. xv, 302 p., 1990.

[13] J. Rasmussen, "Skills, rules, and knowledge: signals, signs, and symbols, and other distinctions in human performance models," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 13, no. 3, pp. 257–266, 1983.

[14] A. Swain and H. Gutterman, "Handbook of human reliability analysis with emphasis on nuclear power plant applications," Sandia Laboratories, Albuquerque, New Mexico, Tech. Rep. NUREG/CR-1278, 1980.

[15] J. Williams, "Heart - a proposed method for achieving high reliability," in *Symposium on the Achievement of Reliability in Operating Plant, Safety and Reliability Society*, 1985, pp. 87–109.

[16] R. Whittingham, *The Blame Machine: Why Human Error Causes Accidents*. Taylor & Francis, 2012.

[17] B. Kirwan, "The validation of three human reliability quantification techniques therp, heart and jhedi: Part 1 -ttechnique descriptions and validation issues," *Applied Ergonomics*, vol. 27, no. 6, pp. 359 – 373, 1996.

[18] B. Kirwan, R. Kennedy, S. Taylor-Adams, and B. Lambert, "The validation of three human reliability quantification techniques therp, heart and jhedi: Part 2 - results of validation exercise," *Applied Ergonomics*, vol. 28, no. 1, pp. 17 – 25, 1997.

[19] B. Kirwan, "The validation of three human reliability quantification techniques therp, heart and jhedi: Part 3 - practical aspects of the usage of the techniques," *Applied Ergonomics*, vol. 28, no. 1, pp. 27 – 39, 1997.

[20] J. Williams, "Validation of human reliability assessment techniques," *Reliability Engineering*, vol. 11, no. 3, pp. 149 – 162, 1985.

[21] A. B. Brown, "Oops! coping with human error in it systems," *Queue*, vol. 2, no. 8, pp. 34–41, Nov. 2004.

[22] V. R. Madduri, M. Gupta, P. De, and V. Anand, "Towards mitigating human errors in it change management process," in *ICSOC*, 2010, pp. 657–662.

[23] L. Shwartz, D. Rosu, D. Loewenstern, M. J. Buco, S. Guo, R. Lavrado, M. Gupta, P. De, V. R. Madduri, and J. K. Singh, "Quality of it service delivery - analysis and framework for human error prevention," in *SOCA*, 2010, pp. 1–8.

[24] C. dos Santos, R. Bezerra, J. Ceron, L. Granville, and L. Rockenbach Tarouco, "On using mashups for composing network management applications," *Communications Magazine, IEEE*, vol. 48, no. 12, pp. 112 –122, december 2010.

[25] OGC, "Information technology infrastructure library v3 (itil v3)," Office of Government Commerce, May 2008. [Online]. Available: http://www.itil-officialsite.com/