International Journal on Advances in Internet Technology

2009 vol. 2 nr. 4

The *International Journal on Advances in Internet Technology* is published by IARIA.
ISSN: 1942-2652
journals site: http://www.iariajournals.org
contact: petre@iaria.org

International Journal on Advances in Internet Technology
Volume 2, Number 4, 2009

International Journal on Advances in Internet Technology
Volume 2, Number 4, 2009

CONTENTS

Securing e-Businesses that use Web Services — A Guided Tour Through BOF4WSS

Jason R. C. Nurse and Jane E. Sinclair
University of Warwick, Coventry, CV4 7AL, UK
{jnurse, jane.sinclair}@dcs.warwick.ac.uk

Abstract

Security in Web services technology itself is a complex and very current issue. When considering the use of this technology suite to support interacting e-businesses, literature has shown that the challenge of achieving security becomes even more elusive. This is particularly true with regard to achieving a level of security beyond just technologies, that is trusted, endorsed and practiced by all businesses involved. In an attempt to address these problems, our research has previously introduced BOF4WSS [1], a business-oriented development methodology, specifically geared to guide e-businesses in defining, and achieving agreed security levels across collaborating enterprises. As that work was only an introduction, the aim of this paper is to provide detailed insight into what exactly BOF4WSS advocates and how these activities and processes aid in building security and trust.

Keywords: Security, Web services, e-business, systems development methodology, cross-enterprise interactions

1. Introduction

E-business has become the fastest growing means of conducting business in today's economy. In achieving the online business-to-business (B2B) collaboration between e-businesses, the use of services-oriented computing, by way of Web services (WS) technology, is playing an increasingly significant role [2]. The novel benefit is rooted in its ability to allow for seamless integration of business processes across disparate enterprises. This is due to the use of standardized protocols and open technologies [3]. One author [4] even states that the facilitation and automation of these business processes is the ultimate goal of Web services. As WS' use expands however, securing these services becomes of utmost importance.

In an attempt to address new security challenges accompanying WS, standard-setting bodies have proposed numerous pioneering standards. As WS matures, the move from lower level security details such as standards and technologies, to higher level considerations however, is imminent [5]. Security, irrespective of the context, is a multilayered phenomenon encompassing aspects such as practices, processes and methodologies. This factor is especially true with WS which, as authors [6] note, substantially complicates the security environment for e-businesses.

Considering this, and with special appreciation of the inter-organizational security issue now facing businesses interacting using WS, in previous work we have introduced the Business-Oriented Framework for enhancing Web Services Security for e-business (BOF4WSS) [1] to address some of these issues. At its core, this framework supplies a cross-enterprise development methodology that can be used by businesses—in a joint manner—to manage the comprehensive concern that security in the WS environment has become. Building on the introduction to BOF4WSS given in that work therefore, this paper presents thorough coverage of the framework, its ideas, the tasks involved, and also their justifications.

The remainder of this extended paper is structured as follows: Section 2 contains a brief review of the security advancements in WS use for e-business with the aim of identifying outstanding security issues, and therefore paving the way for BOF4WSS. Next in Section 3, a detailed discussion of the framework, including its novelty and use, is given. Conclusions and future work will be outlined in Section 4.

2. Web Services Security within e-Business

2.1. State of the Art

Albeit a promising implementation technology for the Service-Oriented Architecture (SOA), and an increasingly used enabler of e-business, WS comes at a high price of an unstable security foundation. The literature identifies numerous challenges [5], [7], but the most pertinent for our research is the reality that WS adds significant complexity to the e-business security landscape [6]. This complexity makes security a much broader and comprehensive concern, which cuts across business lines much easier and quicker than before. As such, an inadequate security posture in one company can mean an increased, real-time security risk for its partners—both immediate and extended.

To address the new security challenges mentioned above, consortiums such as OASIS and W3C have developed and ratified numerous pioneering standards (as can be seen in [5]). These standards aim to both solve problems caused by common threats and also to further the WS paradigm by enabling substantially more dynamic security interactions between services. Beyond addressing the perceived inadequacies of the current standards base, researchers are now targeting the more general components of a security solution such as best practices and processes. These actions give life to a prediction made by NIST, which emphasized that as WS technology matured, methodologies and recommended practices for security would become the next step in the goal of developing secure systems [5].

Some of the most pertinent, and noteworthy proposals focusing on these higher layers are: [8], which builds on existing technologies and the theory of Aspect-Oriented Programming, to provide a framework for securing WS compositions (necessary in collaborative e-business) using the WS-Security and WS-Policy standards; [9] aims to provide a methodical development approach for constructing security architectures for WS-based systems; [10] which provides integrated WS design strategies and best practices for end-to-end security; [11] – a method that uses fuzzy logic to measure the risk associated with WS, with full appreciation of the fact that due to WS' volatility, information on threats is usually incomplete or imprecise; and lastly the Event-driven Framework for Service Oriented Computing in [12] – a standard agnostic, multilayered framework that aims to address the problem of defining and enforcing access control rules for securing services use at the level of business processes. In their work, authors particularly focus on dynamic authorization, independent of specific standards [12].

2.2. Outstanding Security Issues

WS security approaches should aim to be thorough in planning, developing and maintaining an adequate solution. Standard security components encompass technologies, but as recent literature [13] in the study of security in general has emphasized, it also includes policies, processes, methodologies, and best practices. To WS' detriment however, this fact does not appear to be unanimously shared as any attention on these other aspects is being drowned out by the proliferation of various new technology standards. One can easily see this fact when comparing the few higher layer approaches mentioned in [1] to the vast number of standards and technical systems highlighted in [5]. It may therefore be very tempting to regard such mechanisms as the 'solutions' to the WS security problem. Whilst the work of technologists is valuable to building security and trust however, alone they cannot form the entire solution. In fact, all these mechanisms address is the technology layer of security, and the threats which emanate at that level; thus only providing a stepping-stone in the goal of reliable, comprehensive, multilayered security. This perspective is supported by authors in [5] and they identify processes such as effective risk management, and defence-in-depth through security engineering, as critical to developing robust, secure systems.

A final concern regarding standards is that there are already too many available [14]. Therefore, as opposed to benefiting WS, this plethora of sometimes overlapping standards ultimately confuses developers and acts to complicate secure WS implementation and use. The importance of these factors is magnified when assessing WS use for the already complex field of e-business.

To briefly assess the aforementioned research in [8], [9], [10], [11], these are all seen to successfully complement available technologies, and provide useful security approaches. Their main caveat however is that they consider security predominantly from one company's internal viewpoint, i.e., what should a company do internally to secure itself. This highly isolated perspective is inadequate due to the very nature of WS, and the high degrees of interconnection between businesses—spanning exposure of legacy systems to purpose-built Web applications—that WS readily facilitates. In [12], even though this allows for a layered, and more comprehensive model for WS security during business process execution, its predominant focus is towards access control, and particularly for highly dynamic environments. Both these aspects act to make it too specific a framework for our purposes as mentioned before.

Looking beyond these advancements, an intriguing research area which has received little emphasis is at the level of *cross-enterprise interaction* (i.e. interactions spanning, and including collaborating businesses and their internal systems). Specifically, we refer to providing some comprehensive approach to aid businesses in collectively handling security as the broad, inter-organizational concern it has become. This approach would not be solely at the technical level but look generally at a number of other fundamental aspects (e.g. security directives, policies, government regulations, best practice security standards, business risk considerations, and negotiations necessary) that businesses should jointly consider when developing and engaging in B2B interactions employing WS. This is particularly with the knowledge that in WS, lack of security in one business can very easily mean elevated security risk for a partnering entity, its systems and its data [6].

The basic notion behind such a proposal can been seen in the largely exploratory research study done in [15]. In that article, the authors accepted the comprehensive security dilemma e-businesses face and proposed a generic model to enhance security. In many respects our research's general proposals are an extension of that exploratory work, to delve into the intricacies of what would constitute such a comprehensive security approach.

Further to the previously mentioned goals, this new approach would also aim towards facilitating the increased trust in business partners, their systems, and the overall service interactions, as an intrinsic objective. The importance of trust in e-business (with or without WS) is stressed by several authors [16] and at the risk of oversimplifying its elusive nature, some of its most salient attributes in this context are transparency, accountability, predictability, reliability and benevolence [17], [18]. This approach would aim to foster trust between partners, their systems (which are no longer 'black boxes' to partners), and the overall service interactions, by stressing these and related factors.

With regards to security and trust in general, the approach could be seen to facilitate a level of confidence in services and partners not obtainable if businesses integrate security merely at the technology level. Technology-level integration, even though essential, is only part of the complete security solution. In discussing the general topic of WS' usage for B2B, Alonso et al. [19] note that WS enables "a company to open its IT infrastructure to external partners" however it does "not help with the many legal and contractual issues involved in B2B interactions". Similarly, technology-level security integration can be done, but to allow for a more holistic security solution in B2B—and particularly in businesses which have cross-enterprise security as a critical goal—other higher level aspects must be considered. These aspects go beyond the flashiness of dynamic security and trust negotiation possible with WS standards, and deal with a business-level security approach to risks and each organization's needs and goals. Typical areas in which cross-enterprise security might be a such an important goal would be businesses with substantial and long-term investments. Also, companies bound to strict contractual or government regulations that must be enforced. And lastly, businesses that deal with mission-critical systems, such as the health or banking sectors.

In summary, there are a number of unaddressed issues as e-businesses look towards creating, and maintaining a comprehensive, trustworthy WS security solution. Primarily these stem from (i) an overly reliant emphasis on technology, alluding to standards and systems as the complete solution to WS security, and (ii) an overly isolated security stance, focusing on the process *one* company should follow to secure itself internally, therefore ignoring the comprehensive security issue introduced by WS use. As was previously mentioned in Section 1, to address these issues, BOF4WSS was proposed. The goal of the next section therefore is to expand on the introduction in [1] and provide an in depth look at the inner workings and activities in BOF4WSS

3. BOF4WSS

3.1. Overview

To address the outstanding security issues above, and strengthen available solutions, the Business-Oriented Framework for enhancing Web Services Security for e-business (BOF4WSS) displayed in Figure 1 was conceived. As is illustrated, the framework consists of nine stages which in general, semantically resemble those found in typical systems development methodologies. Formally these stages are, Requirements Elicitation, Negotiations, Agreements, Analysis/Architectural, Agreements, Systems Design, Agreements (for Quality-of-Services), Development and Testing, and Maintenance.

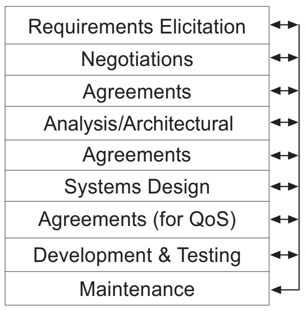


Figure 1. BOF4WSS Overview

The Waterfall Model (WM) methodology in particular was the main influence for the framework's design. This can be seen when comparing BOF4WSS's phases to those of the WM i.e. system feasibility study, requirement analysis and project planning, system design, detailed design, coding, testing and integration, installation, and maintenance [20]. The WM was preferred to other methodologies due to the transparent, well-organized, highly documented, and strongly disciplined process it can bring to this large inter-organizational development project [20], [21]. Some practitioners even view the structure possible with the WM as an ideal fit for the corporate (and somewhat bureaucratic) world, and a key reason why the WM is here to stay [22].

With appreciation of the flexibility and quick turnaround benefits of agile and more lightweight methods, these were also considered at length. These techniques were not chosen as a foundation however, because literature [23], [24] does not advise them in situations: (i) of large development projects; (ii) where development teams might be in different places and dealing with complicated interactions with other hardware and software; or (iii) in critical systems development. These are all likely situations where BOF4WSS might be used, as mentioned in previous and also, later sections.

Despite the benefits listed, it is accepted that the WM is not perfect and does have shortcomings. For example, researchers have identified that it freezes requirements too early, lacks flexibility in the original model when traversing stages, and results in excessive documentation [20]. As opposed to adopting a different methodology however, BOF4WSS addresses these shortcomings by allowing for flexibility through bottom-up progression and feedback (shown on the right in Figure 1), and stressing the involvement of key stakeholders throughout the entire process. Additionally, even though requirements are determined early in the framework, these are only high-level requirements (as opposed to the traditional WM that defines all requirements) which can, and may change at subsequent stages closer to design. The inclusion of the Negotiations and various Agreements stages at the points specified is necessary due to the inter-organizational process, and the importance of companies discussing and agreeing on goals.

The prime novelty in BOF4WSS is the emphasis on providing an expanded formalization of a development methodology that focuses on security, which can accommodate multiple autonomous businesses working together. As will be seen below, the framework and its phases give detailed guidance on what should occur and how, and its pertinence in attaining desired levels of holistic security for these *cross-enterprise interactions*. To recap, *cross-enterprise interaction* security refers to ensuring businesses are secured *internally*, but also that the *external* interactions encompassing collaborating businesses are secure to some level. External interactions to a company simply mean interactions that occur in transit (i.e. while they are being passed between companies), and to some extent what occurs regarding the security of these interactions while being processed by business partners. This internal and external focus is revisited at various points in BOF4WSS's presentation below.

Returning to the point regarding the detailed guidance given by the framework, this will involve defining the expected inputs to stages, along with their required outputs/outcomes, but especially the recommended low-level goals, activities, and steps within those stages that can help achieve the outcomes. Where suitable, this guidance aims to reuse existing methods and practices—both from industry and academia—thus concentrating on the compilation of these into a coherent, well-defined process instead of reinventing standardized parts of the proverbial security wheel.

Another main design goal of the framework is to promote/utilize Web services specifications and tools wherever, and whenever useful. This is done to provide companies that adopt BOF4WSS with a practical methodology that pulls together key WS-specific specifications and tools from the plethora of technologies available, and shows exactly where and how they can fit into the development of a Web services solution. To date, the authors are not aware of such a broad methodology as BOF4WSS, which aims to fit together some critical pieces of the WS security puzzle in the context of cross-enterprise, highly structured, extensible (by allowing different approaches to be plugged in), business-oriented framework.

BOF4WSS's close alignment with Web services, security, and cross-enterprise development, differentiate it from somewhat related, existing frameworks and models such as TOGAF [25] (a detailed method and a set of supporting tools for developing an enterprise architecture), SABSA [26] (a framework for delivering cohesive information security solutions to enterprises), and the Web services development lifecycle [4]. These are all very adequate, de facto approaches, but aim at a much more generic level than is of interest in this research. Otherwise, there are various similarities between these models and BOF4WSS, including identification and involvement of key stakeholders, definition of conceptual data models for foreseen interactions, phase inputs and outputs, and architectural design and technical level implementations.

To support the largely textual description of the framework's activities below, a number of diagrams are included illustrating each stage and its respective workflow. Since security issues are a central concern to BOF4WSS, the discussion concentrates primarily on these aspects rather than an isolated discourse on functional and quality related aspects. (Quality aspects or requirements in this regard refer to non-functional requirements excluding security, e.g. performance, scalability, maintainability, and so on.) At some stages however, in the interest of completeness, this paper does attempt to give some guidance on these areas. This is particularly when they relate to key WS standards and technologies. Lastly, BOF4WSS assumes that businesses have previously agreed (through feasibility studies, initial dialogue, and so on) to use WS to support a generally defined business scenario. In other words, the broad scenario is known. BOF4WSS's task therefore is to provide a methodology for its planning, development and implementation. Below, the framework's stages are presented.

3.2. Requirements Elicitation Phase

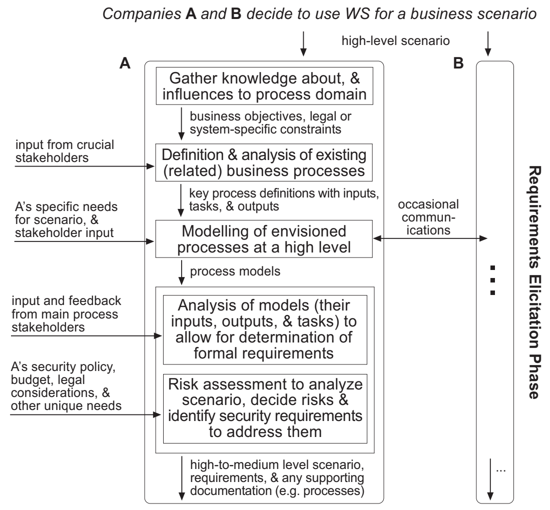

Companies A and B decide to use WS for a business scenario
high-level scenario
A
B
Gather knowledge about, & influences to process domain
business objectives, legal or system-specific constraints
input from crucial stakeholders
Definition & analysis of existing (related) business processes
key process definitions with inputs, tasks, & outputs
A's specific needs for scenario, & stakeholder input
Modelling of envisioned processes at a high level
occasional commun-ications
process models
input and feedback from main process stakeholders
Analysis of models (their inputs, outputs, & tasks) to allow for determination of formal requirements
A's security policy, budget, legal considerations, & other unique needs
Risk assessment to analyze scenario, decide risks & identify security requirements to address them
high-to-medium level scenario, requirements, & any supporting documentation (e.g. processes)
...
Requirements Elicitation Phase

Figure 2. Workflow model of the Requirements Elicitation phase

The **Requirements Elicitation phase** is displayed in Figure 2 and assumes two companies, A and B; more companies however are possible. Within this first phase, each company works largely by itself, analyzing internal business objectives, constraints, security polices, relevant laws and regulations and so on. This is done to determine their high-level requirements for the expected WS business scenario. Typically, a company team should be assembled that would be responsible for project management, system development, cross-enterprise communications, and generally steering and championing the project from inception to fruition.

To aid in the Requirements Elicitation process, the phase utilizes the methods proposed by [27], which focus on the definition and analysis of business process models to elicit requirements (functional, quality, and security-specific). This approach is preferred as it is a tested technique that also has an innate emphasis on business processes—i.e. the culmination of service interactions. During these methods, as with some of the subsequent stages, the framework heavily stresses the involvement of stakeholders, and especially top management buy-in (i.e. support). Validated by studies in [28], these are critical success factors in managing and developing information systems.

As illustrated in Figure 2, the approach in [27] consists of firstly gathering relevant knowledge about the process domain and what influences it. This information could include business objectives, legal or system-specific constraints, existing process models, system architectures, and so on. The second task is the analysis and modelling of current processes (particularly if existing models are not accurate) to enable for a full appreciation of critical process flows, and their inputs and outputs. This will primarily focus on internal and external processes directly involved in envisioned WS interactions, and those that are candidates for redesign. Legacy applications are an important consideration at this point because these are likely to supply critical functionality in the foreseen scenario. These systems will therefore have to be thoroughly understood, and their business functionality/logic rationalized and defined. This is particularly useful in the next task as legacy applications functions are packaged, modelled and included in envisioned processes.

Crucial persons (i.e. stakeholders) of reference for the information mentioned will be top executives, domain experts, project managers, systems analysts and end users. For the modelling activity in this second task, the Unified Modeling Language (UML) is suggested for use as it is a standard technique likely to be known by both enterprises. If companies are entering a process they have not done before (i.e. there are no ‘current processes’ specifically related to the envisioned interactions), this task will not be as relevant. Instead, the aim will be to consider how their internal processes will integrate with this newly envisioned interactions.

The third task is the modelling of new processes. At this point, the needs of new business interactions (driven by the companies and at the core, the stakeholders) result in new processes, but often also include enhanced, and updated existing processes. Legacy systems deserve special emphasis because if they are to be included in new processes, they can be either re-engineered (reimplemented), repurposed (changing interface and encapsulating some business logic), or partitioned and packaged into deployable functional components [29]. The choice between these methods will largely be dependent on benefit versus cost, and whether legacy systems can adequately fulfill new business goals.

Generally, the processes defined in this task are expected to be high-level, and mainly cover internal (i.e. known) as opposed to external (i.e. envisioned) operations. This however may not always be the case, for example, if the external processes with the other company are known due to prior transactions, businesses may be able to develop initial medium-level process flows which encompass the external interactions. In either case, occasional communications with business partners is required to enable useful processes to be defined. Also, again UML is suggested for (i) the reason above, and particularly because these high-level models can be used to aid in discussions in the Negotiations Phase, and (ii) the fact that it adequately enables for high- or medium-level processes to be defined.

The last task in the approach proposed in [27] is the actual requirements determination. This is accomplished through analysis of the newly defined process models. By assessing the inputs, outputs, and tasks involved, general requirements (functional and quality-based) for each stage of the process can be defined at a high level. For quality requirements in particular it is understood that these may be hard to state this early, and at this rather high level, but businesses should make an effort to give some idea of their desires for system quality. To elicit security-specific requirements, the authors mainly analyze the access restrictions of the actors (users or applications) on the processes, and process inputs and outputs. In these last two stages, BOF4WSS heavily involves the previously highlighted stakeholders.

In addition to the security requirements identified above, a scenario risk assessment is strongly suggested to provide more detailed and extensive security information. This assessment, as opposed to the one above which focuses primarily on access restrictions in processes, enables for a comprehensive, security-driven analysis of the scenario. The assessment is strongly suggested primarily to combat the unfortunate reality that a significant number of businesses simply do not carry out formal security risk assessments to identify key risks faced [30]. Or, if companies do engage in risk assessments, studies show that major gaps in risk assessment coverage often are apparent that could result in significant risks being overlooked [31]. To aid in this process, there are a range of assessment methods. BOF4WSS suggests well-documented, and internationally validated, time-tested techniques such as NIST Special Publication 800-

30: Risk Management Guide [32], OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) [33], and CORAS (Construct a platform for Risk Analysis of Security Critical Systems) [34].

Generally, some of the crucial factors considered in a chosen technique should include risks (constituted of assets, threats, vulnerabilities) and their priority levels (i.e. severity and impact if risks materialize), organizational security policies (which directly convey a company's security posture), pertinent laws and regulations (those governing internal operations, and those with respect to working with external parties), security budgets (balancing cost and security is paramount), and security needs expected to be met by new business partners. All of these factors significantly aid in the determination of the security that should be factored in during these envisioned WS communications. These requirements should particularly address areas that (i) need additional security internally (and relate to the overall scenario), and (ii) relate to the interactions with the business partner. After these requirements have been gathered, they are added to the previously identified requirements, and documented to provide the stage's output—i.e. *a high-to-medium level scenario process (inclusive of the models defined), high-level requirements (functional, quality and security), and any other the supporting information*.

3.3. Negotiations Phase

In the **Negotiations phase** next, teams consisting of project managers, business and systems analysts, domain experts, and IT security professionals from the companies meet, bringing together their requirements from the previous phase for discussion and negotiations. Figure 3 displays the workflow. The purpose is to use the inputs to this stage as a basis to chart an **agreed** path forward in terms of scenario and business requirements, and high-to-medium level process definitions. This is especially noting the varying expectations each company is likely to have towards security. Expectations (and requirements) could vary with regards to whether a process (or set of service interactions) needs to be secured, to what level is it to be secured, how will security be applied, and so on. Specifically the two main tasks in this phase therefore are: Discussion and negotiation on (i) functional and quality requirements, and then (ii) security actions and requirements. Depending on the preferences of businesses using the framework, the latter of these tasks may include a joint risk analysis aimed at identifying any risks (and thus requirements) not conceived previously. Deliberations on statutory and regulatory requirements are especially important when discussing security, as businesses may not be in the same industry or even, country. Where necessary, as is seen in the workflow, backward progression from the security requirements definitions to functional/quality requirement definitions is allowed. This is mainly to support balancing between functional/quality and security actions and requirements.

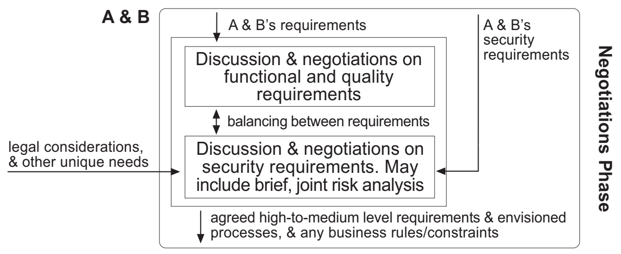

A & B
A & B's requirements
A & B's security requirements
Discussion & negotiations on functional and quality requirements
balancing between requirements
legal considerations, & other unique needs
Discussion & negotiations on security requirements. May include brief, joint risk analysis
agreed high-to-medium level requirements & envisioned processes, & any business rules/constraints
Negotiations Phase

Figure 3. Workflow model of the Negotiations phase

The Negotiations phase facilitates its purpose by accepting that each business constitutes a different security domain (and is likely to have different desires and obligations), and therefore explicitly stresses the need to negotiate on security actions, rather than adopting one company's needs, or assuming integration of desires at this level will be seamless. Work in [35] clearly highlights that in forming these extended networks or partnerships of companies, this integration task is formidable. Regardless however, this is a necessary, and pivotal precursor to engaging in interactions. After the identified tasks have been completed, the expected output of this stage will be *the agreed high-to-medium level requirements, high-to-medium level envisioned processes, and any business rules/logic and constraints, important for future stages*.

3.4. Agreements Phase

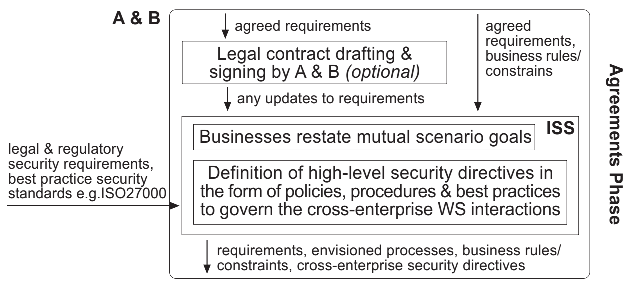

A & B
agreed requirements
agreed requirements, business rules/ constrains
Legal contract drafting & signing by A & B (optional)
any updates to requirements
ISS
Businesses restate mutual scenario goals
legal & regulatory security requirements, best practice security standards e.g.ISO27000
Definition of high-level security directives in the form of policies, procedures & best practices to govern the cross-enterprise WS interactions
requirements, envisioned processes, business rules/ constraints, cross-enterprise security directives
Agreements Phase

Figure 4. Workflow model of the Agreements phase

The **Agreements phase** depicted in Figure 4 builds on the concluded negotiations and initially advocates a legal contract to solidify the understanding of the requirements between companies thus far. A legal agreement at this point is not compulsory however, as it is appreciated that businesses may choose to include the contract at another stage, or to have only one main contract at a later stage when details of interactions are finalized. The reason the contract is suggested here is to create a safety net for both companies during these early stages of planning and negotiations. The contract would focus on two main aspects, binding the parties to negotiations for possibly future business interactions in good faith (non-disclosure agreements may be used for example), and secondly, defining the groundwork for a more comprehensive contract to follow in later stages. The agreement and definition of requirements in the Negotiations phase makes the latter of these tasks (i.e. defining the groundwork) less complex and arduous.

This legal document is followed by the Interaction Security Strategy (ISS) which, as opposed to the contract, is a less rigid management structure that defines high-level, cross-enterprise security directives to guide the interactions and relevant security decisions internal to companies. These directives are typically in the form of security strategies, policies, procedures, best practices, and objectives. Figure 4 shows that the central activities in this stage are: (i) the restating of the businesses' mutual goals for the scenario—this will provide a clear vision for the strategy; and (ii) the actual definition of the security strategy's directives. In addition to the use of requirements, and business constraints, when defining these directives, the framework emphasizes consideration of two aspects, i.e. the legal and regulatory mandates which may influence companies and interactions, and secondly the best practice security standards available from industry. These are discussed below.

In business today, legal and regulatory requirements pertaining to security are becoming increasingly important. This is especially within the arena of online business. These mandatory requirements cover topics such as data protection, data privacy, computer misuse, incident disclosure and notification, third-party auditing, and even security within business relationships. The aim of the ISS with regards to these requirements is mainly to stress that businesses make themselves aware of the content of these laws and regulations. This is not only to fulfill the statutory need, but also because a number of these laws stress principles of good, reliable security that should be practiced by businesses.

Some of the most relevant laws businesses should consider include the Sarbanes-Oxley Act (SOX) of 2002 (U.S.)—this emphasizes the maintenance of adequate internal controls to ensure the accuracy of financial information [10], [36]; Health Insurance Privacy and Portability Act (HIPAA) of 1996 (U.S.)—focuses on confidentiality, integrity, and availability of personal data (medical or personal records) ensuring it is protected whilst in storage, and during transmission, both within and external to the company [36]; Data Protection Directive 95/46/EC of 1995 (E.U.)—this is targeted towards personal data, ensuring that it is adequate, accurate, and processed lawfully, amongst other things [10]; and Gramm-Leach-Bliley Act (GLB) of 1999 (U.S.)—mainly aimed at financial institutions, this act stresses activities such as the evaluation of IT environments to understand their security risks, establishment of security policies to assess and control risks, and the scrutiny of business relationships to ensure partners have adequate security in place [36]. Knowledge of, and adherence to these regulations is critical as companies look to conduct business in an increasingly regulated marketplace.

In addition to promoting the compliance to legal and regulatory requirements, the ISS emphasizes the incorporation of best practice security standards in the approaches by companies towards inter-organizational security. Whilst it may be tempting to assume that businesses already accommodate such standards, recent surveys [30] have shown that companies are largely not aware of key security guidelines. The ISO/IEC 27000 series is a perfect example of important standards, and as Figure 4 shows, they form a key input into this stage. This standards set in particular, is targeted at the provision of an internationally recognized, organization independent framework for effective, extensive information security management [37]. Themes addressed include the definition of essentials (in terms of specifying information security policies, conducting in-depth risk assessments, and so on) for the creation of an adequate information security management system (formally, the ISMS)—this is covered in ISO/IEC 27001:2005; a code of recommended best practices for planning and implementing the ISMS—see ISO/IEC 27002:2005; and detailed guidelines for the information security risk management process to support the ISMS—see ISO/IEC 27005 [37]. The creation and maintenance of a well-conceived, and thorough internal security management system for an organization is the fundamental objective of this standards set.

To put the ISS directives (e.g. laws, standards, and policies) into context, Figure 5 is included below. This illustration, based on work in [38], shows how each aspect covered by the ISS fits in to provide a layered model for the e-business security environment.

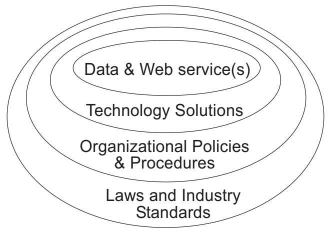

Data & Web service(s)
Technology Solutions
Organizational Policies & Procedures
Laws and Industry Standards

Figure 5. The e-business security environment (based on [38])

Looking directly at the ISS' emphasis on standards during this Agreements phase, there are countless benefits. As mentioned prior, the term *cross-enterprise interactions* denote interactions spanning, and including collaborating businesses and their internal systems. Therefore, securing the *internals* of businesses which participate in these interactions is also a crucial goal—this is especially where the ISO/IEC 27000 standards set is useful. Two specific benefits of applying these standards are that they provide organizations with a systematic way of fulfilling legal and regulatory responsibilities (for example, some standards can help meet SOX requirements), and secondly, through accreditation schemes, businesses which can demonstrate adherence to guidelines, can be issued with a certificate to show customers and business partners that their systems and practices are secure to an international standard [37].

At the *external* level these standards also prove useful as certain clauses (e.g. ISO/IEC 27001, Control A.6.2) deal specially with external parties, and attempting to maintain the security of an organization's information assets as they are accessed, processed, communicated to, or managed by external parties [37]. The two main tasks involved in this attempt are the identification, and addressing of risks directly related to external parties; these are two activities that were completed to some extent during the risk assessment in the preceding Requirements Elicitation phase. Reflecting on Control A.6.2 therefore, as opposed to resulting in an exhaustive legal contract (as is suggested by the Control), any new risks and their respective controls which were identified, would feed into the cross-enterprise security directives for the ISS.

Having discussed the ISS, its goals and its main influences, a brief look is taken at some examples of what the ISS could cover. The first example is the specification of best practices each company should abide by internally. One best practice might be related to ensuring companies maintain sufficient logs of system events; this information would be very useful in cases of a security breach. Another example of an aspect the ISS would address would be the definition of scenario incident response activities i.e. what procedures should companies follow if a security incident is suspected, or has occurred. The third, and somewhat general example relates to the responsibilities, and expectations of companies towards security. The ISS would enable companies to almost always have some clear vision of what their partners should be doing, (likely stated in terms of policies, and procedures) relating to aspects of security. The final example is the creation of a small, cross-enterprise team specifically to handle security matters, and updating the ISS and other security measures as, and when appropriate. Here, the ISS recognizes and appreciates that security is an ongoing concern. Therefore, it calls for a team to be formed constituting of persons from both enterprises to manage this concern. In essence, the ISS forces businesses engaging in joint interactions to consider and address security issues, both internally and externally, that previously may have been overlooked due to overly simplistic, or isolated approaches towards security.

By jointly creating an ISS companies can have some degree of certainty that partners are committed to maintaining an acceptable security posture. This leads to another central goal of this strategy, i.e. to foster trust amongst business partners. The ISS aims to foster trust through predictability and transparency in security approaches, by outlining a security strategy and subsequent framework that all businesses agreed to adopt and follow. Trust within e-business was outlined before (see Section 2.2), and its importance should not be neglected. This paper does note other, more direct methods to assess a business partner's commitment to security, such as audits, on-site visits, and questionnaires (as suggested in [36]), but leaves this choice to individual organizations that adopt BOF4WSS. Within very closely-knit and highly collaborative relationships (such as the e-supply chains) however, audits amongst other precautionary mechanisms are strongly recommended; this opinion is supported by [39]. The closer businesses are, the more likely they are to be affected by each other's security risks. Businesses should be mindful of this factor as they seek to work with other enterprises. To complete this Agreements phase the following documents and information are prepared to be carried forward to the next stage; these are *the high-to-medium level requirements, high-to-medium level envisioned processes, any business rules and constraints, and cross-enterprise security directives in the form of strategies, policies, procedures, best practices, and security objectives (or more formally the ISS)*.

3.5. Analysis/Architectural Phase

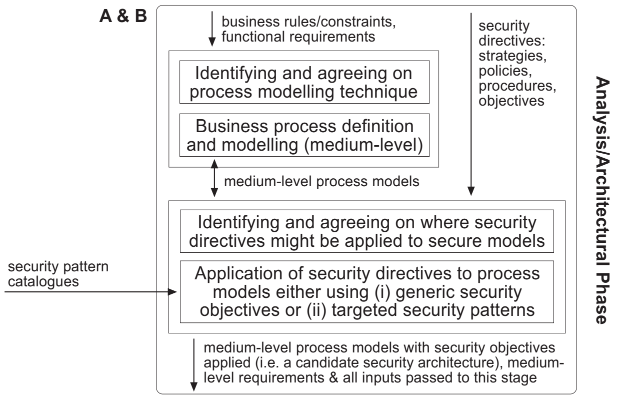

A & B
business rules/constraints, functional requirements
security directives: strategies, policies, procedures, objectives
Identifying and agreeing on process modelling technique
Business process definition and modelling (medium-level)
medium-level process models
Identifying and agreeing on where security directives might be applied to secure models
security pattern catalogues
Application of security directives to process models either using (i) generic security objectives or (ii) targeted security patterns
medium-level process models with security objectives applied (i.e. a candidate security architecture), medium-level requirements & all inputs passed to this stage
Analysis/Architectural Phase

Figure 6. Workflow model of the Analysis/Architectural phase

Following on from agreements, next is the **Analysis/Architectural phase**. The workflow of this stage is given in Figure 6. This phase's purpose to enable companies to take the agreed requirements, and define conceptual (medium-level) business process models for the foreseen interactions. These models are expected to encompass not only the high-level company-to-company process flow, but each company's internal process flows that constitute part of the general business scenario. Internal process definition and sharing is encouraged to cultivate an atmosphere of openness between the companies, but especially to make companies properly analyze the expected internal flows and how they fit into the general scenario. At this point, it is still relatively easy for companies to make any necessary updates. With this in place, the directives (policies, best practices, and so on) from the ISS can then be applied to secure the models. This two-stage method to securing business processes is adopted from research done in [40], which focused on decomposing processes into flows with inputs and outputs, then applying derived security objectives (these are encompassed in our security directives) to secure process components. [41] is an example of other work which adopts a similar, stepped approach to secure e-business processes during design.

To define the medium-level business process models needed, various standard modelling techniques are available (see [29], [42], [43]). Some of the most popular of these are UML (inclusive of its many specialized profiles), Data Flow Diagrams (DFD), Integration Definition for Function Modeling (IDEF) techniques (e.g. IDEF0, IDEF1x, IDEF3), and the Business Process Modeling Notation (BPMN). The UML 2.0 extension for SOA, UML4SOA [44], is a recent proposal from research groups which also provides an interesting technique. This profile however appears to be only targeted at service orchestration (i.e. internal, as opposed to cross-enterprise systems). Yet another option is the UML profile in [45] for Web service composition. This could be very useful because a main design goal is the inclusion of transformation rules that allow designed UML models to be transformed to Web service compositions that are executable (e.g. BPEL, albeit an older version)—a necessary task in future stages.

Having mentioned the SOA, the framework notes that businesses may or may not model processes in terms of *services* at this point. The notion of a *service* here refers to its abstract meaning i.e. distinct units of logic [46]. This is therefore the conceptual prerequisite to actual technology-based Web services. If modelling in terms of abstract services, businesses for example might start defining functionality or processes to package together to form referable units of logic. If initial modelling in previous phases define components to encapsulate legacy system functionality, these could be starting points to regard as services. The benefits of service modelling at this point, is that it could give early insight to where services are likely to fit in, and secondly, that it forces businesses to view processes in terms of services early on. If a company is looking towards full adoption of an SOA framework internally, the latter of these benefits is more crucial.

Although services modelling is an option, there is arguably no need for companies to rush into the services creation task as yet. This is because the forthcoming Design phase which covers a lower level of analysis, addresses this concern in detail. As with many other parts of BOF4WSS however, the final decision is left up to businesses and what suits their ideology and situation best. For the more standardized techniques above, [47] provides a brief outline of the software and tools available to support modelling. The importance of tools cannot be stressed enough, as these are critical in streamlining and easing the modelling process for companies.

Since it is almost certain that companies would have engaged in process modelling at some point before, they are likely to have preferred techniques; because of this, first an agreement is required on the technique that they will use. As can be concluded from Figure 6, the framework does not stipulate that any particular method be used. It however does advice businesses to carefully deliberate the benefits, and shortcomings of the options available. Any assessment should bear in mind: (i) the goal of this phase i.e. definition of **secured** medium-level process models; (ii) the fact that these models will have to be further decomposed and used to express varying aspects (e.g. security or scheduling constraints) at lower level, and therefore having standard ways to state these aspects may be beneficial; and (iii) the impending need to translate (with, or without tool support) these process models into more WS-specific formats, for external and internal usage. Regarding the latter two points,

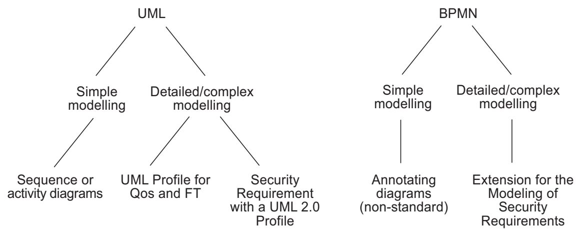

UML
Simple modelling
Detailed/complex modelling
Sequence or activity diagrams
UML Profile for Qos and FT
Security Requirement with a UML 2.0 Profile
BPMN
Simple modelling
Detailed/complex modelling
Annotating diagrams (non-standard)
Extension for the Modeling of Security Requirements

Figure 7. Options for modelling security with UML and BPMN

businesses for example might find it useful to know that firstly, there have been proposed extensions to UML to account for security, and secondly, with highly esteemed options like UML and BPMN, there are mechanisms publicized that can translate these medium-level models to WS-specific languages, as will be seen in subsequent sections. Figure 7 is one guide that can be supplied to companies to give a summary view of UML and BPMN with respect to the options for modelling security. Information on UML profile for QoS and FT, Security requirement with a UML 2.0 profile, and Extension for the Modeling of Security Requirements can be found in [48], [49], [50] respectively.

Researchers in [42] and [43] have investigated into the nuances of a number of popular process modelling techniques, and their findings would be a first point of reference (used by BOF4WSS) to guide companies in choosing a method. The first article provides a taxonomy of modelling techniques to assist decision makers in evaluating and selecting a suitable option based on the project, and/or the specific purpose for modelling [42]. Purposes could range from functional (task-focused) to informational (data flow-based), or from process development to simply enabling for understanding and communication. The second article is a more recent review of the techniques for modelling and culminates in a detailed summary of these approaches (covering their attributes, characteristics, strengths and weaknesses), and a framework classifying them according to their purposes [43]. For more on BPMN and UML4SOA, see [29] and [44] respectively. All of the information on these techniques from resources identified above can be used by businesses to aid in the selection of the most appropriate process modelling approach to suit their specific organizations and needs.

Once the modelling technique has been agreed, Figure 6 shows that businesses then proceed to use the phase's inputs to define and model the cross-enterprise processes. During this task, companies should be wary of the temptation to prematurely define the processes in great detail. Even though it is understood that this is the next step (i.e. the Design Phase), and that for some security objectives low-level analysis is ideal, agreeing on and defining a conceptual model is a critical base step to the following stages. This degree of modelling enables visualization and description of process at an abstract but holistic level, which is comprehensible by all members of the companies' teams, as opposed to only systems designers or software developers. Conceptual process definition can allow companies to analyze processes, weigh alternatives, and assess process inter-relations. Most importantly however, it enables the achievement of agreement on the vision for the medium-level architecture and process flow, in and across enterprises prior to low-level design.

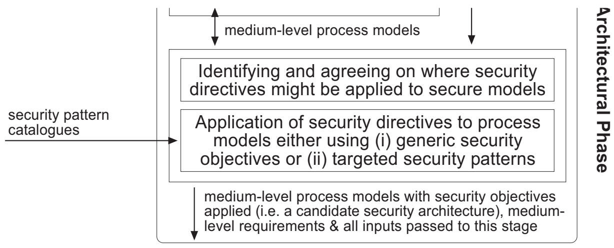

medium-level process models
Identifying and agreeing on where security directives might be applied to secure models
security pattern catalogues
Application of security directives to process models either using (i) generic security objectives or (ii) targeted security patterns
medium-level process models with security objectives applied (i.e. a candidate security architecture), medium-level requirements & all inputs passed to this stage
Architectural Phase

Figure 8. Identification and application of directives in the Analysis/Architectural phase

After defining the cross-enterprise process models, the next general task presented in Figure 8 is to apply the security directives. Due to the range of directives, and the variety of possibilities in which they could be applied to even these medium-level models, businesses are faced with a complex undertaking. Initially therefore, the framework suggests that companies focus on identifying and agreeing on where security directives might, and should be applied to secure the models. A detailed table is one simple way that companies could match security directives to the processes they will affect. The framework accepts that not all directives may be process-specific or -related (for e.g. monthly updates on ISS). When the matching has been completed, there are two methods in which directives can be actually applied to the process models, these are either (i) through the use of generic *security objectives* (as done in [40]) or (ii) by employing targeted *security patterns* (see [10]). These two methods are preferred in BOF4WSS because they provide decent security procedures which are generic enough to be applied, even if only by way of annotations, to a number of the aforementioned modelling techniques. Figure 9 diagrammatically presents the general process.

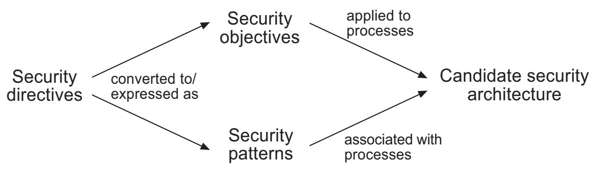

Security directives
converted to/ expressed as
Security objectives
Security patterns
applied to processes
associated with processes
Candidate security architecture

Figure 9. Process from security directives to security architecture

To use the first approach (i.e. [40]) in its original form, companies will have to ensure that process-related security directives are stated with regard to the *security objectives* of confidentiality, integrity, availability, and accountability. This however is not a limitation, because the framework does appreciate and support the desire of businesses to add other, possibly relevant objectives that reflect the directives. These additions might include objectives on nonrepudiation, authentication, and authorization for example. After this is complete, individual process components (i.e. inputs, outputs, activities, and actors—users of process activities) are assigned rating values (for e.g. High, Medium, Low) in terms of these objectives. These values indicate level of security desired for the component, and should be based on previous risk analysis findings and the security directives, as opposed to being just randomly chosen. The following gives an example of an assignment; if a data value α is output from an activity, and the risk analysis or security directives dictate that α is very sensitive data and its confidentiality is likely to be threatened, companies might assign process component α with a confidentiality rating of High. This type of assignment activity is done for all process components in the previously defined models.

The second approach is the application of *security patterns* to secure the process models [10]. Formally, "a security pattern describes a particular recurring security problem that arises in specific contexts and presents a well-proven, generic scheme for its solution" [51]. Simply, it can be thought of as a well-proven, generic solution to a recurring security problem. An immediate benefit of employing this approach therefore is that it would utilize catalogues of proven and tested security patterns to address the requirements in the security directives. This accounts for the input of the security pattern catalogues to this stage as shown in Figure 8. Authors in [10] have investigated this topic in detail, and have provided an extensive listing of existing and new patterns spanning the Web, Business, Web services, and Infrastructure and Quality of Services tiers of a typical company's systems. Using the example of data value α from the previous approach above, personnel at companies would check through the security catalogues for an appropriate pattern to protect α. Having identified suitable alternatives, these would then be noted for formal analysis and application during the subsequent Design Phase. The goal at this Architectural stage therefore (as illustrated in Figure 9 and also done in [10]) is mainly the identification of relevant security patterns.

To briefly compare the security objectives [40] and security patterns [10] methods, the first approach is likely to be more time consuming, as applying priorities for the security objectives to each process component is a substantial task. Conversely, two benefits accompanying this method are, the simplicity of use and application, and secondly, that it naturally enables for the security priorities (e.g. High, Medium, Low) to be associated with the specific components. The latter of these tasks is not inherently accommodated in the security pattern concept, albeit easy to add in some cases. If companies chose to use patterns, the advantages include, having their security problems addressed in a structured way, and also the ability of non-security experts to reference and apply proven security solutions (through the use of pattern catalogues) to solve otherwise overly complex problems [51]. An additional benefit of using the pattern catalogue in [10] specifically is that it largely is geared towards Web services interactions and is thus equipped with standards and technologies that can be used to implement the pattern in later stages. Regardless of the method chosen, the Architectural stage's output should be *medium-level process models with security directives applied (formally, this constitutes the candidate security architecture), the medium-level requirements (functional, and security-specific) accompanying these models, and the inputs passed into this phase*.

3.6. Agreements Phase

Following the formal conceptual process definition, the framework suggests the use of another **Agreements phase**. The respective workflow can be viewed in Figure 10. At this point, the agreement is in the form of a more thorough legal contract reflecting detailed expectations of the parties included in the envisaged business scenario. The business rules and constraints, functional requirements, and security

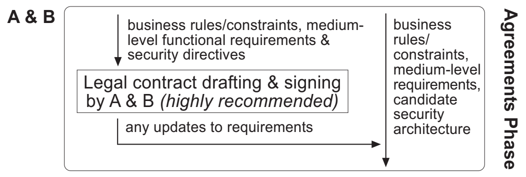

A & B
business rules/constraints, medium-level functional requirements & security directives
Legal contract drafting & signing by A & B (highly recommended)
any updates to requirements
business rules/ constraints, medium-level requirements, candidate security architecture
Agreements Phase

Figure 10. Workflow model of the Agreements phase

requirements all factor into this contract. The medium-level requirements are especially important as they provide further detail on the agreed interactions. During contract drafting, it is accepted that requirements may change, and therefore any updates made are fed back into the known requirements and process models. Again, this legal document is used primarily as a safety net (in the event that companies have an irreconcilable disagreement and need formal arbitration), and therefore still relinquishes the role of governing day-to-day interactions to the ISS. Many authors [17], [26] support this and similar views, and highlight a number of drawbacks to using contracts as the sole basis for conducting business. The outputs of this phase are *the medium-level process models with security directives applied (formally, this constitutes the candidate security architecture), the updated medium-level requirements (functional, and security-specific) accompanying these models, the business rules and constraints, and any other the inputs passed into this phase*.

3.7. Systems Design Phase

The **Design phase** next is analogous to a company's internal systems design process (for e.g. see [10]) and therefore targets the definition of a low-level (or logical) systems-related view of exactly how the conceptual model from the Architectural phase will be put in place. In Figure 11 the specific tasks in this stage are presented diagrammatically. As is shown, the first activity is for the teams from each business to jointly define the low-level process models. The framework advises businesses to reuse the modelling technique chosen before (in the Architectural phase) but on this iteration, to break down the medium-level models to the lowest level of detail. The goal is to decompose models such that the individual message flows between companies can be seen, and also the specific tasks which constitute each process activity. In defining these low-level interactions, it is critical for company teams to identify the actual *services*, and define the interactions in terms of these services. Work in [46] is one commendable reference that examines moving from business processes to service models and designs, which also provides thorough guidance. Generally however, businesses should be attempting to identify aspects of functionality within processes that could form distinct logic units. To exemplify this task, Figure 12 is used.

This diagram shows a simplified medium-level process

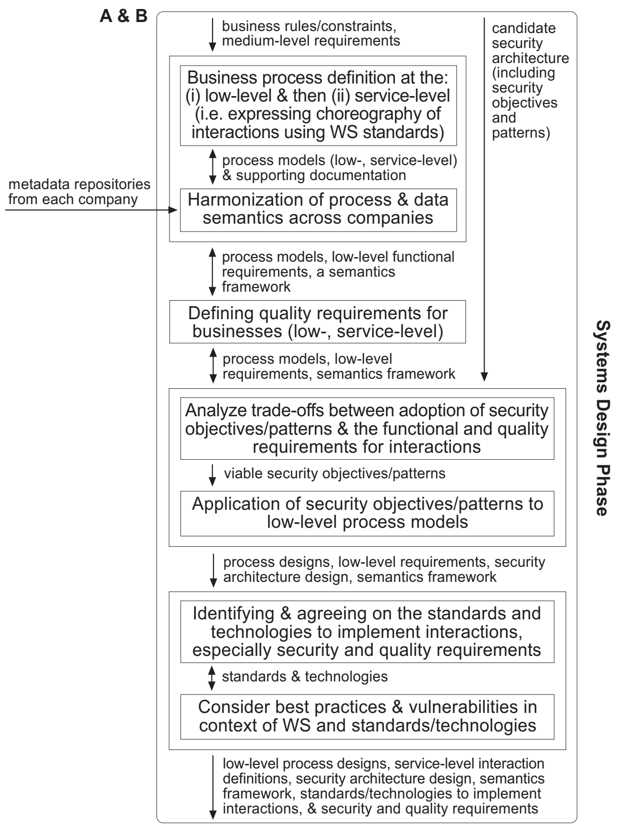

A & B
business rules/constraints, medium-level requirements
candidate security architecture (including security objectives and patterns)
Business process definition at the: (i) low-level & then (ii) service-level (i.e. expressing choreography of interactions using WS standards)
process models (low-, service-level) & supporting documentation
metadata repositories from each company
Harmonization of process & data semantics across companies
process models, low-level functional requirements, a semantics framework
Defining quality requirements for businesses (low-, service-level)
process models, low-level requirements, semantics framework
Analyze trade-offs between adoption of security objectives/patterns & the functional and quality requirements for interactions
viable security objectives/patterns
Application of security objectives/patterns to low-level process models
process designs, low-level requirements, security architecture design, semantics framework
Identifying & agreeing on the standards and technologies to implement interactions, especially security and quality requirements
standards & technologies
Consider best practices & vulnerabilities in context of WS and standards/technologies
low-level process designs, service-level interaction definitions, security architecture design, semantics framework, standards/technologies to implement interactions, & security and quality requirements
Systems Design Phase

Figure 11. Workflow model of the Systems Design phase

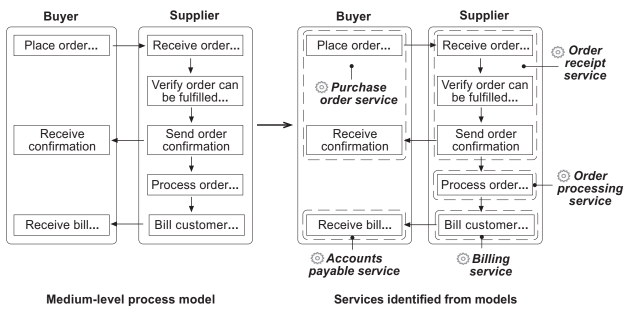


Figure 12. An example of moving from processes to services

flow of a typical order processing scenario (on the left), and next to it (on the right) the services that were deduced from it. In identifying services, special attention was paid to subprocesses that could be somewhat independent, and could be grouped and encapsulated with related tasks. The purchase order service is a good example of this as it encapsulates the ‘place order’ and ‘receive confirmation’ subprocesses into one unit of functionality that can be referenced.

Depending on how open companies have chosen to be with how their processes (or systems) will work internally, the low-level process definition purported might be primarily of the interactions *between* companies, or the interactions *between and also within* the businesses. To use Figure 12 to explain this point, the former of these tasks refers mainly to the arrows connecting the Buyer and Supplier, whereas the latter refers to those arrows plus the arrows and flows within companies. Even though the ultimate degree of openness maintained by companies throughout the framework’s activities is largely left to the individual teams, BOF4WSS stresses that openness and transparency could foster trust between these companies. This trust will be a key ingredient to successful future business interactions.

Building on the low-level process definitions, Figure 11 shows that the following task is the application of WS process specification technologies, to state these low-level definitions in terms of WS-level interactions (expressing them in terms of Web services wherever appropriate). This transformation task is made much easier once the low-level processes have been stated to resemble *services*. WS should be viewed as the Internet-based implementation technology that will implement designed services. At this point, expressing the interactions from a global perspective (i.e. showing interactions *between* companies rather than *internal* process flows) is desired as it allows for the creation of a contract that defines a jointly agreed set of orderings and constraint rules whereby WS message exchanges can take place [52]. To facilitate the expression of this global services contract, the framework suggests one of two options, either (i) the use of W3C’s Web Services Choreography Description Language (WS-CDL)—WS-CDL provides a standard mechanism for defining WS collaborations, and choreographies of message exchanges from a global viewpoint [52]; or (ii) BPEL4Chor—a recent proposal from the research community built on Business Process Execution Language (BPEL), that aims to address a number of perceived shortcomings of WS-CDL [53], [54]. These approaches were chosen specially because of their suitability for WS, and ability to produce formal, Web service-level process specifications that could feed into future framework phases. ebXML’s Business Process Specification Schema (BPSS) is another popular option that can specify business transactions and their choreography [29]; this method is not preferred because of BOF4WSS’s aim to primely utilize WS-specific technologies.

In deciding whether to use WS-CDL or BPEL4Chor, the framework highlights the following factors for consideration by businesses. This paragraph assesses WS-CDL and the next, BPEL4Chor. In terms of politics in the standards world, WS-CDL is likely to have more support from industry because it is under the charter of the W3C. (Notably however, a concise research survey in [55] provides a counter-argument to this assumption as they state that interest in this specification has dwindled.) A second advantage of WS-CDL is that, from the WS-CDL document defined, companies are largely able to generate BPEL workflow templates for their *internal* process flows, that reflect the global business agreement [4], [56]. Third, because WS-CDL leaves actual implementation decisions and details to companies, it allows them the flexibility to use preferred internal technologies. A high-level example is given in [52], where one company may use BPEL engines to drive workflow whilst another uses a more traditional J2EE^{TM}solution. Another factor regarding WS-CDL is that if companies had chosen to use the UML Profile for Schedulability, Performance, and Time Specification [57] to model processes in the Architectural phase, research work in [58] has investigated a method for translating those models into WS-CDL documents. This could therefore be plugged in, and used by companies to automate document creation. Lastly, there is some (albeit very limited) tool support targeted at providing users with the ability to produce, view, simulate, and validate WS-CDL choreographies—namely WS-CDL Eclipse [59], Pi4SOA [60] and LTSA WS-Engineer [61]. These could be employed by companies to assist in creating and testing the WS-CDL definitions.

At its core, the second approach i.e. BPEL4Chor, defines extensions to BPEL to enable the definition of choreographies [53]. In light of this close association, BPEL4Chor can be seen to be specially suited for situations where businesses will desire subsequent BPEL workflow specifications for their internal process flows. The ability to allow for a seamless transition between choreographies (in BPEL4Chor) and orchestrations (in BPEL) is actually one of the main advantages this approach has over WS-CDL (when considering moving from WS-CDL to BPEL workflows) according to its proponents [53]. A second noteworthy factor is that if businesses have used BPMN to model processes in previous stages, research in [54] describes how these BPMN models can be reused, and largely transformed to BPEL4Chor. A plug-in for an available graphical modelling environment is also proposed to aid in this transformation. [53] should be referenced for more nuances of this approach as compared to WS-CDL. In summary, WS-CDL and BPEL4Chor are both viable solutions for Web service-level process specification. With the information provided above, companies can chose their technologies of preference.

Along with the low-level process definition shown in Figure 11, harmonization of process and data semantics across companies is critical. In this paper however, this activity is not covered as it would necessitate an extensive discussion that digresses considerably from the overall focus on security. For information, some of the main aims during this stage would be tackling the semantic interoperability problem at both the data and business process levels. This problem, as it relates to the B2B context, is discussed in detail in [4]. Addressing these issues would likely include the use of tools such as ontologies, shared vocabularies, metadata repositories, and depending on companies', also technologies such as Semantic Web Services, ebXML process definitions, and RosettaNet's Partner Interface Processes (details of each, available in [4]).

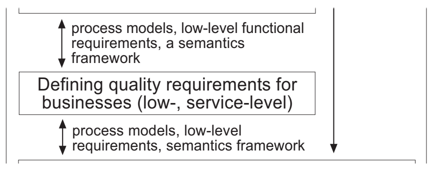

process models, low-level functional requirements, a semantics framework
Defining quality requirements for businesses (low-, service-level)
process models, low-level requirements, semantics framework

Figure 13. Definition of quality requirements task in the Systems Design phase

Following the definition of processes and harmonization of semantics, the goal switches to the determination of the quality requirements at these lower levels. For ease of reference Figure 13 illustrates the task. In earlier stages, quality requirements were produced at a high level and these form the base for the actions here. For this task, businesses, especially their analyst and systems designers play central roles. Business teams need to decide details such as availability (or uptime) of systems and services, acceptable latency levels, performance expectations by parties, and more general aspects including usability, scalability, and even maintainability of envisioned systems. [62] has compiled an appropriate listing of WS quality of service attributes that can be used as a starting point by teams. It is important that these requirements and their relation to processes be well thought through because they constitute prime factors against which the security design will have to be balanced. Businesses can either mainly discuss and agree on these quality requirements, or if a more "hands on" approach is preferred, use available techniques to specify requirements. UML for example has a profile for modelling quality of service characteristics (see [48]) that can be used.

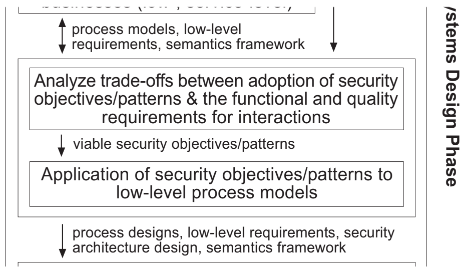

process models, low-level requirements, semantics framework
Analyze trade-offs between adoption of security objectives/patterns & the functional and quality requirements for interactions
viable security objectives/patterns
Application of security objectives/patterns to low-level process models
process designs, low-level requirements, security architecture design, semantics framework
stems Design Phase

Figure 14. Security analysis and application tasks in the Systems Design phase

The next step in BOF4WSS (an excerpt is shown in Figure 14) returns the focus to security and aims to finalize the security architecture and build the security design. The first task in fulfilling this aim is analyzing the trade-offs between the adoption of security objectives/patterns and the low-level functional and quality requirements from prior tasks. Cost, where possible, should be generally factored in by business teams as it pertains to adopting the security directives, remembering that these will translate into security mechanisms and technologies later. Systems designers and security professionals with knowledge of this area can aid significantly in this task. Work in [26] yields a perfect example of the hard task faced by businesses in attempting to balance these often conflicting objectives. Abstracting to the three basic, conflicting aspects, namely security (i.e. a security requirements), cost (i.e. a general limitation) and usability (i.e. a quality requirement), the author states, "To obtain higher security . . . will cost more. To increase security often impacts upon usability, and visa versa" [26]. Another brief example is a company's use of three security patterns to ensure the integrity of messages passed between it and its business partners. From a security perspective, this is ideal (the more security the better), but from a performance perspective, it is unlikely to be accepted because excessive security will undoubtedly negatively affect processing time. Looking at the security objectives method, even though businesses may desire to have every message secured to the highest priority level in terms of confidentiality and integrity, financially, this may simply not be realistic noting cost of certificates, software and so on. These are the types of factors to be assessed in this step.

Once the analysis is complete, the viable security objectives/patterns are then applied to the low-level process models to fashion the business process designs. Figure 14 covers this task. In the Architectural phase, security objectives have already been applied therefore if businesses have utilized this method, the task now is to break down the secured medium-level processes, and associate the objectives with lower-level process components (from the low-level models above). For example, as opposed to specifying a confidentiality objective of 'High' on all outputs from one activity (or task or system) to another (as was likely done in the Architectural phase), businesses should consider the individual messages output and whether they all need the 'High' confidentiality rating. The messages should be visible from low-level process models, therefore the ideal situation would be to take the low-level models, and modify them to show the new, specific levels of security required for all process components.

For the application of viable security patterns, depending on the modelling technique chosen, patterns can be easily woven into the low-level process models. Companies will first need to gather the associations made between the medium-level processes and security patterns from the Architectural phase. Then, using the associations, teams can begin to link low-level processes (from which the medium-level processes were defined) to the relevant security patterns. This is followed by the actual application of patterns to models either conceptually (by way of detailed annotations), or logically (within the formal models). Even though some techniques may prove more efficient at this application task, the conceptual solution that security patterns provide should enable a relatively manageable task for the security professionals on the teams. To give an example of possible output, a snippet of a UML process model with patterns applied is included in Figure 15.

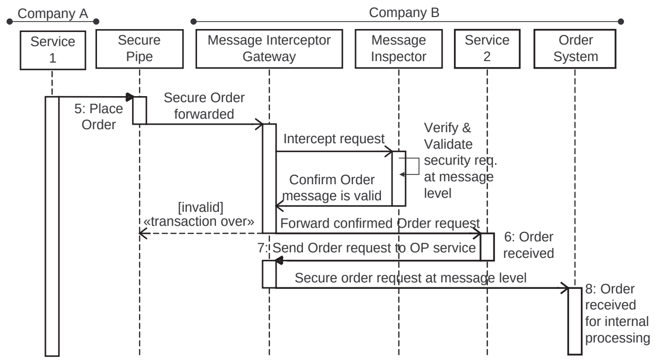

Company A
Company B
Service 1
Secure Pipe
Message Interceptor Gateway
Message Inspector
Service 2
Order System
5: Place Order
Secure Order forwarded
Intercept request
Verify & Validate security req. at message level
Confirm Order message is valid
[invalid] «transaction over»
Forward confirmed Order request
6: Order received
7: Send Order request to OP service
Secure order request at message level
8: Order received for internal processing

Figure 15. UML process model with patterns applied

In this UML sequence diagram, three security patterns have been applied to protect order-related communications/messages between systems at Company A and Company B. These are (i) secure pipe—for securing a basic connection between trading parties; (ii) message interceptor gateway—a central location to manage security enforcement tasks; and last (iii) message inspector—this is responsible for the verification and validation of the security elements in the data or message delivered. These patterns were sourced from [10], and more examples of their use and application can be gathered from that reference.

Due to its versatility and extensibility, UML again forms one of the better techniques for the modelling task. In Figure 7, it was shown that for simple modelling, sequence or activity diagrams are useful; an example of which is seen above. To facilitate detailed modelling, one suggested option is the UML profile for security, quality and fault tolerance requirements. This profile is defined in [48] and provides a standard mechanism for expressing security. Another noteworthy option still within the structured confines of UML can be found in [49]. This research work supplies a UML profile specifically for secured business process modelling using activity diagrams. Security aspects accommodated include auditing, and security requirements such as integrity, attack detection, non-repudiation, access control and privacy. UMLsec [63] and SecureUML [64] are two additional, more detailed security-related extensions to UML that might also be of interest to businesses.

With regards to BPMN, the inclusion of security aspects in models is much less researched and standardized when compared to advances in UML. Authors in [50] state that BPMN "does not explicitly consider mechanisms to represent the security requirements". Because of this fact, BOF4WSS primarily suggests annotations to models and detailed supporting documentation when engaged in simple modelling. Detailed modelling as highlighted in Figure 7 can be partially addressed by recent research work—see [50]. In that article, researchers recognize the need to have the capability to include security in models, and therefore develop an extension to BPMN for modelling security requirements in business process. Albeit not fully complete, this work provides an invaluable start for companies in moving from detailed notes to formal, standards-based models. The caveat to adopting this approach however is its newness. This means that there may be changes or updates in future security modelling notations (thus the possible need to reconstruct process models), and also that it lacks tool support (the need for tools to streamline and ease process modelling should not be overlooked). For all of the other modelling techniques, where no special accommodation is made for modelling security actions or requirements, the framework advocates annotating the models and making detailed supporting documentation. This is not ideal because requirements are not directly and practically applied to models however, it should provide teams with enough relevant information to proceed.

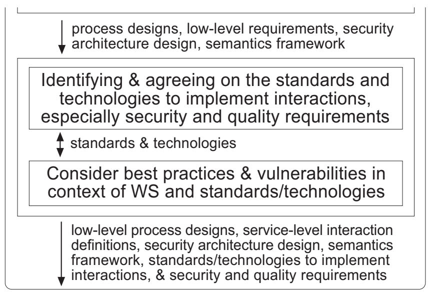

process designs, low-level requirements, security architecture design, semantics framework
Identifying & agreeing on the standards and technologies to implement interactions, especially security and quality requirements
standards & technologies
Consider best practices & vulnerabilities in context of WS and standards/technologies
low-level process designs, service-level interaction definitions, security architecture design, semantics framework, standards/technologies to implement interactions, & security and quality requirements

Figure 16. WS standards agreement and assessment tasks in the Systems Design phase

The penultimate task in the Design phase depicted by Figure 16, is identifying and agreeing on the standards that will be used to implement the services, and especially the security and quality-of-service (QoS) requirements. In general, even though WS is one of the leading interoperability technologies today, basic tasks such as agreeing on standards (within WS) is still crucial to a successful deployment. Authors in [14] allude to this fact as they discuss the "What's missing" in Web services technology. The main interoperability problems they identified stem from the existence of too many standards (over sixty already), the tweaking of standards by individual companies, and the numerous versions of even the basic WS standards [14]. Authors accept that WS-Interoperability (WS-I) [65] profiles can address some of these problems, however they note that this is only possible if companies make their Web services compatible to the WS-I profiles. Their work provides just one example of the importance of the agreement on the standards to be used by businesses.

In this task, systems analysts and designers knowledgeable in the intricacies of WS technologies should take the lead at this point. Whereas analyst help to provide the bridge between the previous works (requirements, low-level processes, and so on), designers look at service and technology details. Due to the extensive number of technologies available and the frequent updates made, instead of covering the standards within the framework, BOF4WSS provides companies with key information sources which they can reference. Sources range from published texts [4], [10], [19], [66] for introductory- and intermediate-level material, to the actual standards Web sites i.e. W3C, OASIS, Liberty Alliance Project and WS-I, for up-to-date, definitive information.

To identify security standards, the work of [10] is particularly relevant if companies have used their security pattern catalogue in previous stages in the framework. The reason for this is because within their catalogue, also supplied is a list of standards and technologies that can implement the respective patterns. For example, to implement the message inspector pattern mentioned above and shown in Figure 15, [10] suggests options of XML Encryption, XML Signature, SAML and XKMS to name a few. Information on these and other security standards and technologies can be found in NIST guidelines such as [5]. One of this article's core purposes is to provide companies with "practical, real-world guidance on current and emerging standards applicable to Web services" [5]. Briefly touching the topic of standards and technologies for QoS requirements, this area is less developed. Companies however can find some information in articles such as [67]. This covers a number of WS QoS aspects, mentions standards which are used to implement them, and also discusses techniques to improve Web service quality.

A final short point companies should be mindful of during the identification and selection of standards is the tool support available to actually use the standards in a production environment. If there is an absence of tools, regardless of the benefits of standards proposed, these standards cannot be used. Common reasons for little, or no tool support include newly developed/ratified standards (i.e. they lack maturity), and rejection of standards from key tool provider companies such as Microsoft Corporation (with .NET), or Sun Microsystems Inc. (with J2EE).

Having agreed to some degree on the standards and technologies to be employed, BOF4WSS (see Figure 16) advises companies to consider (i) the common vulnerabilities and pitfalls in WS and the mechanisms chosen, and (ii) the best practices in using the WS standards/tools, and implementing them as securely as possible. Both of these factors may have been analyzed in some respects before, but because of their significance and the complexities regarding technologies themselves, it is reiterated here. As done above with standards and technologies in the previous task, because of the large number of vulnerabilities and range of best practices, BOF4WSS references more complete and detailed sources rather than listing them. For the first factor i.e. common vulnerabilities and pitfalls in WS, two prime sources are documents from organizations such as NIST (see [5]) and WS-I (see [68]). These give information on common attacks, risks, and typical security challenges.

For the second task in Figure 16, namely the consideration of best practices in using standards and dealing with the various security challenges, the following articles provide designers and developers with some useful techniques. [5] provides general guidance in addressing threats and on secure implementation tools and technologies. [10] gives various best practices and design strategies. [68] identifies typical countermeasures (technologies and protocols) to mitigate common WS threats. Finally, [69] lists techniques to protect against more threats to WS. In light of the vulnerabilities and best practices discussed, BOF4WSS gives companies the option of revisiting the preceding task to reassess the standards and technologies chosen. This progression can be seen in Figure 16, and is highlighted because depending on vulnerabilities or best practices, teams may often opt to use different, more robust standards, or technologies with extensive guidance (practices) on their use. This completes the Design phase and the expected outputs are *low-level process designs, service-level interaction definitions, security architecture design, a semantics framework, the standards and technologies of choice to implement the WS interactions, and the low-level requirements (functional, security and quality)*.

3.8. Agreements (for QoS) Phase

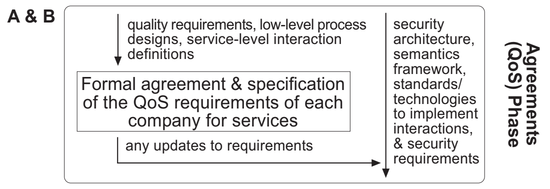

A & B
quality requirements, low-level process designs, service-level interaction definitions
Formal agreement & specification of the QoS requirements of each company for services
any updates to requirements
security architecture, semantics framework, standards/ technologies to implement interactions, & security requirements
Agreements (QoS) Phase

Figure 17. Workflow model of the Agreements (for QoS) phase

With the low-level process designs, and service-level interactions defined, the **Agreements phase** concentrates on the agreements necessary at the quality-of-service (QoS) level. During the task shown in Figure 17, the goal is to specify the mutual understanding of the priorities, responsibilities, and guarantees expected by each business with respect to the other entity, regarding the actual Web services. This phase directly extends work on quality requirements in the Design phase, and in the end, results in a set of formal and contractual agreements. As done before, QoS requirements typically assessed include service availability needs (e.g. a service uptime of 99.98%), performance requirements (e.g. average response time of 30 milliseconds), and so on. Besides quality requirements, process designs and service interactions are necessary for input because they too need to be considered in defining appropriate QoS levels for services and systems.

To specify the QoS requirements agreed, businesses have a few alternatives. The first and most common option is a contractual, natural language agreement referred to as a Service-Level Agreement or SLA. SLAs date back to many years before WS, and since their inception have proved very useful mechanisms to define levels of service in a measurable way (to allow for monitoring), and also the penalties where agreed levels are not fulfilled. For WS, SLAs will have the same usage and general mode of application. The only difference may occur in how services are monitored, as more WS-specific tools and techniques are likely to be employed which enable increased granularity and efficiency in monitoring. For more details on SLAs and what can be included in a WS context, companies can reference [4].

Another option proposed by the research community in [70] is to make use of accepted policy standards such as WS-Policy to specify a service's quality requirements. This method however is ideally suited for dynamic interactions where quality requirements greatly influence the services, or service providers chosen for use. The last noteworthy approach is the Web Services Level Agreement (WSLA) framework described in [71]. Broadly, this framework allows for the specification and monitoring of SLAs for WS. It enables service users and providers (i.e. companies in BOF4WSS context) to define a variety of SLAs, specify the SLA parameters (e.g. availability, response time) and the method for their measurement, and finally relate them to implementation systems. Implementations of the WSLA framework have been built and are available for use in some IBM products—see [72]. Once the specification of the QoS requirements of each company for services is complete, the outputs of the phase to be made ready are *QoS agreements, low-level process designs, service-level interaction definitions, security architecture design, a semantics framework, the standards and technologies of choice to implement the WS interactions, and the updated low-level requirements (functional, security and quality)*.

3.9. Development and Testing Phase

As with most methodologies, the penultimate stage in BOF4WSS is the **Development and Testing phase**. Having discussed how services and systems would interact in a cross-enterprise context, this phase (shown in Figure 18)

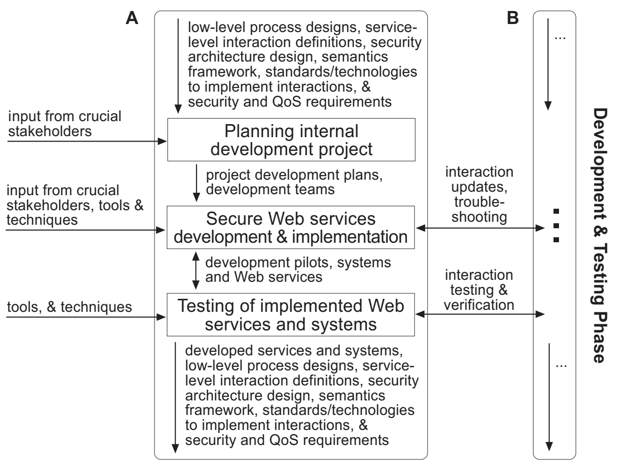

A
B
low-level process designs, service-level interaction definitions, security architecture design, semantics framework, standards/technologies to implement interactions, & security and QoS requirements
input from crucial stakeholders
Planning internal development project
project development plans, development teams
input from crucial stakeholders, tools & techniques
Secure Web services development & implementation
interaction updates, trouble-shooting
development pilots, systems and Web services
interaction testing & verification
tools, & techniques
Testing of implemented Web services and systems
developed services and systems, low-level process designs, service-level interaction definitions, security architecture design, semantics framework, standards/technologies to implement interactions, & security and QoS requirements
Development & Testing Phase

Figure 18. Workflow model of the Development and Testing phase

is centered on the actual development, implementation, deployment and testing of services and systems at the companies. Because of this factor, it is mainly carried out by companies individually, with each company working on their own systems development. Occasional, or even prolonged joint interactions are however greatly appreciated especially for services testing, updates, troubleshooting and systems verification to the requirements established in previous framework phases. All the inputs to this phase are to be used by companies and their development teams to steer the internal systems implementation. It is stressed that even though Testing is presented last (i.e. after discussing Development), companies may choose to do some testing as services and systems are developed.

Unlike some of the previous tasks covered by BOF4WSS, activities for the development stage appear to be somewhat well-established in literature and practice. This is consistent with this paper's argument regarding the significant focus on technology-based and -oriented solutions (which are dominant during this phase). The benefit of this to the framework is that there are a variety of tested development processes, techniques and tools that can be plugged in during this framework phase. As a result, this phase is much less strictly prescribed, with Figure 18 mentioning only three very generic tasks (Planning, Development and Implementation, and Testing) which are not structured in detail like prior tasks. BOF4WSS's aim at this pointer therefore, becomes the identification of relevant, mature and largely complete development processes, techniques and tools that can be employed, and allowing companies the freedom to combine them to best suit their respective situations. Two such processes which are instrumental in aiding in this internal process are [4], [9].

In the former work, [4] presents a WS lifecycle methodology that concentrates on critical internal aspects. These include application integration, packaging legacy applications into reusable components, migration from old to new WS-based processes, and the 'best-fit' ways of implementation which appreciate company constraints, risks, costs, and returns on investment. This methodology is cyclic (as opposed to linear) and consists of nine stages, namely Planning, Analysis, Design, Construction, Testing, Provisioning, Deployment, Execution, and Monitoring. This process is one of the most appropriate and comprehensive within the literature for SOA-based deployment. It covers from initial analysis of internal systems, to the construction and final installation or deployment of services.

A caveat to the lifecycle methodology however, is its lack of emphasis on security concerns—a prime target and goal within BOF4WSS. To compensate for this shortcoming, the framework additionally suggests the integration of PWSSec [9]—a detailed development process for secure Web services. The novelty behind this process is (i) its appreciation of the complex task faced by businesses as they attempt to make use of WS, (ii) the highly structured, methodical approach to constructing a security architecture for WS systems, and (iii) the emphasis on traceability and reusability which translates into the establishment and use of a number of repositories and record stores. The three phases in PWSSec are, Web Services Security Requirements, Web Services Security Architecture, and Web Services Security Technologies. These work together to enable the development of secure WS systems. In brief, another general point of reference to supplement the two already mentioned can be found in [73]. This text provides some useful guidelines that can be applied within the planning task, related to the planning and staffing a WS development project.

Probably the biggest benefit of using the processes listed above is that almost all of the information gathered and produced earlier in the framework can be reused to quickly complete their initial stages. Such information includes functional, security and QoS requirements, risk assessment data, and business process models. If we consider the Analysis phase in [4] for example, in BOF4WSS's Requirements Elicitation and Architectural phases, companies have already worked on the current and envisioned (or "to-be" processes). Regarding the Design phase (in [4]) and the specification of business processes (looking towards WS-CDL and BPEL), BOF4WSS's Architectural and Systems Design phases have previously defined business processes to even these lower levels. Even though the framework's focus was on WS-CDL (and BPEL4Chor), these process definitions can be converted to the BPEL advocated in [4]. For the more security-specific PWSSec [9], the medium- and low-level security requirements, and security patterns identified from BOF4WSS can be reused in PWSSec's Requirements and Architectural stages. [9] also uses UML and the profile for security ([48]) for some of its modelling; one would recall that this is a method supported in BOF4WSS. These are just a few concise examples of how the outputs from BOF4WSS's previous stages can be reused in these processes.

In addition to the identified processes, as mentioned above, literature has supplied a number of techniques and tools to help in this internal development task. An area in particular which has received great focus is the automated creation of BPEL processes from theoretical modelling techniques (e.g. UML, BPMN). To recap, BPEL allows for the specification of business process behaviour based on Web services. Amongst other things, it is an execution language which can be run by software engines to orchestrate message, control and data flows. If companies have modelled process in UML or BPMN therefore, techniques such as [45], [74], [75] that offer some aid in translating these models to executable processes (in BPEL) are quite ideal. Specifically, [74] works with the translation of BPMN models to BPEL definitions, whereas [45], [75] aim at transforming their UML variants and extensions (which may be have been used by companies) to their respective BPEL process representations.

A critical activity in the Development phase is the implementation of the security standards and technologies that have been agreed. Implementation includes the actual application of standards and security levels to the services and systems, but also the correct configuration of the security mechanisms employed. Even though output from the previous phases gives a clear outline of security and where, and to some extent how it is to be applied, noting the peculiarities of WS (e.g. service policy specifications, federated security), this task is still far from trivial.

Researching security configurations for WS, [76] highlights the difficulty in this task and the usability problem faced by developers regarding choosing cryptographic algorithms, encryption keys and so on. To aid in this activity therefore, they propose a tool to fill the gap between business-level security requirements and the lower-level, concrete, technology-specific policies implementing them. This GUI tool, called the WS-Policy Organizer (WSPO), enables users to partially create a platform-specific WS-SecurityPolicy document from a somewhat high-level process definition, through the use of a number of preset security patterns. The integration of this tool within the framework should be reasonably simple because the process scenarios necessary are available from previous BOF4WSS stages, and secondly, the preset security patterns used can easily be matched to the security objectives and patterns from the Architectural and Design phases.

Before moving on, it is worth explicitly stating the importance of including tools for monitoring, both the QoS levels defined in the SLAs, and the security implementations for their reliability and robustness. QoS monitoring constitutes the main focus of the Monitoring stage in the WS lifecycle methodology from [4]. Companies that use that methodology therefore can receive more information on it there. Regarding security monitoring, the key is to install softwares to maintain adequate logs, audit trails and records that can be referred to as required. Authors in [10] highlight that having these audit trails has even become a requirement of some laws e.g. SOX. Intrusion detection, or prevention software may also be of interest to businesses. Fortunately, some of the softwares mentioned are implementations of typical, higher-level security patterns, and therefore are very likely to be included in developed systems.

The final task within this phase is the testing of the developed Web services and systems. This is done to verify that the developed applications meet the intended requirements. It can, and should be done at a *cross-enterprise level* (i.e. both *internally*, and *externally*, across companies). Testing can occur from three main perspectives; functional (do Web services do what they should), quality (are the set performance, usability, scalability, etc. requirements met), and security (is there adequate protection in place for Web services and systems). Guidance on testing the functional and quality requirements is given in lifecycle methodology [4] mentioned before. A much more complex operation is testing the security of the applications developed. Whereas one can pass input data into a system or process and (based on the output) quickly determine whether a functional requirement has been met, security is not that absolute nor can it be so easily measured [26]. Conversely, this however does not mean that testing is impossible, or should it be viewed as a task to be avoided by businesses.

Like approaches for the other testing perspectives, the initial activities are the same i.e. identify requirements (these may be in terms of needs, goals, threats that should be handled), and carry out controlled tests to see if, or how well requirements have been addressed. For testing the security of the implemented WS, [77] offers a number of strategies and guidelines. These are both generic (i.e. just highlight the use of test suits, test patterns and so on), and targeted (i.e. focus on specifics such as testing application data). Vulnerability analysis is another aspect that needs to be addressed in detail during testing. For this task, companies can refer to [78] regarding various guidelines on software vulnerability analysis for WS. These include checking for cross-site scripting, services traversal, DoS attacks, and access validation attacks. Actually, businesses can reuse the original listing of threats that were factored into security requirements determination, and conduct penetration tests against services to evaluate how well the implemented security addresses these threats. Particularly keen companies, or businesses that lack the expertise internally may consider employing security companies to conduct these tests. This decision however, should not be taken lightly as exposing systems to external parties demands great amounts of trust.

Processes, guidelines, and techniques are all essential in testing, but to enhance or at least ease this task, tool support would be ideal. Unfortunately, there has not been much notable work in this area as yet; this is likely because WS testing is a discipline still in its infancy [77]. One tool that has surfaced (and been referenced in academia [69] for its use) however, is wsChess [79]. WsChess is described by its makers as a freely available toolkit for WS assessments and defense [79]. Authors in [69] give a brief example of how wsChess can be used to probe for vulnerabilities and formulate attacks against Web services. To assess Web applications that may constitute part of the WS systems, a number of tools are available. [80] is a perfect source of information on these tools and an objective discussion of their aims. These industry-based researchers also outline a taxonomy of tools which encompass prime testing areas such as source-code analyzers, Web application scanners, runtime analysis tools, and configuration management tools, to assist companies with their tool selection [80]. The tools and techniques supplied here and those available from other sources should be used wherever possible to enable for a thorough, adequate testing of the developed Web services systems. This testing activity completes the Development and Testing phase; the outputs of this phase are the *developed services and systems, low-level process designs, service-level interaction definitions, security architecture design, a semantics framework, the standards and technologies of choice to implement the WS interactions, and the low-level requirements (functional, security and QoS agreements).*

3.10. Maintenance Phase

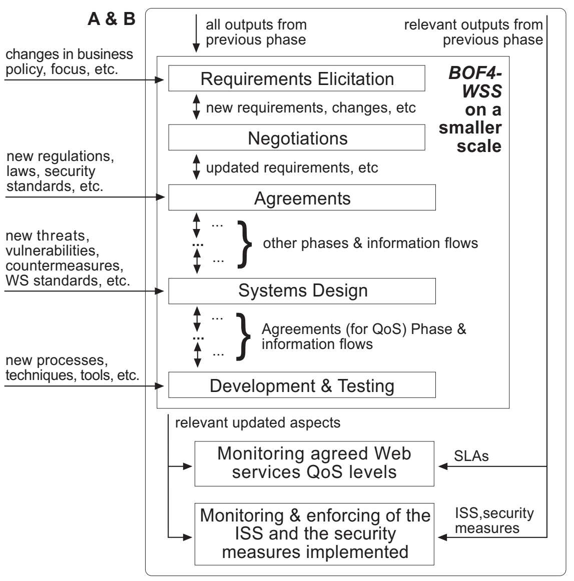

A & B
all outputs from previous phase
relevant outputs from previous phase
changes in business policy, focus, etc.
Requirements Elicitation
BOF4-WSS on a smaller scale
new requirements, changes, etc
Negotiations
new regulations, laws, security standards, etc.
updated requirements, etc
Agreements
new threats, vulnerabilities, countermeasures, WS standards, etc.
other phases & information flows
Systems Design
Agreements (for QoS) Phase & information flows
new processes, techniques, tools, etc.
Development & Testing
Maintenance Phase
relevant updated aspects
Monitoring agreed Web services QoS levels
SLAs
Monitoring & enforcing of the ISS and the security measures implemented
ISS,security measures

Figure 19. Workflow model of the Maintenance phase

Having developed this comprehensive, multilayered security solution, its upkeep becomes the next crucial undertaking. BOF4WSS addresses this and other typical monitoring and preservation tasks in the **Maintenance phase** shown by Figure 19. It is important to understand that this phase is a continuous one (unlike the others which have clearly defined endpoints), and will last for the lifetime of the implemented systems. Specifically, this stage will involve continuous functional and quality-based system enhancements, but additionally will stress the continued updating and enforcement of security measures, both in developed systems and the overarching ISS. To facilitate the required maintenance activities, the framework strongly suggests that businesses form cross-enterprise maintenance and monitoring teams. Ideally, the majority of the persons chosen should be members of the teams that participated in the full BOF4WSS process. The advantage of this is the experience they bring and the avoidance of having to deal with too steep of a learning curve. One team already mentioned in BOF4WSS is the security team from the first agreements stage. These personnel are entrusted with the responsibility of monitoring the *internal* and *external* environments, and considering new threats, laws, and security requirements, and how these will be included in system and WS interactions updates.

When considering the updating activities of the Maintenance phase, companies must be extremely careful in how they make changes and updates to cross-enterprise agreements, directives, and systems. This is true even when these updates are agreed by both companies involved. Changes should not be made in isolation without first analyzing what effects they might have on other system aspects, and whether respective updates to these other aspects would be necessary. It is for this reason that a smaller scale BOF4WSS process is suggested in this phase (see Figure 19). By reiterating this process for new needs in the form of updates and changes, it allows modifications to be made in a structured and controlled context. Repetition of previous phases is not uncommon during software maintenance as noted in [23]. Because the BOF4WSS process has been discussed in detail previously, it is not covered here or in Figure 19. Instead, Figure 19 is used to display some of the key **new** inputs (i.e. in addition to the ones outlined in previous phases) which are very likely to be incurred. Examples are, changes in the business policies (reflecting possibly new goals, aims), new regulations (therefore new, mandatory security for interactions and systems), new threats and vulnerabilities (these need to be assessed and addressed), and new techniques and tools (these may facilitate easier development or even system testing).

The other tasks depicted by Figure 19 focus on monitoring in general, but specifically as it relates to (i) QoS levels, and (ii) the monitoring and enforcement of the ISS and implemented security measures. In the first task, the goal is to take the actual service levels (recorded by management, auditing or tracking software added in the Development phase) and compare them with the SLAs and QoS agreements made earlier, to determine if quality requirements are being met by parties. Particularly of interest to companies will be aspects that affect general Web service performance levels such as service response times, and system downtime and latency. SLAs also are the point of reference that dictates the penalties and options for recourse if the agreed levels are not fulfilled.

The second task deals with the monitoring and enforcement of the ISS and the security measures implemented. Security, in every regard, is a constant process. Authors in [37] when they describe information security liken it to a journey, not a destination. Within this journey, monitoring of the implemented security mechanisms and rules is critical. The reason for this is that new threats may surface, new attacks might be launched, and consequently, there needs to be constant monitoring to detect (and initiate a reaction to) these advances. Again, the output (e.g. audit trails, logs) from monitoring and detection softwares is used in this activity. Beyond tracking new threats, and attacks, it is imperative that companies use this information to identify areas where directives and measures may need to be enforced. This relates to both *internal* and *external* to a company. Therefore, in addition to monitoring and following up on internal security concerns, business partners should be periodically assessed to ensure that they are maintaining the agreed levels of security. These levels can be found in the ISS, and systems design documentation amongst other documents. Some of the common options to assess the security posture of partners has been covered before (in the first agreements phase) and includes audits (by a third party possibly), and on-site visits.

A final noteworthy aspect shown in Figure 19, is that as smaller scale BOF4WSS processes are conducted to accommodate for updates, the final updates are then re-input into the respective monitoring tasks. This is done to keep the information used for monitoring as up-to-date and relevant as possible. This last task concludes the BOF4WSS process. Next, a brief summary and justification of BOF4WSS is presented. That section also identifies the main target group of businesses for which the framework is intended.

3.11. Summarizing BOF4WSS

As can be seen from the preceding in-depth discussion of BOF4WSS, the framework provides a detailed guidance model for inter-organizational cooperation. Beyond this, the next aim in this research (discussed in Section 4) is to drill down into the framework's specifics and provide a practical implementation base. This includes investigation into how stages of the architecture can be expanded, when or where can existing mechanisms be used, and lastly in the provision of suitable infrastructure and tool support to aid in framework use.

Reflecting on BOF4WSS in its entirety, specially with regard to its use by companies, it is obvious that this is not a process to be taken flippantly. In the design of this framework, not only were security practices within WS and business processes in general assessed, but also literature on joint business ventures such as the extended enterprise (e.g. [17]), and how security—beyond the technical layer—is reached, and maintained across enterprises there. With these factors in mind, the framework is thus aimed particularly towards businesses that *emphasize trust and medium-to-high levels of security, and expect long-term interactions as opposed to the short-term, highly dynamic, e-marketplace-type interactions also possible with WS. Ideally, a set of business partners in the early planning stages for a WS project will adopt BOF4WSS to create an agreed, communications security infrastructure*. Due to the long-term nature envisioned, it is not expected that companies will frequently enter or leave the business scenario, therefore scalability is not a critical issue. Should companies be added however, it is crucial that they go through some of BOF4WSS's phases. At that point, it will be up to existing businesses whether the new partners adopt the active security charters and infrastructure, or if they all recomplete key security-related framework phases.

In general, the framework tasks to be executed when new partners join will be very context dependent. For example, depending on the new company and its purpose, additional services may need to be created by all companies, or only a small subset of companies. The extent of the services necessary, or the companies that are required to make modifications to their systems, will then determine the level of systems development that is required using BOF4WSS. There may even be cases where new partners already have their systems exposed as services, and therefore technical integration is not a problem (therefore no need for in-depth emphasis on later framework phases). In situations like these however, existing companies may choose to more focus on initial phases of BOF4WSS i.e. identifying risks and negotiating on security actions, and then ensuring that companies share the same goals with regards to cross-enterprise security. The ISS would be very relevant in this regard.

To utilize this approach, companies will have to be prepared to work together and devote resources—financial and nonfinancial (e.g. time, skills, experience)—to this venture. Many changes in how the businesses worked before WS adoption will be necessary. However as stated in [66] concerning WS in general, "the potential benefits—both financial and strategic—to adopting Web services are sufficiently large to justify such [business] changes." The same fact is true when focusing on security specifically.

Another crucial factor supporting the highly involved approach to security central to BOF4WSS, is the emerging legislative requirement-base. These regulations (partially shown in [10]) demand that companies now look both *internal* and *external* (i.e. business relationships) in their considerations of security. In [6], authors commenting on the new security responsibilities in WS, state that "risks must be assessed and managed across a collection of organizations, which is a new and very challenging security responsibility". They also make the point that to ensure collective WS offerings between businesses are secure, elements such as strategies and structured approaches to security must be used [6]. All these requirements fuel the need for a security approach such as BOF4WSS.

3.12. Limitations

There are two known, noteworthy limitations of the framework. The first relates to the longevity of BOF4WSS and the perspective that it risks being outdated quickly. This is because BOF4WSS is arguably not as abstract as a framework/methodology should be. Therefore, even though identifying standards, laws, tools and technologies is beneficial as gives e-businesses detailed guidance and insight into online WS interactions, it ties the framework too closely with current practices. This is a valid concern and the only solution to it that is in line with the original aim of the framework is to update BOF4WSS periodically. This would allow updates in relevant laws, tools and so on, and also enable any structural changes to be made based on field tests and adopting companies' feedback. Updating frameworks (and even more abstract frameworks) is an accepted reality as is exemplified in the various versions of the industry accepted model, TOGAF [25]—currently up to version 9. Furthermore, considering the volatility of the online security field, the updating of all security-focused models is vital.

The second limitation results from the framework's basis on the Waterfall Model (WM). Even though this model is believed to be the most suitable (for reasons identified above), there are reservations about the time taken for overall project completion, and flexibility and turnaround time within individual phases. Possible ways to address these issues include attempting to incorporate quicker and more flexible development techniques within specific phases of the WM-based framework. Additional benefits with these techniques might also be attainable in the areas of project risk management (common with iterative methods), and purpose-built support tools (apparent in methods such as rational unified process). Hines et al. [81] provide a good start for this with regards to integrating agile methods in the WM. Such techniques will need to be evaluated in depth before being included in BOF4WSS however, to ensure that structure and benefits of the WM to large or critical system projects are not affected.

4. Conclusion and Future Work

In this paper, we extended the work in [1] by engaging in a detailed discussion of our cross-enterprise development methodology, BOF4WSS. This discussion included a step-by-step analysis of its nine phases, where we presented the activities involved, justified the guidance given, and highlighted how the activities proposed could aid in building the requisite security and trust across collaborating e-businesses. Throughout this work, our main contention was that because of the very nature of Web services technology, the security of interacting e-businesses was now a much broader, and more 'real-time' issue than ever before. The broad issue was as a result of the ease in which threats and attacks by way of WS, could propagate from poorly secured companies to the systems of their unsuspecting business partners. Whereas the 'real-time' issue refers to the speed in which attacks can spread between interacting companies.

The novelty of our approach is that it considers the full nature of WS, and its security implications (technical and otherwise); recognizes and targets the 'live' inter-organizational security issue now faced by interacting e-businesses; and finally, promotes the use of a joint approach, where businesses work closely together and follow a structured process, to achieve enhanced levels of security and trust across partners. Our approach therefore aims to be a facilitator of, instead of a panacea to the security of e-businesses which use WS. Similarly, the goal is to provide another important piece of the security puzzle that is complementary to existing approaches.

In future work, the first area of interest is the provisioning of systems support for the framework itself. As can be seen, BOF4WSS is a quite extensive process. To aid in its use therefore, we intend to further examine each stage and the interface between stages, and provide support wherever applicable. One area already identified (through an initial exploratory investigation), concerns the outputs the *individual* Requirements Elicitation phase and their immediate usefulness as inputs to the *joint* Negotiations phase. This is of interest because of the inherent difficulty (relating to security actions/requirements format, prioritization schemes, and so on) in attempting to quickly and easily compare, and negotiate on the high-level security actions/requirements of each business as they are passed from the Requirements Elicitation, to Negotiations phases. To address this issue we are currently investigating into a tool-based approach to streamline security actions/requirements comparison and negotiations. The first steps in our work can be seen in [82], [83].

Once the framework with added systems support is complete, our next goal will be its evaluation to determine how well BOF4WSS's aims of enhancing security and trust across businesses, have been achieved. Noting the complexity and scope of the framework, the evaluation process necessary is far from trivial. In an attempt at a thorough evaluation which appreciates these difficulties, a three-stage process is planned.

First, industry-based security professionals would be interviewed to get their views on the suitability, and application of the framework. This stage provides useful and quick feedback from a variety of experienced and expert sources, on the framework and the activities it proposes. The next stage adopts a more practical perspective and focuses an evaluation of the systems support developed, to ascertain the actual degree of support supplied to the framework; specific scenarios are envisioned for use, which test numerous aspects of tool support. Initial work in this area can be seen in [84].

The last stage would be the full application of BOF4WSS to a real-life case scenario to critically evaluate its suitability and strength in achieving its goals. This would involve engaging a small set of companies to use the framework in their business scenario, then monitoring them, and constructing a case study from the observations, difficulties, uses and so on. The case could then be analyzed in-depth to make inferences on the applicability and effectiveness of the framework in real-world scenarios. This three-stage process would enable key aspects of BOF4WSS to be evaluated and substantiated conclusions made on its proposals.

References

[1] J. R. Nurse and J. E. Sinclair, "BOF4WSS: A Business-Oriented Framework for Enhancing Web Services Security for e-Business," in *4th International Conference on Internet and Web Applications and Services (ICIW) 2009*. IEEE Computer Society, 2009, pp. 286–291.

[2] J. Zhang, "Trustworthy web services: Actions for now," *IT Professional*, vol. 7, no. 1, pp. 32–36, 2005.

[3] M. Chen, "An analysis of the driving forces for web services adoption," *Information Systems and e-Business Management*, vol. 3, no. 3, pp. 265–279, 2005.

[4] M. P. Papazoglou, *Web Services: Principles and Technology*. Harlow, Essex: Prentice Hall, 2007.

[5] A. Singhal, T. Winograd, and K. Scarfone, "Guide to secure web services (NIST Special Publication 800-95)," National Institute of Standards and Technology (NIST), Tech. Rep., 2007.

[6] B. Hartman, D. J. Flinn, K. Beznosov, and S. Kawamoto, *Mastering Web Services Security*. Indianapolis: Wiley, 2003.

[7] R. J. Boncella, "Web services and web services security," *Communications of the Association for Information Systems*, vol. 14, no. 18, pp. 344–363, 2004.

[8] A. Charfi and M. Mezini, "Using aspects for security engineering of web service compositions," in *IEEE International Conference on Web Services*, Orlando, 2005, pp. 59–66.

[9] C. Gutiérrez, E. Fernández-Medina, and M. Piattini, "PWSSec: Process for web services security," in *IEEE International Conference on Web Services*, Chicago, IL, 2006, pp. 213–222.

[10] C. Steel, R. Nagappan, and R. Lai, *Core Security Patterns: Best Practices and Strategies for J2EE^{TM}, Web Services, and Identity Management*. Prentice Hall PTR, 2005.

[11] P. Wang, K.-M. Chao, C.-C. Lo, C.-L. Huang, and M. Younas, "A fuzzy outranking approach in risk analysis of web service security," *Cluster Computing*, vol. 10, no. 1, pp. 47–55, 2007.

[12] W.-J. van den Heuvel, K. Leune, and M. P. Papazoglou, "EFSOC: A layered framework for developing secure interactions between web-services," *Distributed Parallel Databases*, vol. 18, no. 2, pp. 115–145, 2005.

[13] B. Schneier, *Secrets and Lies: Digital Security in a Networked World*. Indianapolis: Wiley, 2004.

[14] S. Fischer and C. Werner, "Towards service-oriented architectures," in *Semantic Web Services: Concepts, Technologies, and Applications*, R. Studer, S. Grimm, and A. Abecker, Eds. Berlin: Springer-Verlag, 2007, pp. 15–24.

[15] T. Ishaya and J. R. Nurse, "Cross-enterprise policy model for e-business web services security," in *Information Security and Digital Forensics*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, D. Weerasinghe, Ed. Heidelberg: Springer, 2010, vol. 41, pp. 163–171.

[16] T. Tsiakis, E. Evagelou, G. Stephanides, and G. Pekos, "Identification of trust requirements in an e-business framework," in *The 8th WSEAS International Conference on Communications*, Athens, Greece, 2004.

[17] E. W. Davis and R. E. Spekman, *The Extended Enterprise: Gaining Competitive Advantage through Collaborative Supply Chains*. Upper Saddle River, NJ: FT Prentice Hall, 2004.

[18] C. Van Slyke and F. Bélanger, *E-Business Technologies: Supporting the Net-Enhanced Organization*. New York: Wiley, 2003.

[19] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services: Concepts, Architectures and Applications*. Berlin: Springer-Verlag, 2004.

[20] M. Khalifa and J. M. Verner, "Drivers for software development method usage," *IEEE Transactions on Engineering Management*, vol. 47, no. 3, pp. 360–369, 2000.

[21] H.-J. Bullinger, K.-P. Fähnrich, and T. Meiren, "Service engineering—methodical development of new service products," *International Journal of Production Economics*, vol. 85, no. 3, pp. 275–287, 2003.

[22] S. Chatterjee, "The waterfall that won't go away," *SIGSOFT Softw. Eng. Notes*, vol. 35, no. 1, pp. 9–10, 2010.

[23] I. Sommerville, *Software Engineering*, 8th ed. Essex: Pearson Education Ltd., 2007.

[24] H. Van Vliet, *Software Engineering: Principles and Practice*, 3rd ed. Chichester: John Wiley & Sons Ltd., 2008.

[25] The Open Group, "TOGAF^{TM}Version 9," 2009, http://www.opengroup.org/togaf/ (Accessed 8 February 2010).

[26] J. Sherwood, A. Clark, and D. Lynas, *Enterprise Security Architecture: A Business-Driven Approach*. San Francisco, CA: CMP Books, 2005.

[27] O. Demirörs, Ç. Gencel, and A. Tarhan, "Utilizing business process models for requirements elicitation," in *The 29th Conference on EUROMICRO*. IEEE, 2003, pp. 409–412.

[28] F. Hartman and R. A. Ashrafi, "Project management in the information systems and information technologies industries," *Project Management Journal*, vol. 33, no. 3, pp. 5–15, 2002.

[29] M. P. Papazoglou and P. Ribbers, *e-Business: Organizational and Technical Foundations*. Chichester, West Sussex: John Wiley & Sons Ltd., 2006.

[30] UK Department of Business, Enterprise and Regulatory Reform (BERR), "2008 information security breaches survey," 2008,

http://www.pwc.co.uk/pdf/BERR_2008_Executive_summary.pdf (Accessed 8 February 2010).

[31] R. S. Coles and R. Moulton, "Operationalizing IT risk management," *Computers & Security*, vol. 22, no. 6, pp. 487–493, 2003.

[32] G. Stoneburner, A. Goguen, and A. Feringa, "Risk management guide for information technology systems (NIST special publication 800-30)," National Institute of Standards and Technology (NIST), Tech. Rep., July 2002.

[33] C. Alberts and A. Dorofee, *Managing Information Security Risks : The OCTAVE Approach*. Boston: Addison-Wesley, 2003.

[34] F. den Braber, I. Hogganvik, M. S. Lund, K. Stølen, and F. Vraalsen, "Model-based security analysis in seven steps - a guided tour to the CORAS method," *BT Technology Journal*, vol. 25, no. 1, pp. 101–117, 2007.

[35] J. S. Tiller, *The Ethical Hack: A Framework for Business Value Penetration Testing*. Boca Raton, FL: Auerbach Publications, 2005.

[36] J. Misrahi, "Validating your business partners," in *Information Security Management Handbook*, 6th ed., H. F. Tipton and M. Krause, Eds. Boca Raton, FL: Auerbach Publications, 2007, vol. 1, pp. 123–131.

[37] A. Calder and S. Watkins, *IT Governance: A Manager's Guide to Data Security and ISO 27001/ISO 27002*, 4th ed. London: Kogan Page Limited, 2008.

[38] K. C. Laudon and C. G. Traver, *E-commerce: Business, Technology, Society*, 3rd ed. New Jersey: Prentice Hall, 2007.

[39] W. H. Baker, G. E. Smith, and K. J. Watson, "Information security risk in the e-supply chain," in *E-Supply Chain Technologies and Management*, Q. Zhang, Ed. Hershey, PA: Idea Group Inc., 2007, pp. 142–161.

[40] S. Röhrig and K. Knorr, "Security analysis of electronic business processes," *Electronic Commerce Research*, vol. 4, no. 1-2, pp. 59–81, 2004.

[41] S. Nachtigal, "eBPSM: A new security paradigm for e-business organisations (e-business process security model)," in *The Ninth International Conference on Electronic commerce*, Minneapolis, MN, 2007, pp. 101–106.

[42] G. M. Giaglis, "A taxonomy of business process modeling and information systems modeling techniques," *International Journal of Flexible Manufacturing Systems*, vol. 13, no. 2, pp. 209–228, 2001.

[43] R. S. Aguilar-Savén, "Business process modelling: Review and framework," *International Journal of Production Economics*, vol. 90, no. 2, pp. 129–149, July 2004.

[44] P. Mayer, A. Schroeder, and N. Koch, "A model-driven approach to service orchestration," in *IEEE International Conference on Services Computing*. Honolulu, Hawaii: IEEE Computer Society, 2008, pp. 533–536.

[45] D. Skogan, R. Groenmo, and I. Solheim, "Web service composition in UML," in *Eighth IEEE International Enterprise Distributed Object Computing Conference*, 2004, pp. 47–57.

[46] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ: Pearson Education, 2005.

[47] J. Recker, "Process modeling in the 21st century (BPTrends columns & articles)," May 2006, http://www.bptrends.com/publicationfiles/05-06-ART-ProcessModeling21stCent-Recker1.pdf (Accessed 8 February 2010).

[48] Object Management Group (OMG), "UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms, Version 1.1," n.d., http://www.omg.org/cgi-bin/doc?formal/08-04-05.pdf (Accessed 8 February 2010).

[49] A. Rodríguez, E. Fernández-Medina, and M. Piattini, "Security requirement with a UML 2.0 profile," in *The First International Conference on Availability, Reliability and Security*, 2006, pp. 670–677.

[50] ——, "A BPMN extension for the modeling of security requirements in business processes," *IEICE - Transactions on Information and Systems*, vol. E90-D, no. 4, pp. 745–752, 2007.

[51] M. Schumacher and U. Roedig, "Security engineering with patterns," in *The 8th Conference on Pattern Languages of Programs (PLoP)*, Monticello, Illinois, 2001.

[52] World Wide Web Consortium (W3C), "Web services choreography description language version 1.0," 2005, http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/ (Accessed 8 February 2010).

[53] G. Decker, O. Kopp, F. Leymann, and M. Weske, "BPEL4Chor: Extending BPEL for modeling choreographies," in *IEEE International Conference on Web Services*. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 296–303.

[54] G. Decker, O. Kopp, F. Leymann, K. Pfitzner, and M. Weske, "Modeling service choreographies using BPMN and BPEL4Chor," in *Advanced Information Systems Engineering*, ser. Lecture Notes in Computer Science, Z. Bellahsène and M. Léonard, Eds. Heidelberg: Springer, 2008, vol. 5074, pp. 79–93.

[55] A. Barker and J. van Hemert, "Scientific workflow: A survey and research directions," in *Parallel Processing and Applied Mathematics*, ser. Lecture Notes in Computer Science, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski, Eds. Heidelberg: Springer, 2008, vol. 4967, pp. 746–753.

[56] J. Mendling and M. Hafner, "From WS-CDL choreography to BPEL process orchestration," *Journal of Enterprise Information Management*, vol. 21, no. 5, pp. 525–542, 2008.

[57] Object Management Group (OMG), "UML profile for schedulability, performance, and time specification, version 1.1," n.d., http://www.omg.org/cgi-bin/doc?formal/05-01-02.pdf (Accessed 8 February 2010).

[58] M. Cambronero, G. Díaz, J. Pardo, V. Valero, and F. L. Pelayo, "RT-UML for modeling real-time web services," in *IEEE Services Computing Workshops*, 2006, pp. 131–139.

[59] Anonymous, "WS-CDL Eclipse," n.d., http://wscdl-eclipse.sourceforge.net/main.htm (Accessed 8 February 2010).

[60] Pi4 Technologies Foundation, "Pi4SOA," n.d., http://pi4soa.wiki.sourceforge.net/ (Accessed 19 December 2008).

[61] H. Foster, "WS-Engineer 2008: A service architecture, behaviour and deployment verification platform," in *Service-Oriented Computing ICSOC 2008*, ser. Lecture Notes in Computer Science, A. Bouguettaya, I. Krueger, and T. Margaria, Eds. Heidelberg: Springer, 2008, vol. 5364, pp. 728–729.

[62] D. Z. Garcia and M. B. Felgar de Toledo, "A policy approach supporting web service-based business processes," in *First Brazilian Workshop on Business Process Management*

(WBPM 2007), Gramado, RS, Brazil, 2007.

[63] J. Jürjens, *Secure Systems Development with UML*. Berlin: Springer-Verlag, 2005.

[64] T. Lodderstedt, D. A. Basin, and J. Doser, "SecureUML: A UML-based modeling language for model-driven security," in *UML 2002: The Unified Modeling Language*, ser. Lecture Notes in Computer Science, J.-M. Jézéquel, H. Hussmann, and S. Cook, Eds. Heidelberg: Springer, 2002, vol. 2460, pp. 426–441.

[65] Web Services Interoperability Organization (WS-I), "WS-I," n.d., http://www.ws-i.org/ (Accessed 8 February 2010).

[66] S. Chatterjee and J. Webber, *Developing Enterprise Web Services: An Architect's Guide*. Upper Saddle River, NJ: Prentice Hall PTR, 2004.

[67] W. D. Yu, R. B. Radhakrishna, S. Pingali, and V. Kolluri, "Modeling the measurements of QoS requirements in web service systems," vol. 83, no. 1, 2007, pp. 75–91.

[68] Web Services Interoperability Organization (WS-I), "Security challenges, threats and counter-measures version 1.0," 2005, http://www.ws-i.org/Profiles/BasicSecurity/SecurityChallenges-1.0.pdf (Accessed 8 February 2010).

[69] N. Sidharth and J. Liu, "IAPF: A framework for enhancing web services security," in *31st Annual International Computer Software and Applications Conference (COMPSAC)*, Beijing, China, 2007, pp. 23–30.

[70] D. Z. Garcia and M. B. Felgar de Toledo, "A policy-based web service infrastructure for autonomic service integration," in *First Latin American Autonomic Computing Symposium (LAACS)*, Campo Grande, MS, 2006.

[71] A. Keller and H. Ludwig, "The WSLA framework: Specifying and monitoring service level agreements for web services," *Journal of Network and Systems Management*, vol. 11, no. 1, pp. 57–81, 2003.

[72] International Business Machines (IBM) Corp., "Emerging Technologies Toolkit (ETTK) for Web Services," n.d., http://www.alphaworks.ibm.com/tech/ettk (Accessed 8 February 2010).

[73] O. Zimmermann, M. R. Tomlinson, and S. Peuser, *Perspectives on Web Services: Applying SOAP, WSDL, and UDDI to Real-World Projects*. Berlin: Springer, 2003.

[74] C. Ouyang, M. Dumas, A. H. M. ter Hofstede, and W. M. P. van der Aalst, "From BPMN Process Models to BPEL Web Services," in *IEEE International Conference on Web Services*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 285–292.

[75] M. Cambronero, G. Díaz, J. Pardo, and V. Valero, "Using UML diagrams to model real-time web services," in *Second International Conference on Internet and Web Applications and Services*, 2007.

[76] M. Tatsubori, T. Imamura, and Y. Nakamura, "Best-practice patterns and tool support for configuring secure web services messaging," in *IEEE International Conference on Web Services*. Athens, Greece: IEEE Computer Society, 2004, pp. 244–251.

[77] A. Barbir, C. Hobbs, E. Bertino, F. Hirsch, and L. Martino, "Challenges of testing web services and security in SOA implementations," in *Test and Analysis of Web Services*, L. Baresi and E. Di Nitto, Eds. Heidelberg: Springer, 2007, pp. 395–440.

[78] W. D. Yu, D. Aravind, and P. Supthaweesuk, "Software vulnerability analysis for web services software systems," in *IEEE Symposium on Computers and Communications*. IEEE, 2006, pp. 740–748.

[79] Net-Square Solutions Pvt. Ltd., "wsChess - web services assessment and defense toolkit," n.d., http://net-square.com/wschess/index.shtml (Accessed 8 February 2010).

[80] M. Curphey and R. Arawo, "Web application security assessment tools," *IEEE Security & Privacy*, vol. 4, no. 4, pp. 32–41, 2006.

[81] L. Hines, S. Baldwin, M. Giles, and J. Peralta, "Implementing agile development in a waterfall project," 2009, http://www.ibm.com/developerworks/websphere/techjournal/0907_hines/0907_hines.html (Accessed 8 February 2010).

[82] J. R. Nurse and J. E. Sinclair, "Supporting the comparison of business-level security requirements within cross-enterprise service development," in *Business Information Systems*, ser. Lecture Notes in Business Information Processing, W. Abramowicz, Ed. Heidelberg: Springer, 2009, vol. 21, pp. 61–72.

[83] ——, "A Solution Model and Tool for Supporting the Negotiation of Security Decisions in E-Business Collaborations (to appear)," in *5th International Conference on Internet and Web Applications and Services (ICIW) 2010*. IEEE Computer Society, 2010.

[84] ——, "Evaluating the Compatibility of a Tool to Support E-Businesses' Security Negotiations (to appear)," in *The International Conference of Information Security and Internet Engineering (ICISIE 2010), part of World Congress on Engineering (WCE) 2010*, London, UK, 2010.

A Model-Driven Development Process and Runtime Platform for Adaptive Composite Web Applications

Stefan Pietschmann

Technische Universität Dresden
Faculty of Computer Science, Chair of Multimedia Technology
01062 Dresden, Germany
Stefan.Pietschmann@tu-dresden.de

***Abstract*—So far, little research has addressed composition and integration at the presentation layer of web applications. Service-oriented architectures provide uniform models for encapsulation and reuse of data and application logic in the form of web services, but this paradigm has not yet been applied to the presentation layer, impeding a *universal* composition of web applications. Thus, UIs are usually hand-crafted, lack flexibility and reusability, resulting in an expensive and onerous development process. We address these issues with a model-driven development process and a corresponding runtime architecture facilitating the universal, dynamic composition of web applications. Therein, user interface parts are as well provided "as a service" and can thus be selected, customized and exchanged with respect to the current context. We validated our approach using a prototypical implementation and a number of sample applications.**

***Keywords*-web engineering, model-driven development, composite applications, mashups, user interfaces, adaptive hypermedia**

I. Introduction and Motivation

The WWW has evolved into a rather stable, universal software platform. Numerous applications are provided as "Software as a Service" (SaaS) over the Internet, leveraging the location- and time-independent access as well as new business models like pay-per-use.

An enabling paradigm for this trend is the so-called *Programmable Web* [1]. Therein, data and application logic are provided in a decoupled and technology-independent fashion via generic service interfaces or APIs. Web-based applications can and will increasingly be composed from such distributed and reusable parts or services. A seminal[1] type of such *composite applications* are *mashups*, which create added value by combining web resources, e. g., data and logic from local or remote services. They allow for a shorter development cycle and thereby more situational applications that foster the *Long Tail* [4] of software needs. In this context, component-based concepts from academia for a more structured design process based on a "universal composition" have been proposed [5]. We build on this idea of composite applications and use the term synonymously to "user interface (UI) mashup" throughout this paper.

[1]According to Gartner, by 2010 mashups will be the predominant model (80%) for the development of composite enterprise applications [2]. Other institutes, e. g., Forrester, also underline their growing importance [3].

The underlying, service-oriented approach has simplified the integration at the data and application layers through standardization efforts and frameworks. Web services allow for the technology-independent encapsulation and deployment of functionality, which facilitates flexibility of the business logic by their exchange and custom configuration. However, *presentation integration* has not yet been addressed by research adequately [6], so there is a lack of comparable efforts for the presentation layer. Current concepts and technologies lack proper reuse mechanisms and interoperability.

Due to mobile and decentralized access to web applications, mashup developers face the problem of the heterogeneity of users and devices. To fully exploit the advantage of time-, location- and device-independent access, web applications need to adapt to the current situation, i. e., context (location, screen resolution, etc.), while preserving usability standards. This has dramatically complicated UI development. However, research in this field is still largely restricted to basic hypermedia systems and suffers from the "open corpus problem" [7]: Those approaches only work well for closed systems with predefined structures and preindexed or annotated documents, but fail when it comes to context-aware Rich Internet Applications (RIA) and unforeseeable, dynamic content. Thus, users are yet again facing "one-size-fits-all" UI solutions, which seems like a step backwards from the achievements of the "Adaptive Hypermedia" community over the last decade.

Additionally, UI developers are confronted with a myriad of (not necessarily new) programming languages, web frameworks and technologies to choose from. These offer a high level of UI individualization as opposed to classical desktop applications relying on uniform window-based UI libraries. This degree of freedom is an advantage, but it often results in inconsistent interaction metaphors between different applications, in low usability and thus confused users. Overall, development and maintenance of user interfaces still add up to about 70% of the overall software development [8] – a problem which is further intensified by challenge of context-awareness described above.

To address the above-mentioned problems, we strive for a model-driven, platform-independent development of composite web applications, and their context-aware execution by a service-oriented UI integration and composition system [9], [10]. By extending the service-oriented approach from the business layer to the presentation layer, we facilitate reusability and flexibility therein and thus simplify the development of context-aware rich web applications.

This article is structured as follows. In Section II we discuss relevant work related to web-based UI composition models and systems. Section III describes our model-driven approach to develop mashup applications, providing details on the underlying component and composition models. The concept of the related composition infrastructure, including the deployment process and run time integration of UI components, is presented in Section IV. After a brief discussion of implementation details, Section V illustrates the practicability of our approach by means of two sample applications. Section VI concludes this article and outlines future research directions.

II. Related Work

As described in the last section, the Web lacks uniform models for web-based components as well as open models and systems for their composition. We therefore present and discuss related composition and integration efforts.

Research in the field of composite web applications with focus on *presentation integration*, faces five fundamental challenges [11]. These comprise the development of a *component model*, a *composition model*, adequate inter-component *communication styles*, as well as mechanisms for *discovery and binding* and *visualization* of the UI components. In this article we focus on the first three issues, since run time discovery of user interface components is still ongoing work and visualization is usually carried out by the browser.

There already exist numerous component and composition models for the Web. The problem with both client-side (JavaScript frameworks, Applets, ActiveX Controls, Flash, etc.) as well as server-side (Portlets, ASP.NET Web Parts, etc.) solutions is that they all imply their very own interfaces, communication models, and, moreover, technological platforms. We aim for a uniform approach wrapping technology-specifics behind a generic component interface, comparable to web services. Also, with regards to popular UI frameworks such as the YUI library, we provide more complex, high-level components with integrated presentation and application logic.

As an example, *Portlets* are one of the oldest and most mature models for UI integration on the web. By composing them within a *Web Portal*, users are presented a consistent interface of several integrated UI parts forming a "Single Point of Access" for different back end services [12]. Thus, a Web Portal constitutes both a composition framework and common presentation layer of Service-Oriented Architectures [13], [14]. However, the use of portals comes with a number of disadvantages [15]. In contrast to more modern approaches they are limited to the server-side aggregation and communication of portlets. Despite the standardization of portlets, other portal characteristics, e. g., the layout, remain vendor-specific. Furthermore, with respect to our requirements, sophisticated concepts for a model-driven, platform-independent development and for the adaptation of portal applications are missing.

As already mentioned in the last section, *mashups* are an emerging, lightweight trend for the development of composite applications. However, the majority of current tools and platforms are still in their infancy [16] and concentrate on the integration of data and application logic, thus overlapping with composition systems like portals [17]. Integration is usually based on the *Piping* style [18] and supported by (often visual) composition languages and tools (Yahoo Pipes[2], JOpera [19]). User interface development is not [19] or insufficiently supported, as in Microsoft Popfly[3]. Enterprise-oriented platforms equally require authors to program the UI traditionally with the help of JavaScript libraries (JackBe Presto[4]) or WYSIWYG editors (Serena Business Mashups[5]).

Lately, an alternative SOA composition principle to portals has been proposed for the presentation layer [14]. Therein, so-called *widgets* or *mashlets* [15] are combined, each representing a self-contained application with its own user interface. Standardization of such "web parts" is currently pushed, e. g., by the W3C [20] and Google [21].

Few scientific approaches have addressed the challenges of *presentation integration* [11], one of the first being the "Presentation Integration Framework" *Mixup* [22]. Using a *Composition Middleware*, heterogeneous presentation components are assembled based on a declarative composition description and platform-specific adapters. Rather than adapters, our approach uses a generic wrapper that provides platform-specific UI components as a service. Thereby, components can be distributed and exchanged at run time, while in [22], [23] only design time composition of locally available components is supported. Such components are loosely coupled by dividing the component and composition models, and by using *publish-subscribe* mechanism for communication. As the *Mashup Component Model* presented in [24], Mixup only composes portlet-like components that constitute full applications. Thus, the approach lacks a separation of the traditional application layers and impedes UI flexibility, i. e., adaptivity, at run time.

[2]http://pipes.yahoo.com/
[3]http://www.popfly.ms/
[4]http:/www.jackbe.com/
[5]http://www.serena.com/geo/de/products/business-mashups/

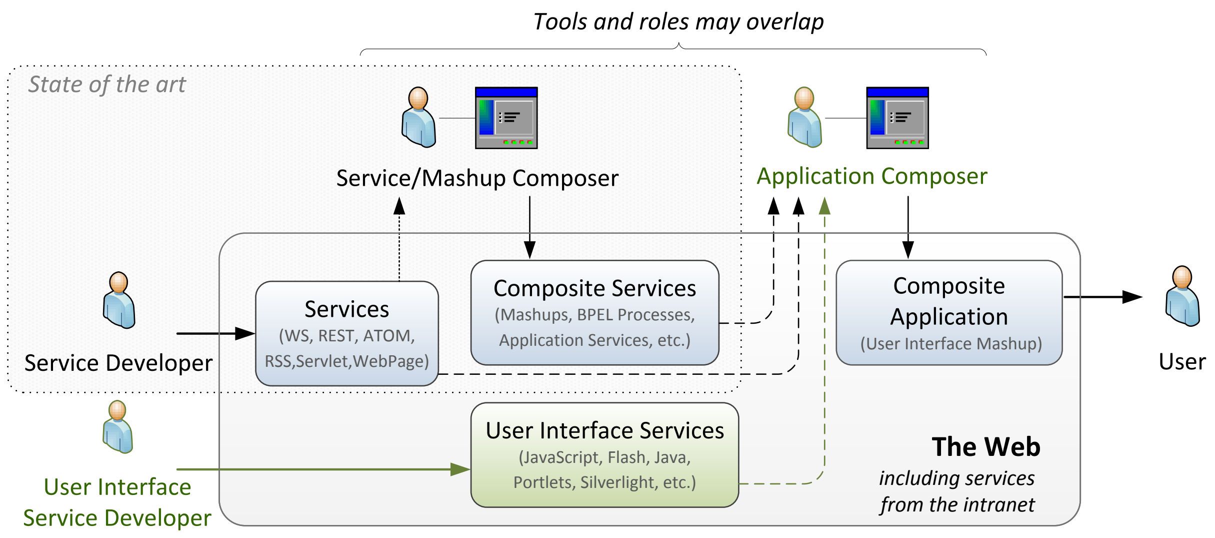

Tools and roles may overlap
State of the art
Service/Mashup Composer
Application Composer
Service Developer
Services
(WS, REST, ATOM, RSS,Servlet,WebPage)
Composite Services
(Mashups, BPEL Processes, Application Services, etc.)
Composite Application
(User Interface Mashup)
User
User Interface Service Developer
User Interface Services
(JavaScript, Flash, Java, Portlets, Silverlight, etc.)
The Web
including services from the intranet

Figure 1. Role model of the UI mashup development

Within the follow-up project mashArt [5] the component model was extended to be "universal" for UI, application and data building blocks. A fixed client-server infrastructure is provided, combining the client-side Mixup platform with a server-side part, whose necessity can be seen as an disadvantage [25]. Means for specifying control flow and adaptivity of an application as well as their support by the runtime environment are missing. Most importantly, in contrast to our approach, both Mixup and mashArt do not offer a platform-independent, model-driven development.

An alternative approach for service composition at the presentation layer is proposed by *ServFace* [26]. Herein, the user interface is generated from UI annotations of the orchestrated services beneath at design time. Dynamic context-awareness is not considered in this generation process, though. Moreover, such a transformation usually results in simplistic, form-based UIs, while we aim for the integration of rich, potentially multi-modal UI components that can undergo functional and usability tests prior to deployment.

As can be seen, there exist promising concepts for the integration and composition of web-based services and resources. However, due to the lack of (de-facto) standards for the description of components and compositions, those approaches suffer from interoperability problems. Compositions are usually based on proprietary models and lack support of desirable and increasingly necessary aspects like dynamic configuration and composition [16], control flow, and adaptivity of such applications. Traditional, model-driven concepts for the development of web applications, such as WebML [27] and OO-H [28] can not be directly applied to the mashup domain since they are too complex [5] and document-centric. As a result, the development of adaptive, context-aware mashup applications remains highly time- and money-consuming.

III. Model-Driven Development of Context-Aware, Composite Web Applications

To overcome the restrictions discussed above, we present a novel concept for the model-driven development and deployment of composite applications. Its central idea is the application of the service-oriented paradigm to the mashup presentation layer to support a universal composition on all application layers (data, application logic, and UI) and to simplify the overall development of context-aware, composite web applications. By using services to compose a web-based user interface, we facilitate reuse, customizability and technology-independence. We do this by (1) the encapsulation of generic, reusable web UI components (UIC), (2) their distributed deployment as so-called *User Interface Services* (UIS) and (3) their context-aware, dynamic invocation, configuration and integration with other mashup components, resulting in a fully service-oriented, composite web application.

Figure 1 gives an overview of the roles participating in the development of such a universal composite application. Traditional service developers (on the left) provide application- or domain-specific data as well as application logic via services, which are combined with the help of existing composition tools and frameworks, e. g., data mashup platforms and BPEL engines. Our concept of *User Interface Services* introduces reusable, service-based UI components that can be composed together with traditional services to a composite application, i. e., *user interface mashup*.

In this section we provide details on the underlying, generic component model for the encapsulation of reusable parts of a mashup, on their description and service-oriented provision, and on the composition model that defines an application as an arrangement of such components.

A. Component Model

To allow for a universal composition of an application, its constituent parts need to adhere to a generic component model. It defines the basic structure and interface of application building blocks, being either user interface parts, application logic or data providers. The model discussed in the following is a successor of our previous results presented in [9]. Following the principles introduced with web services, it does not dictate any internal component structure or format. However, it specifies fundamental characteristics of component interfaces, which are relied upon for their integration and the communication among themselves.

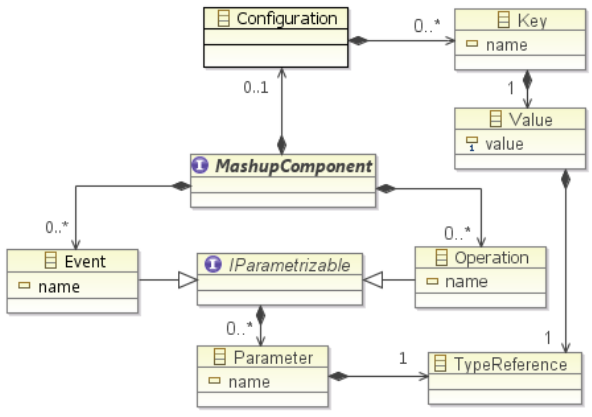


Figure 2. Universal Mashup Component Model

As illustrated by Figure 2, our model characterizes mashup components by three abstractions, namely configuration, event, and operation.

The *configuration* of a component resembles its visible state, which differs somehow from the service-oriented paradigm mentioned: While (web) services are loosely coupled and stateless by design, there is a special need for modeling a state of mashup components, as they form "a natural bridge between application services and data-oriented services" [5]. In our model, the state is represented by arbitrary key-value pairs, which are defined by the component developer and may contain any data that seems relevant to the component's surrounding. Examples include a graph's type (line, bar, pie) and a map's projection or type (normal, satellite, hybrid). In contrast to prevalent solutions, e. g., mashArt [5], our model supports complex value lists and trees by allowing XML-schema complex types to be able to map complex component-internal data structures more naturally to external properties.

To publish state changes to other components or the composition environment, a component can issue *events*. They may be triggered by user interaction (UI), time (logic) or notifications from external services (model). Besides a name, events can contain data in the form of a number of typed parameters, i. e., key-value pairs. Picking up the example from above, a map could issue an event *Location-Selected(String countryCode)* upon user interaction.

Operations are methods of a component which are triggered by events. They can include any functionality foreseen by the developer, such as state changes, calculations, or service requests. As an input, they consume parameters provided by the trigger events. As an example, an operation *getCountryInfo(String countryCode)* could be triggered by the aforementioned map event, which would result in a SOAP request by the component to a web information service providing information about the country. Its response would again be published as an event to be consumable by other components. As event and operation parameters don't always fit as nicely as in our example, we facilitate the definition of a mapping, so that parameter names and order become irrelevant to the wiring.

The component model presented here is specifically designed to support the loose coupling of application components based on a publish-subscribe mechanism (cf. Section III-B). Thus, events and operations form the basis of all application-internal communication.

Within a user interface mashup, we can distinguish different component types. Although they all comply with the component model presented above, they differ in the semantics, i. e., in the application layers they apply to. In the composition model discussed later, we differentiate between four types: At the topmost layer, *User Interface Components* (UIC) encapsulate parts of an application UI with corresponding presentation logic. A popular example would be an interactive map, as provided by Google or Yahoo. To support an efficient communication between components, we can employ *Logic Components* (LC). They provide means for data manipulations, e. g., transformation, filtering, or aggregation, so that parameters of events and operations fit, even in combinations unforeseen by their developers. Especially the use of complex data types, such as `Person`, necessitates data transformations to make parts of it (surname, age, gender) processable by operations of other components. Finally, at the lowest application level, *Service Components* wrap access to services providing data or complex application logics, e. g., via SOAP or REST.

Since we aim for flexible, context-aware user interfaces, deployment and description of UI components differ slightly from other component types. As already mentioned, **User Interface Services** (UIS) form an integral part of our concept. They facilitate the distributed deployment, technology-independent provision, and integration of above-mentioned UI components via a public service interface – something already common to back end services. A trend towards such services for the presentation layer can already be witnessed, prominent examples being Google's Maps or Visualization APIs, that offer the integration of configurable, interactive

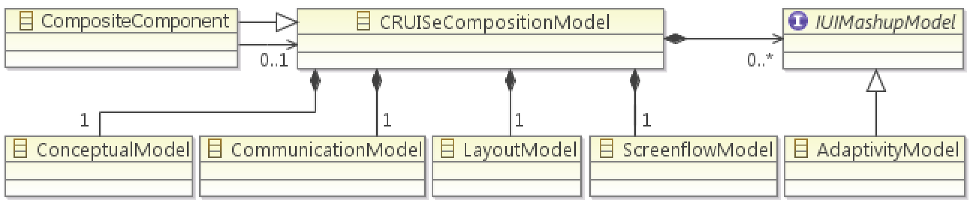


Figure 3. Main parts of the Composition Metamodel

maps and charts from a remote server. We generalize such techniques and propose a concept, in which the whole web application UI results from the integration and composition of UIS, or, more precisely, the UI components provided by them. UIS imply that back end development and UI design can be completely decoupled by using services, whose combination eventually results in a composite application.

Following this approach, we can equally bind web and UI services at runtime. We can even link the UIS selection and integration process with contextual information, such as user preferences, device capabilities, and the integration platform. This allows for context-aware, composite application UIs, but also implies that we need some kind of UIS description.

To dynamically select a UI service, an open interface description is needed to specify the characteristics of the corresponding UIC, including its semantics (purpose) and signature (operations and events). Moreover, since – opposed to traditional services – UI services provide components to be integrated (not executed remotely), those need to be wrapped with respect to the integration platform, i e., the technology and framework used. Consequently, a UI description needs to provide information on the compatibility and language bindings of a component.

Thus, similarly to service components described by WSDL or WADL, the interface of UIS is specified with the help of a *User Interface Service Description Language* (UISDL). It is an XML-based description of all information needed to select, integrate and use a UI component provided by the corresponding UIS. Therefore, it consists of two parts: the UISDL *class* description defines the generic interface of a component, i. e., its name, semantic concepts, license, and signature (properties, operations, events); the UISDL *binding* describes the mapping of a platform-specific component implementation to a class, including constructor information, references to required libraries, etc. UISDL metadata is stored in a *Component Registry* (cf. Figure 4) and used to dynamically match application requirements and context data with available UIS – a process which is discussed later in Section IV-B. Details on the UISDL are out of focus of this article and will be published separately.

In summary, the lightweight component model presented here describes reusable building blocks for the composition of mashup applications and supports synchronization on all its layers. Service components allow for the integration of arbitrary external services, their data being transformed by logic components and visualized by UI components. The latter are distributed and provided in a service-oriented fashion, the uniform component interface and the declarative interface description language UISDL hiding specific APIs and technologies. This approach enables interoperability and run time exchangeability of user interface parts and thus forms a basis for context-aware mashup user interfaces.

B. Composition Model

To compose web applications from components as discussed in the last section, a platform-independent model is needed to define all relevant aspects of an application, including the components used, the communication among them, the overall control flow as well as the layout of the user interface. Therefore, we have developed an extensible metamodel to describe all of the above aspects. Figure 3 provides a simplified overview of the *CRUISeCompositionModel* and its submodels, each describing a specific application aspect. As can be seen, any composition model also represents a component itself (*CompositeComponent*) and can thus be included in higher level compositions, either by reference or direct inclusion. The interface *IUIMashupModel* facilitates extensibility to model additional aspects, as exemplified with the *Adaptivity Model*. In the following, we give a brief overview of the main parts, i. e., submodels of our composition model. Further details are beyond the scope of this article and will be published separately.

The *Conceptual Model* contains all application-wide concepts. Most importantly, these comprise the components of an application – the different types presented above including specific configurations. Since their events and operations contain typed parameters, data type definitions (XML Schema) of all complex types used are included or referenced there. Additionally, reusable style classes can be modeled and applied to different UI components – comparable to CSS – to achieve a homogeneous *look and feel* of an application. Finally, two more *special* components round off the Conceptual Model: one allows for accessing context variables which are provided by a dedicated service at run time. Those context parameters can be connected with other application components, e. g., to constantly feed a map with the current location of a user. The other special component defines the external interface and execution of a composition. It includes application-wide variables, events to be issued at application initialization, and references to those events and operations of the composition that shall be accessible externally by a higher-level composition.

The layout of the composite application is described by the *Layout Model*. A developer can use multiple predefined layouts, following layout managers as common in several UI libraries. As an example, a *Grid Layout* with multiple rows and columns can be defined, each cell containing a certain mashup component (or another layout). Layouts can be hierarchically nested to achieve any desired arrangement of UI components.

Layouts are utilized within the *Screenflow Model* to describe several views (pages) of an application. Each view is represented by a specific layout, i. e., a visual state of the mashup with certain UI components visible. One view is the marked as initial, but plenty others can be defined. Transitions between views are triggered by events issued from components or the environment (the runtime system). This allows for very flexible and multi-step user interfaces.

The *Communication Model* supports arbitrary communication paradigms. We currently model communication based a *publish-subscribe*-mechanism, wherein channels connect publishers (events) with subscribers of information (operations). Developers can create manual mappings of channel and operation parameters. This is useful, when semantically different parameters differ in their names, or when event and operation parameters simply occur in a different order.

Finally, the *Adaptation Model* illustrates the extensibility of our metamodel. It was added to facilitate the definition of adaptation *aspects* crosscutting all of the above-mentioned models. Each aspect is triggered by an arbitrary event within the application, which leads to the validation of a condition defined by the composer. An aspect specifies both the part of the model to adapt, and an action which defines exactly what to do. As an example, it could change a layout, insert an additional component's configuration parameter, or exchange one component with another. With the help of this model, we can specify adaptations of a composite application at very different levels of granularity.

A complete composition model is transformed into an executable web application in a multi-step process, including a number of model-to-model and model-to-code transformations. They can be triggered dynamically by a client request, or statically during design. Alternatively, the model can be directly interpreted by dedicated runtime systems.

In conclusion, the composition metamodel presented provides the means to model all necessary aspects describing a service-oriented, composite application in a platform-independent fashion. Therefore, generic components (cf. Section III-A) at different application layers are integrated and linked by communication channels. The composition paradigm supports a seamless integration of service-oriented UI components, similarly to the integration of service-oriented back end logic and data. With the help of an adaptation model, any aspect of an application can be altered with respect to a particular context, resulting in highly flexible, self-adaptive compositions.

IV. CRUISe: A System Architecture for Adaptive User Interface Mashups

While the modeling approach presented in the previous section specifically focuses on design time of adaptive composite applications, this section presents an open and flexible system architecture for their dynamic composition and execution, called CRUISe. After a brief architectural overview, we present our concept in detail. First, we explain how an application is initialized including the dynamic integration and composition of remote UI components to an application UI. Then, we highlight details on its runtime adaptation.

A. Architectural Overview

Figure 4 gives an overview of the overall conceptual infrastructure of CRUISe. Its central concept is the use of distributed services for the dynamic composition of web applications to exploit the advantages of service-oriented architectures, like reusability, customizability and technology-independence at all application tiers, including the presentation layer. As mentioned in Section III-A, we do this by dynamically selecting, configuring, and integrating generic, reusable UIS into an application UI, and binding them to application-specific logic and service components.

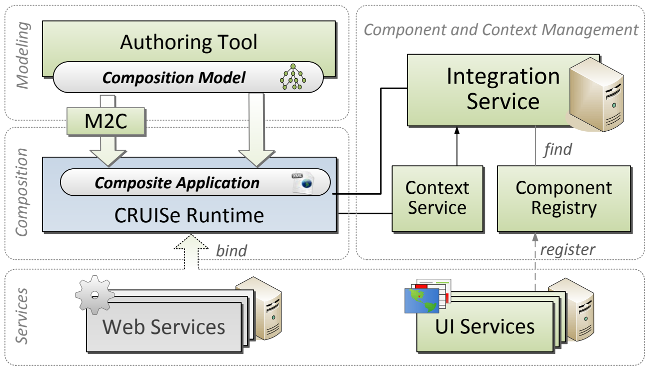


Figure 4. Architectural overview of the CRUISe infrastructure

On the left, Figure 4 illustrates the model-driven nature of our approach: Initially, a composite application is defined with the help of dedicated visual authoring tools and by means of the platform-independent *Composition Model*, presented in the last section. The latter is transformed to a platform-specific, executable application, either statically at design time, or triggered by a client request at run

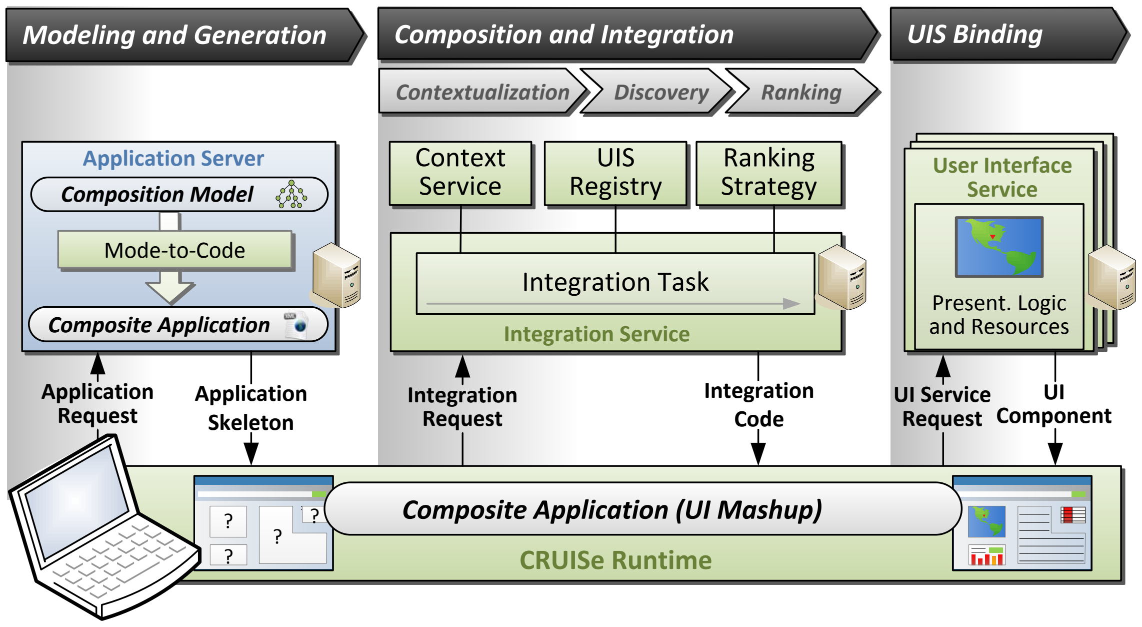


Figure 5. UI composition process

time. Alternatively, the model can be directly interpreted by dedicated run time environments.

The resulting *Composite Application* is executed by a runtime platform which provides all necessary means for controlling the application aspects specified in the composition model (event, data and control flow, service access, etc.) – the *CRUISe Runtime*. During application initialization, it is specifically responsible for invoking the *Integration Service*, which provides UI components “as a service”, matching given application requirements and context information with available UIS listed in the *Component Registry*. Once the integration of UI components into an application is finished, the *Runtime* controls its execution. Additionally, it monitors context data and sends it to a context management service. There, it is processed, refined, and later provided to be used in the discovery and ranking process of UIS, as well as for the dynamic adaptation applications.

With our first prototype [9] we gained some useful knowledge on web-based UI integration and – as an outcome – decided to keep the place of integration conceptually flexible. Thus, the *Runtime* can reside both on the server or on the client-side, depending on the application requirements. For instance, service authentication in an enterprise setting necessitates a server-side part. Alternatively, the *Thin-Server Runtime* [29] allows for a completely browser-based composition and execution of consumer-oriented UI mashups, conforming to the SOFEA architectural style [25].

The next sections provide some more insight into the composition and integration process, as well as into dynamic adaptation mechanisms included with the *Runtime*.

B. Dynamic, Context-Aware Composition

The composition process outlined in the last section is illustrated by Figure 5. As can be seen, the provision of a service-oriented user interface for a mashup application is based on an integration work flow, which consists of three subsequent steps, namely (1) application generation, (2) UIS integration and (3) UIS binding. In this section we focus on steps 2 and 3.

The generation process results in a so-called *skeleton*, containing placeholders instead of actual component instances so that UI components can be “bound” at runtime just like back end services. Thus, at application startup, the *Runtime* is responsible for integrating UI components and for initializing them together with the other components included within the composition. Therefore, it sends a request containing application- and context-dependent requirements for each component to be integrated to the *Integration Service*. Subsequently, the latter starts an *Integration Task*, illustrated by Figure 6, which consists of different modules each responsible for a certain integration aspect. Its purpose is to find those UIS in the *Component Registry* that match given application requirements and context, to rank them by their accuracy of fit, and to return the platform-specific *binding* (cf. Section III-A) for the best match to the *Runtime*.

Since the *Integration Service* has an open service interface for requesting platform-specific components bindings, it can be used by different kinds of integration platforms – both client- and server-side. This even allows for the integration into independent solutions, e. g., into the JSP compilation process [9], into human tasks included in a business process [10], but also directly within the browser [29].

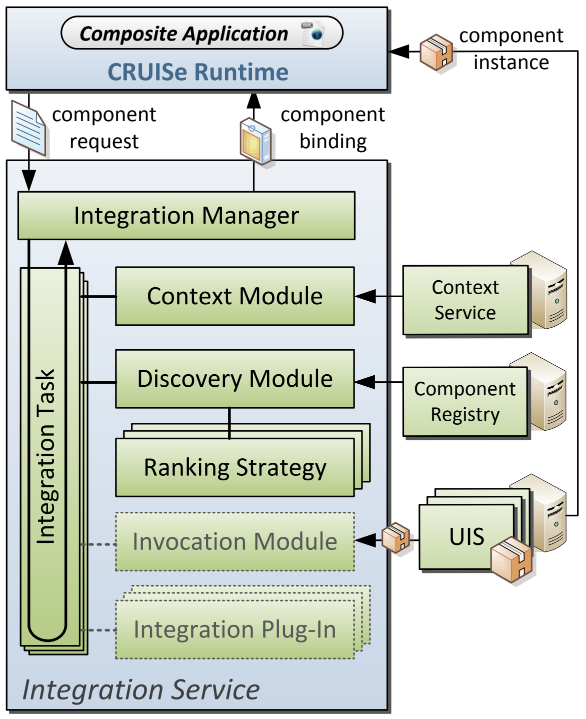


Figure 6. Internal structure of the *Integration Service*

The following modules are involved in the matching and integration process:

Integration Manager: This module handles all incoming component requests. Thus, it marks the remote entry point to the integration system and manages its internal data flow with the help of so-called *Integration Tasks*. For every request, a new task is started, which comprises a chain of actions (represented by individual modules) eventually returning a platform-specific component binding that best meets the given application requirements and context.

Context Module: Since the selection and configuration of UIS are based on contextual data, this module resolves references to context information being part of the transferred requirements into actual data, and evaluates context conditions. The quality and amount of this data heavily depends on the underlying context monitoring and modeling system. Therefore, we suggest the use of a sophisticated, service-oriented solution as described below.

Context Service: Modeling context places high demands on adaptive systems, including consistency checks, validation, and reasoning of information. In [30] we have presented a suitable, service-based solution called CroCo, which allows for the cross-application, ontology-based context management and reasoning, covering the above-mentioned requirements. Arbitrary context providers, such as the *Runtime* itself, other applications on the user's device, or hardware sensors, can send information to CroCo. Likewise, the *Context Module* and the *Runtime* request context data from the service, either synchronously or asynchronously by using a callback interface. More details on CroCo are discussed in [30].

Discovery Module: This module requests suitable component descriptions from the **Component Registry** – comparable to UDDI. Currently, we focus our work on the discovery and integration of UI components. Discovery within the registry is based on component *class*, e. g., "Map". In response, a result set of UISDL bindings is returned, which has to be ranked afterwards to determine the most adequate UIS. Therefore, the *Discovery Module* passes the set to a *Ranking Strategy*.

Ranking Strategy: In this step, the list of UIS in question is sorted with regard to predefined, possibly context-dependent criteria. Different ranking algorithms, or "strategies", may exist. They can be exchanged dynamically to support domain- or application-specific weightings of ranking criteria. Hence, the discovery process is divided into a class-based, functional matching carried out by the registry, and an application-specific, context-aware ranking performed by the strategy within the *Integration Service*.

Invocation Module: The necessity of a server-side invocation of UIS depends on whether there exists a suitable UISDL *binding* for the particular integration platform. As mentioned in Section III-A, it describes, how a component is integrated into a specific technology, i. e., how it can be initialized and how the interface signature maps to internal parameters and methods of the component. If, for example, a UI component should be integrated on the client-side into a JavaScript environment, a corresponding JavaScript-based component can be integrated directly on the client by loading the remote script in the browser. However, if technologies differ, UI components need to be wrapped in platform-specific code by the *Integration Plug-In* and are thus loaded from the UIS beforehand.

Integration Plug-In: In the ideal case, this module only extracts the necessary integration code from the UISDL *binding*. This includes initialization code, e. g., the constructor, as well as dependencies to other libraries. If no suitable binding exists, the component is wrapped with code, specific to the integration platform. Thus, for every runtime platform, there exists a corresponding *Integration Plug-In*. As an example, we have developed a plug-in, which integrates JavaScript-based components into Eclipse RAP[6] applications by automatically creating the corresponding server-side life cycle classes and by dynamically integrating them with the help of the *RAP Runtime*.

Once the *Integration Task* is finished, the *Integration Service* returns a platform-specific *binding* or wrapped component to the *Runtime*. It is interpreted or embedded into the composite application and executed, eventually. This can be referred to as *UIS binding* as shown in Figure 5. Of course,

[6]http://eclipse.org/rap/

this process involves a number of additional tasks, such as error handling and the provision of distinct namespaces to assure unique identifiers for each component included. However, those actions are not discussed in detail here.

After an application has been successfully initialized, the *Runtime* controls the event and data flow between its components as specified in the composition model. It also serves as a homogeneous access layer for various back end services. Furthermore, it monitors context data on the client, like user interactions and device capabilities, and sends them to the *Context Service*. In the end, it also carries out dynamic adaptations of the composite application, which is discussed in the following section.

C. Dynamic Adaptation of the Composite Application

As motivated in the beginning, situational awareness becomes increasingly important for web applications and poses additional challenges for web developers. Web applications need to adapt to different end users (characteristics, preferences, roles), devices (screen size, resolution, plug-ins) and situations (time, location, etc.). In CRUISe, context-awareness can be attained in different ways.

First and foremost, since UI components are selected and configured dynamically at run time, this process can be influenced by arbitrary context data, as discussed in the last section. For instance, the availability of necessary plug-ins on the client (e. g., Flash) can be taken into account when deciding which UI component to integrate.

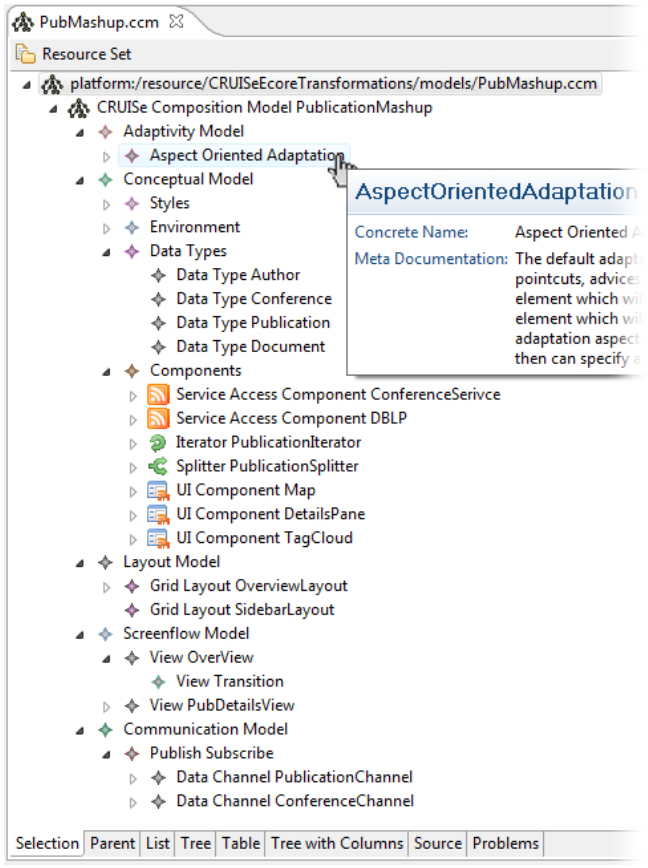
Second, context parameters may directly influence the configuration and state of a mashup component. This can be achieved by wiring contextual events from the context component (cf. Section III-B) with other components of an application. As an example, a location-aware map can be configured in such a way, that events from the context component providing the current geolocation trigger the map operation that updates its marker accordingly. Similarly, context parameters can be referenced within initialization events of an application, which results in a context-aware component configuration.

Finally, the *Runtime* contains an adaptation infrastructure, which dynamically evaluates context conditions specified in the *Adaptivity Model*, and carries out adaptations accordingly. They include component reconfiguration and exchange, adaptive layout and communication, as well as migration of components between client and server. Adaptation within the *Runtime* is defined with the help of rules, which define comporise context events, corresponding conditions, component references, and adaptation actions. Context data is requested or actively pushed from an external context service, e. g., CroCo. Details on adaptation rules, techniques, and context management are beyond the scope of this paper and will be published separately.

V. Implementation

To verify the concepts presented in the previous sections we implemented the composition model and the CRUISe infrastructure, and tested them with the help of different, exemplary composite applications.

The composition metamodel discussed in Section III-B was realized based on the meta-metamodel Ecore being part of the Eclipse Modeling Framework[7] (EMF). By using EMF, application models can be serialized in XMI and hence become exchangeable and tool-independent. Furthermore, an API can easily be created from the metamodel, simplifying the subsequent transformation step by using a number of already existing languages, such as QVT [31]. We also generated a tree editor, which integrates seamlessly with Eclipse (cf. Figure 7), and makes modeling composite applications rather easy with integrated validation support. Overall, Eclipse offers a powerful and flexible environment for future extensions, including the development of a corresponding visual editor.


```
PubMashup.ccm
Resource Set
platform:/resource/CRUISeEcoreTransformations/models/PubMashup.ccm
  CRUISe Composition Model PublicationMashup
    Adaptivity Model
      Aspect Oriented Adaptation
    Conceptual Model
      Styles
      Environment
      Data Types
        Data Type Author
        Data Type Conference
        Data Type Publication
        Data Type Document
      Components
        Service Access Component ConferenceSerivce
        Service Access Component DBLP
        Iterator PublicationIterator
        Splitter PublicationSplitter
        UI Component Map
        UI Component DetailsPane
        UI Component TagCloud
    Layout Model
      Grid Layout OverviewLayout
      Grid Layout SidebarLayout
    Screenflow Model
      View OverView
        View Transition
      View PubDetailsView
    Communication Model
      Publish Subscribe
        Data Channel PublicationChannel
        Data Channel ConferenceChannel

AspectOrientedAdaptation
Concrete Name:        Aspect Oriented A
Meta Documentation:   The default adapt
                      pointcuts, advice
                      element which wi
                      element which wi
                      adaptation aspect
                      then can specify a
```

Figure 7. Composition model tree editor

To validate the flexibility of the integration process, we put into practice different *Runtimes*, both server- and client-side. Our first prototype used a *Client-Server Runtime*, in

[7]http://eclipse.org/modeling/emf/

which UI integration takes place on the server as part of the JSP compilation process. The integration is HTML-based, utilizing the jMaki[8] widget framework, which manages component interaction on the client side. To assure uniqueness of class and object identifiers of the integrated components (with respect to the composite application), we use so-called *Universally Unique IDentifiers* [32], which are assigned during the integration process.

In [10] we presented the dynamic integration of UI components into a BPEL-environment based on the BPEL4People and WS-HumanTask (WSHT) standards. Therein, our *Runtime* is a Java-based extension of the ActiveBPEL[9] *Task List Client*, which lists and presents all human-involved tasks to users. Our extension includes two components: a *Parser* interprets the serialized composition model, which is embedded in the Human Task description, and *Bridge* component handles all the communication with the *Integration Service* and dynamically embeds UI components into the task UI.

As part of our latest work, we developed a *Thin-Server Runtime* (TSR) [29] which runs completely within the browser, following the SOFEA architectural style [25]. Therefore, we extended the JavaScript framework Ext[10]. The resulting architecture manages the components' dynamic integration, life cycles and communication, and provides a homogeneous service access layer which redirects external service access to a proxy provided by the *Integration Service*. This is needed to bypass the client-side "same origin policy" [33] which prevents access to services outside the original application domain. All *Runtimes* feature adequate error handling strategies for different faults, e. g., during component request, integration, and initialization. Typically, the action is repeated first, before discovery of alternative components is started. Additional run time security mechanisms are in the working.

The *Integration Service* is realized in Java according to the architecture illustrated in Figure 6 with additional functionality like local resource management and caching. It features both a lightweight REST and a SOAP interface. Functionality of the latter is largely based on Apache Axis2[11] and thus benefits from steady improvements and comprehensive standards support. The *Component Registry* builds on the WSMO framework [34] and internally models components uses the Web Service Modeling Language (WSML).

For demonstration and testing purposes, several prototypical, composite web applications were designed and built. To this end, a number of UIS were developed, encapsulating typical UI components, such as maps (Google Map, Yahoo Map), charts (Google Visualization API), an image browser, a feed viewer, a tag cloud, etc. Technologically, these components range from simple HTML and JavaScript (Google Maps, Dojo) to Flash (Flex) and Google GWT.

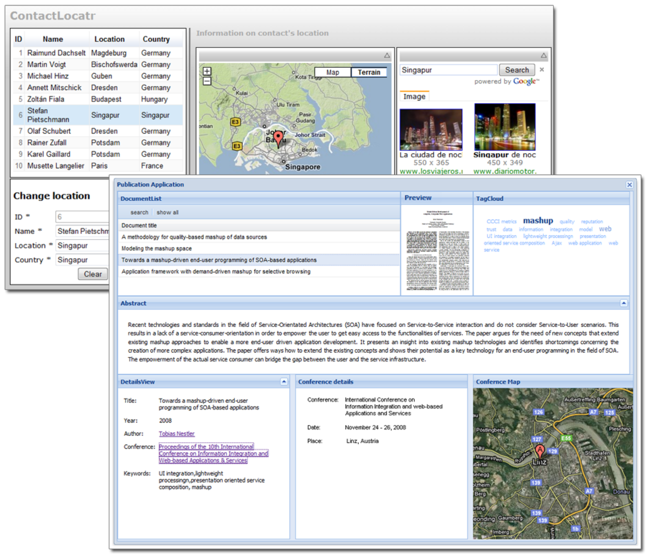
[8]https://ajax.dev.java.net/

[9]http://www.activebpel.org

[10]http://www.extjs.com/products/extcore/

[11]http://ws.apache.org/axis2/

Figure 8 shows two prototypes. The rear one is one of the first sample applications built upon our approach (cf. [9]). It allows for the management of contacts and provides additional information on their current location. Users may edit their data with the help of a form or by changing their location directly on a map. The application in the front lets users browse publications listed in a digital library. They can see details on the papers and check the corresponding conferences for related work. Different REST and SOAP services are utilized by this mashup, providing information on the authors and conferences which are visualized accordingly (e. g., keywords in a tag cloud, conferences on a map). The underlying composition model is partly shown in Figure 7.


ContactLocatr

ID	Name	Location	Country
1	Raimund Dachselt	Magdeburg	Germany
2	Martin Voigt	Bischofswerda	Germany
3	Michael Hinz	Guben	Germany
4	Annett Mitschick	Dresden	Germany
5	Zoltán Fiala	Budapest	Hungary
6	Stefan Pietschmann	Singapur	Singapur
7	Olaf Schubert	Dresden	Germany
8	Rainer Zufall	Potsdam	Germany
9	Karel Gaillard	Potsdam	Germany
10	Musette Langelier	Paris	France

Information on contact's location

Singapur

La ciudad de noc 550 x 365 www.losviajeros.
Singapur de noc 450 x 349 www.diariomotor.

Change location

ID * 6
Name * Stefan Pietschm
Location * Singapur
Country * Singapur

Publication Application

DocumentList

Document title
A methodology for quality-based mashup of data sources
Modeling the mashup space
Towards a mashup-driven end-user programming of SOA-based applications
Application framework with demand-driven mashup for selective browsing

Abstract

Recent technologies and standards in the field of Service-Orientated Architectures (SOA) have focused on Service-to-Service interaction and do not consider Service-to-User scenarios. This results in a lack of a service-consumer-orientation in order to empower the user to get easy access to the functionalities of services. The paper argues for the need of new concepts that extend existing mashup approaches to enable a more end-user driven application development. It presents an insight into existing mashup technologies and identifies shortcomings concerning the creation of more complex applications. The paper offers ways how to extend the existing concepts and shows their potential as a key technology for an end-user programming in the field of SOA. The empowerment of the actual service consumer can bridge the gap between the user and the service infrastructure.

DetailsView

Title: Towards a mashup-driven end-user programming of SOA-based applications
Year: 2008
Author: Tobias Nestler
Conference: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services
Keywords: UI integration,lightweight processingn,presentation oriented service composition, mashup

Conference details

Conference: International Conference on Information Integration and web-based Applications and Services
Date: November 24 - 26, 2008
Place: Linz, Austria

Figure 8. Two mashup applications integrating several UIS

With the help of these prototypes we could prove the feasibility of our model-driven development process and architecture for different application scenarios and platforms. They exemplify the dynamic integration of UI components into a mashup composition process, the event-based synchronization among components on all application layers, and the technology-independence of our approach.

Our modeling tools support the easy, platform-independent composition of applications from services and UIS at low authoring cost. At run time, the server-side integration and composition seems expedient especially for enterprise scenarios, because the quality and authenticity of back end and front end services can be ensured by the application server. As a downside, this negatively affects performance when the number of services, composite applications, and users grows. Thus, for a large part of use cases, a client-side composition and execution seems favorable, considering ongoing standardization efforts to resolve issues with the "same origin policy". Obviously, performance within a thin-server setting heavily depends on the browser used. In our tests, Google Chrome, Mozilla Firefox, and Opera offered a reasonable performance.

Of course, the proposed abstraction at the presentation layer implies a certain overhead at both design and run time. As application complexity can not be eluded, our concept basically shifts it from the application composer to the component developer. Reliable, elaborate component implementations and accurate descriptions are key factors for our concept to succeed. Given this, composition with our tools and models simplifies and shortens overall development while yielding platform-independence and context-awareness. Run time overhead of the dynamic component discovery and integration is less of a problem, since our infrastructure proves to scale rather well using caching and redundant services.

VI. Conclusion and Future Work

The development of composite web applications with rich user interfaces is a time- and money-consuming task, as current approaches lack a universal composition approach and are limited to integration at the data and application layers. Providing context-aware UIs for such applications poses additional challenges for developers. To address these issues, we have presented a model-driven development process for composite applications based on a universal, platform-independent composition metamodel, and a corresponding execution platform and infrastructure.

Our concept proposes a generic component model defining configurable, reusable parts of an application, and a novel, service-based deployment method for UI components. It implies the dynamic, context-aware composition of a mashup UI from so-called *User Interface Services* (UIS). In this context, a corresponding component description language (UISDL) has been developed. Composite applications are described with a platform-independent composition model, which defines all components used, their configuration, communication, and layout, as well as screen flow and adaptive behavior. On a higher level of abstraction, it specifies the coupling of UI services with back end services.

We have designed and tested the composition infrastructure CRUISe, which provides the necessary means for application composition and execution. This includes the homogeneous binding of back end services and the dynamic, context-aware selection, configuration, and integration of UIS. Moreover, the architecture supports dynamic adaptivity and adaptability of the composite application by means of component reconfiguration, exchange, adaptive layout, etc. To validate our approach, we built several prototypes illustrating the dynamic composition of context-aware mashup applications for different usage scenarios and platforms.

To our knowledge, CRUISe marks the first model-driven and fully service-oriented approach to a universal composition: It greatly simplifies platform-independent development, reuse and maintenance of composite web applications with context-aware UIs by deploying UI components "as a service" – comparable to service-oriented back end logic. Besides, it enables novel business models in the form of potentially commercial UIS, which may offer visualization and interaction at a higher level than standard interfaces.

Currently, we are working on more sophisticated, context-aware selection mechanisms being part of the *Component Registry* and the *Integration Service*. To this end, we are building a semantic classification of UIS to support semantic run time matching. In parallel, we are developing a sandboxing concept to improve security and privacy mechanisms within our *Runtime*s, so that a certain level of stability and information quality can be guaranteed, regardless of the services used by applications. Additionally, we are working on a visual composition tool to further simplify the development process.

Future work includes the extension of component descriptors and the *Adaptation Model* with regard to adaptation: Components' ability to self-adaptation and adaptability must be described to form the basis for defining higher-level *adaptation concerns* [35] ("device independence", "location-awareness", etc.) in the composition model. Moreover, we plan to extend our infrastructure to manage and dynamically include all types of components, including compositions themselves.

Acknowledgements

The CRUISe project is funded by the German Federal Ministry of Education and Research under promotional reference number 01IS08034-C.

References

[1] E. M. Maximilien, A. Ranabahu, and S. Tai, "Swashup: Situational Web Applications Mashups," in *Companion to the 22nd Conf. on Object-Oriented Programming Systems and Applications (OOPSLA'07)*. New York, NY, USA: ACM, 2007, pp. 797–798.

[2] "Gartner Identifies the Top 10 Strategic Technologies for 2008," Gartner Inc., Tech. Rep., October 2007, http://www.gartner.com/it/page.jsp?id=530109.

[3] O. Young, E. Daley, M. Gualtieri, H. Lo, and M. Ashour, "The Mashup Opportunity," Forrester, Tech. Rep., May 2008.

[4] C. Anderson, *The Long Tail: Why the Future of Business Is Selling Less of More*. New York: Hyperion, 2006.

[5] F. Daniel, F. Casati, B. Benatallah, and M.-C. Shan, "Hosted Universal Composition: Models, Languages and Infrastructure in mashArt," in *Proc. of the 28th Intl. Conf. on Conceptual Modeling*, November 2009.

[6] B. Benatallah and H. Nezhad, "Service Oriented Architecture: Overview and Directions," *Advances in Software Engineering: Lipari Summer School*, vol. 5316, pp. 116–130, 2007.

[7] P. Brusilovsky and N. Henze, “Open Corpus Adaptive Educational Hypermedia,” in *Adaptive Web: Methods and Strategies of Web Personalization*, vol. 4321, 2007, pp. 671–696.

[8] A. Kleshchev and V. Gribovy, “From an Ontology-Oriented Approach Conception to User Interface Development,” *Information Theories & Applications*, vol. 10, no. 1, pp. 87–94, 2003.

[9] S. Pietschmann, M. Voigt, and K. Meißner, “Dynamic Composition of Service-Oriented Web User Interfaces,” in *Proc. of the 4th Intl. Conf. on Internet and Web Applications and Services (ICIW 2009).* Mestre/Venice, Italy: IEEE CPS, May 2009, pp. 217–222.

[10] ——, “Adaptive Rich User Interfaces for Human Interaction in Business Processes,” in *Proc. of the 10th Intl. Conf. on Web Information Systems Engineering (WISE 2009)*, WISE. Springer LNCS, October 2009.

[11] F. Daniel, J. Yu, B. Benatallah, F. Casati, M. Matera, and R. Saint-Paul, “Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities,” *IEEE Internet Computing*, vol. 11, no. 3, pp. 59–66, May/June 2007.

[12] O. Diaz and J. J. Rodriguez, “Portlets as Web Components: An Introduction,” *Journal of Universal Computer Science*, vol. 10, no. 4, pp. 454–472, April 2004.

[13] W. Martin and R. Nußdorfer, “Role of Portals in a Service-Oriented Architecture (SOA),” S.A.R.L. Martin, CSA Consulting GmbH, Whitepaper, March 2006.

[14] M. Steger and C. Kappert, “User-facing SOA,” *Java Magazin*, pp. 65–77, March 2008.

[15] S. Abiteboul, O. Greenshpan, and T. Milo, “Modeling the Mashup Space,” in *Proc. of the 10th Intl. Workshop on Web Information and Data Management (WIDM).* Napa Valley, CA, USA: ACM, October 2008.

[16] D. Benslimane, S. Dustdar, and A. Sheth, “Service Mashups,” *IEEE Internet Computing*, vol. 12, no. 5, pp. 13–15, 2008.

[17] V. Hoyer and M. Fischer, “Market Overview of Enterprise Mashup Tools,” in *Proc. of the 6th Intl. Conf. on Service Oriented Computing (ICSOC)*, vol. 5364. Springer-Verlag, 2008, pp. 708–721.

[18] V. Hoyer, K. Stanoesvka-Slabeva, T. Janner, and C. Schroth, “Enterprise Mashups: Design Principles towards the Long Tail of User Needs,” in *Proc. of the Intl. Conf. on Services Computing*, vol. 2. IEEE, 2008, pp. 601–602.

[19] C. Pautasso, “Composing RESTful Services with JOpera,” in *Proc. of the 8th Intl. Conf. on Software Composition*, ser. LNCS, no. 5634. Springer, 2009, pp. 142–159.

[20] M. Caceres, “Widgets 1.0: Packaging and Configuration,” W3C Working Draft, April 2008. [Online]. Available: http://www.w3.org/TR/widgets/

[21] Google Inc., *Gadgets Specification*, http://code.google.com/apis/gadgets/docs/spec.html, Std.

[22] J. Yu, B. Benatallah, F. Casati, F. Daniel, M. Matera, and R. Saint-Paul, “Mixup: A Development and Runtime Environment for Integration at the Presentation Layer,” in *Proc. of the 7th Intl. Conf. on Web Engineering (ICWE’07)*, ser. LNCS 4607, Como, Italy, July 2007, pp. 479–484.

[23] J. Yu, B. Benatallah, R. Saint-Paul, F. Casati, F. Daniel, and M. Matera, “A Framework for Rapid Integration of Presentation Components,” in *WWW ’07: Proc. of the 16th Intl. Conf. on World Wide Web*, 2007, pp. 923–932.

[24] X. Liu, Y. Hui, W. Sun, and H. Liang, “Towards Service Composition Based on Mashup,” in *IEEE Congress on Services*, 2007, pp. 332–339.

[25] G. Prasad, R. Taneja, and V. Todankar, “Life above the Service Tier,” October 2007.

[26] T. Nestler, M. Feldmann, A. Preußner, and A. Schill, “Service Composition at the Presentation Layer using Web Service Annotations,” in *Proc. of the 1st Intl. Workshop on Lightweight Integration on the Web (ComposableWeb’09)*, June 2009.

[27] R. Acerbis, A. Bongio, M. Brambilla, S. Butti, S. Ceri, and P. Fraternali, “Web Applications Design and Development with WebML and WebRatio 5.0,” *Objects, Components, Models and Patterns*, pp. 392–411, 2008.

[28] J. Gómez, C. Cachero, and O. Pastor, “On Conceptual Modeling of Device-Independent Web Applications: Towards a Web-Engineering Approach,” *IEEE Multimedia*, vol. 8, no. 2, pp. 20–32, 2001.

[29] S. Pietschmann, J. Waltsgott, and K. Meißner, “A Thin-Server Runtime Platform for Composite Web Applications,” in *Proc. of the 5th Intl. Conf. on Internet and Web Applications and Services (ICIW 2010).* Barcelona, Spain: IEEE, May 2010.

[30] S. Pietschmann, A. Mitschick, R. Winkler, and K. Meißner, “CroCo: Ontology-Based, Cross-Application Context Management,” in *Proc. of the 3rd Intl. Workshop on Semantic Media Adaptation and Personalization*, December 2008.

[31] *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification*, Object Management Group Std., April 2008. [Online]. Available: http://www.omg.org/spec/QVT/1.0/

[32] *RFC4122: A Universally Unique IDentifier (UUID) URN Namespace*, Internet Engineering Task Force (IETF) Std., July 2005. [Online]. Available: http://tools.ietf.org/html/rfc4122/

[33] C. Jackson, A. Bortz, D. Boneh, and J. C. Mitchell, “Protecting Browser State from Web Privacy Attacks,” in *Proc. of the 15th Intl. Conf. on World Wide Web (WWW 2006).* Edinburgh, UK: ACM, 2006, pp. 737–744.

[34] R. Herzog, H. Lausen, D. Roman, M. Stollberg, and P. Zugmann, *WSMO Registry*, WSMO Working Draft, Std., Rev. 0.1, April 2004, http://www.wsmo.org/2004/d10/v0.1/.

[35] M. Niederhausen, K. van der Sluijs, J. Hidders, E. Leonardi, G.-J. Houben, and K. Meißner, “Harnessing the Power of Semantics-based, Aspect-Oriented Adaptation for AMACONT,” in *Proc. of the 9th Intl. Conf. on Web Engineering (ICWE’09)*, ser. Edition 5648, M. Gaedke, M. Grossniklaus, and O. Díaz, Eds., San Sebastian, Spain, Juni 2009.

A Policy-based Dependability Management Framework for Critical Services

Manuel Gil Pérez, Jorge Bernal Bernabé, Juan M. Marín Pérez, Daniel J. Martínez Manzano,
and Antonio F. Gómez Skarmeta
Departamento de Ingeniería de la Información y las Comunicaciones
University of Murcia, Spain
Email: {mgilperez,jorgebernal,juanmanuel,dmartinez,skarmeta}@um.es

***Abstract*—Many critical activities rely on the correct and uninterrupted operation of networked Computer Information Systems (CIS). Such systems are however exposed to many different kinds of risk, and thus many researches have been taking place for enabling them to perform self-monitoring and self-healing, and so maintaining their operation over time as specified by domain policies. These capabilities are the basis of what is commonly referred to as *dependability*. The DESEREC project has defined a tiered architecture as a policy based framework to increase the dependability of existing and new networked CIS, using technology-independent information which is translated at runtime to suit the managed components. This paper delves into how DESEREC builds and manages large critical systems through an agent-based distributed framework, and how it is able to respond to any adversity effectively, such as intrinsic failures, misbehavior or malicious internal use, and attacks from the outside. An illustrative example is used throughout this paper to demonstrate all the concepts and definitions presented.**

***Keywords*-Dependability; Policy Based Management; Self Healing; Configuration Constraints; Dynamic Reconfiguration**

I. Introduction

This article is an extended and revised version of the conference paper entitled "Towards a Policy-driven Framework for Managing Service Dependability" [1]. It contains a more comprehensive and detailed explanation of the proposed framework, and a complete running example with the aim of demonstrating the concepts and models herein introduced.

As networked Communication and Information Systems (CIS) become more pervasive, even more critical activities rely on their correct and uninterrupted operation. Such systems are however exposed to many different kinds of risk, including hardware failures, software bugs, connection and power outages, and even malicious use. In this context, much research has been taking place with the aim of providing networked CIS with a new capability, beyond the classic concept of fault tolerance and data redundancy [2]. This new feature will enable administrators to perform self-monitoring and self-healing to maintain the system operation over time, as specified by domain policies. This capability is often referred to as *dependability* in literature [3][4].

Dependability covers many properties relevant to the self-management of critical systems [5][6]. Among them, we can emphasize:

- Availability: probability that a service is available for use at any time; it allows for service failure, with the presumption that service rehabilitation is immediate.
- Reliability: measure of the continuous delivery of a service in the absence of failure.
- Safety: non-occurrence of catastrophic consequences or abuse to the environment or its users.
- Survivability: ability of a system to provide crucial services in front of attacks and failures, and restore those services in the least amount of time.
- Maintainability: capability of a system to support changes and evolutions, possibly under hostile conditions.
- Security: it includes some properties to maintain truthfulness and confidence in the managed data. These properties are mainly used for confidentiality, integrity and non-repudiability purposes.

Furthermore, it is important to note that security properties are generally limited to discrete values, e.g., a user is authenticated or not, the information is either available or it is not, etc., whereas, on the contrary, dependability properties are continuous or multi-valued, expressed in terms of probabilistic measures, e.g., a system is highly reliable beyond 95%.

Bearing in mind the above properties, a dependable framework with these goals should be pursued by following a multidisciplinary approach, in the search of a tiered architecture with the following responsibilities:

- Build abstract models of the managed system, the services to be provided by it, and any other needed management information (policies).
- Set up the technical components automatically in the system so that the intended services are provided.
- Perform intensive and extensive monitoring all over the managed system.
- Take appropriate actions when something wrong is detected.
- Quick containment as close as possible to the managed system.

At the sight of the above list, one can easily work out the value of a policy based management framework in such an architecture. Policy based management (PBM) [7][8] is a management paradigm by which rule-like pieces of information are used to describe the desired operation and behavior of a system; these rules are typically stored in a repository from which they are selectively distributed across the concerned enforcement entities. These entities will then make use of their local knowledge in order to guarantee the correct operation of the areas under their management. Since these rules are described in an abstract fashion, enforcement entities may require to perform some technology-dependent translation on them so that they suit the managed components.

Having the above objectives as a goal, the DESEREC project [9] has defined a multi-tiered architecture as a framework to increase the dependability of existing and new networked CIS, by means of the following functional blocks:

- Modeling and simulation. DESEREC devises and develops innovative approaches and tools to design, model, simulate, and plan critical infrastructures to improve their resilience.
- Incident detection and quick containment. DESEREC integrates various detection mechanisms to ensure fast detection of severe incidents, and thereby avoiding any impact propagation.
- Fast reconfiguration with priority to critical activities. DESEREC provides a framework to respond in a quick and appropriate way to a large range of incidents to mitigate the threats to the dependability and thwarts the problem.

Next sections will give a deeper insight on some aspects of the overall solution, as follows: first, Section II introduces the main related work as background information for the reader; then, Section III explains the designed framework, describing its tiered architecture; Section IV presents a running example that will be used by the following sections to demonstrate the explained concepts; Section V describes the abstractions used by the DESEREC framework for binding configurations to services; Section VI delves into the implementation of the policy-based models and engine; the complete reconfiguration framework is explained in detail in Section VII; Section VIII summarizes our experience in the design of a dependable system following this approach; and lastly, some conclusions are drawn in Section IX.

II. Related Work

The modeling of large systems with the aim of reconfiguring automatically the services they offer has been the cornerstone for the last few years, precisely due to the great complexity that these systems convey.

There currently exist a great amount of tools that provide high availability of critical services in case of physical failures, whether due to hardware failures or even natural disasters.

Among them, the most commonly used are: *backup systems* which facilitate an automation of the backup process and a quick restoration of the data; and *clustering of servers* (a “battery” of servers that are monitored each other to detect temporary failures or system crashes). In both cases, these tools are able to restructure dynamically to continue offering the service. However, as main drawback, they are unable to detect attacks from malicious users (whether internal or external).

Many of the existing tools with a similar purpose are mainly focused on a single kind of services that the system must handle, like Web-based [10] or computer-based approaches [11]. New efforts are being made to provide a compact solution that includes the complete life cycle of any large system without human intervention; from the configuration, the monitoring until the reconfiguration. The great challenge behind this is to achieve truly 24x7 systems, or continuous availability.

In [12], the authors presented a framework based on the monitoring, performance evaluations and dynamic reconfiguration of the SIENA network. On the other hand, in [13] the authors extended this management to mobile environments following a similar approach to [12], by monitoring and estimating the redeployment architecture to maintain the availability of this kind of systems.

As a solution to these emerging issues, two initiatives have stood out. On one hand, the SERENITY EU-funded R&D project [4] aims at supplying security solutions and high availability in Ambient Intelligence (AmI) systems and services. These systems are mainly concerned in human and services interactions, especially in mobile distributed environments. In this context, in [14] the authors delve into the SERENITY approach, providing security and dependability (S&D) solutions for dynamic, highly distributed and heterogeneous systems.

On the other hand, the Willow approach [15] focuses on the design of an architecture to provide a great resilience to failures in large distributed information systems. This architecture offers mechanisms based on specifications for fault-tolerance techniques, but it sets aside many fundamental issues related to the dependability, like misconfiguration, misbehavior or malicious use. Such an approach has proven its value when it has been successfully applied to Grid management [16], enabling the dynamic reconfiguration of a grid without human intervention in response to environment changes. Additionally, this research group is currently focusing on the applicability of the Willow approach to the novel computing paradigm of Cloud Computing [17].

All these initiatives offer some initial results for the critical systems management with high availability, but they are still far from providing an integral system that allows managing large systems in a completely autonomous way, as standardized as possible, trying to save costs in managing the inactivity of a service, and so on.

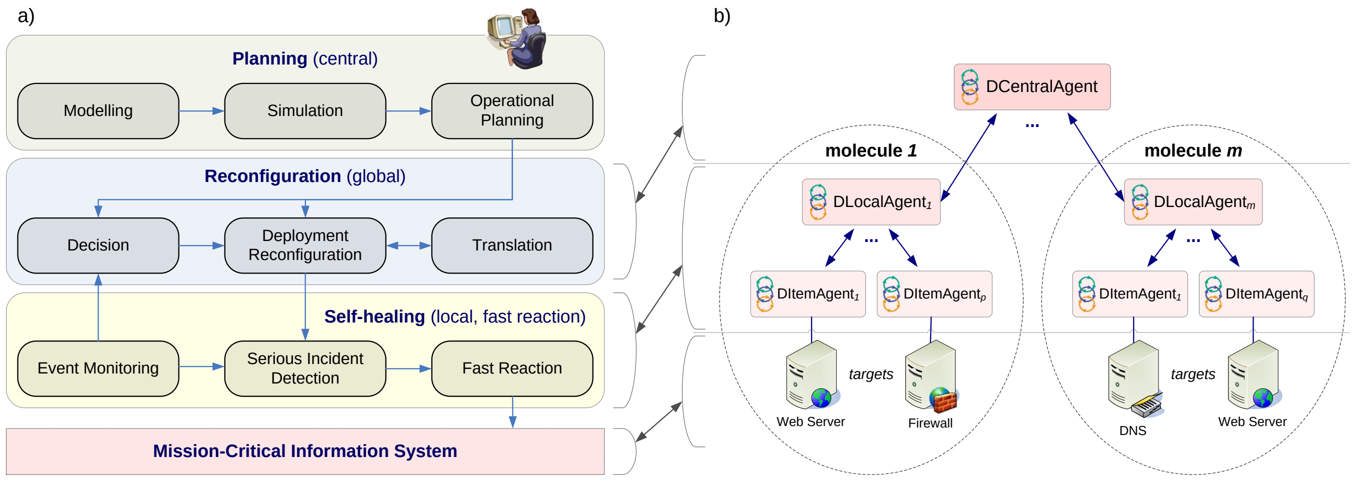


Figure 1. Model-based approach: a) high level functional blocks; and b) the DESEREC framework

III. DESEREC: A Framework to Enhance Dependability

The DESEREC project [9] has defined a 3-tier architecture to increase dependability of any CIS by means of three different and connected approaches: modeling and simulation; detection; and response. Its main goal is to react appropriately upon incidents of any nature, e.g., errors, failures, malicious actions, to maintain critical services always available. The proposed approach supports a mid-term strategy with planning and simulation tools for modeling in a proactive way the performance and dependability of any CIS. Figure 1 depicts how DESEREC manages these critical systems using a model-based approach, which is organized around a 3-tier reaction loop.

As can be seen in Figure 1a), the three objectives proposed by DESEREC can be identified clearly: firstly, plan and model the operations of the system, as well as configuring it properly by the system manager at design time; as second objective, detection and prevention of incidents and potential faults proactively, from the events harvested on the target system; and finally, react to the detected incidents by either reallocating the services or executing a set of abstract commands to fix the problems caused. Both of them (services configuration and abstract commands) are defined in a generic way with the aim of abstracting system managers from technology-dependent details.

A. Three-Tier Agent-based Reaction Loop

The DESEREC runtime architecture has been developed following a multi-layered approach in order to manage large systems by means of splitting the underlying CIS in different areas of autonomic management [18]. As can be seen in Figure 1b), a *molecule* is the minimal sub-division of any CIS with the aim of grouping physical components (servers and network equipments) under the same management control at local level [19]. This division can host one or more technical services such as authentication, IP allocation, etc. Each molecule accommodates one local molecule agent, called *DLocalAgent*, which should:

1) Monitor low level events reported by the managed elements, or directly harvested from them.
2) React locally to serious incidents with enforcement capabilities (usually through secure channels).
3) Apply high-level reconfiguration orders, coming from a central agent, to enforce a new operational plan.

Each DLocalAgent in turn manages one or more item agents, called *DItemAgent*, which are in charge of handling the target infrastructure elements. Each DItemAgent is responsible for monitoring those underlying elements and to enforce available reactions in case of a system failure. Note that in this architecture only this last kind of agent has knowledge about the final technical service implemented (software and version, how should be started/stopped, etc.); that is, they manage vendor-specific information, whereas the rest of agents use technology-independent information.

Finally, a single central agent is placed to have a global view of the whole infrastructure. This agent is called *DCentralAgent* and will receive local events and alarms from DLocalAgents to detect incidents which could have passed as undetected by these latter; for example, by correlating events from different molecules to detect a distributed attack. The DCentralAgent is also able to take decisions based on the above information and thereby launching a global reconfiguration process, which could imply to more than one molecule.

As seen, apart from splitting the CIS in different areas, the main goal is to manage the underlying system as close as possible to the lower layers (target infrastructure). Thus, the detection, decision and reaction logic will be much faster and it will avoid overloading the higher level entities.

B. Planning Block

Initially, the administrator defines both a formal description of the networked system and the Business Services that the system should offer. On the one hand, a high level language has been defined to obtain a suitable description of the network environment, called *System Description Language* (SDL) [20][21]. SDL permits to specify the system design in great detail about its physical and logical infrastructure: network elements such as interfaces, gateways and links; technical services, e.g., software and version; etc. On the other hand, the services information is modeled using the W3C's *Web Services Choreography Description Language* (WS-CDL) [22]. With this language, the administrator is able to describe the services of the system and the relationships between them by way of choreography. This allows defining the sequence and conditions in the information exchange between the participants.

The next step for the administrator is to provide the requirements or constraints that define the behavior of the Business Services. From these constraints, defined at high level, our framework is able to generate automatically the configuration model in a software-independent way. These configurations, plus a set of rules, are what we denominate *Operational Plan* (OP) [18].

An OP is an allocation strategy that defines how the services should be mapped onto molecules or technical services, and how this allocation should change when undesired incidents happen.

It comprises the following items:

- One or more *Operational Configurations* (OC). Each of them describes a particular allocation of the offered services onto either molecules (global) or technical services (local), depending on the level where the OC will be applied, called *High-level Operation Configurations* (HOC) and *Low-level Operation Configurations* (LOC) respectively. For each of these allocations, an OC includes one or more high-level configuration policies for setting up the final service.
- *Detection Scenario*. It describes which foreseen incidents we are interested in; for example, the free disk space is reaching a critical point (above 95%) or a certain command is executed by an unauthorized user. This Detection Scenario is divided into two different abstraction levels, called *Global Detection Scenario* (GDS) and *Local Detection Scenario* (LDS).
- *Reaction Scenario*. It specifies how to react when each one of the above incidents happens. This reaction consists of either switching between Operational Configurations or executing a set of abstract commands in order to fix the problems caused by the incident detected. As before, this Reaction Scenario is divided into two different ones, called *Global Reaction Scenario* (GRS) and *Local Reaction Scenario* (LRS).

Thus, an OP can be seen as a graph of Operational Configurations in which nodes are individual OCs whereas links are allocation and configuration changes launched by the detection of known incidents. Due to the multi-layered strategy employed by this architecture, there exist two kinds of OPs, called *High-level Operational Plan* (HOP) and *Local-level Operational Plan* (LOP), which can be better seen as:

$$HOP = \{\{HOC_1, HOC_2, ..., HOC_l\}, GDS, GRS\}$$

$$HOC_i = \{\{LOP_{i1}, LOP_{i2}, ..., LOP_{im}\}\} \ \forall i \in [1..l]$$

$$LOP_{ij} = \{\{LOC_{ij1}, ..., LOC_{ijn}\}, LDS_{ij}, LRS_{ij}\} \ \forall j \in [1..m]$$

Where m is the number of molecules in the system, whereas l and n can vary according to the possible Operational Configurations defined for each given plan.

A HOP -there will normally be only one per managed system- contains a list of possible HOCs (high-level allocation of the services onto molecules) and two scenarios for detection and reaction purposes, GDS and GRS respectively, which specify how to switch between HOCs when a problem arises. Each of these HOCs carries a set of LOPs (one per molecule) which represent a low-level allocation and configuration plan. Each LOP contains in turn a list of possible LOCs and two refined scenarios for detection and reaction purposes at local level (LDS and LRS, respectively). These LOCs represent services that are mapped onto the system components, all belong to a particular molecule, whereas the scenarios express how to switch between LOCs after detecting a fault in the system. Note that each of these local allocations will only affect to the molecule where the problem arises without involving others.

C. Reconfiguration Framework

The DESEREC modeling framework is completed with a detection and reaction model, which describes when and how to reconfigure the system in response to an incident. The reaction stage is carried out once the DESEREC framework detects that a serious incident has occurred and, in consequence, the system should react by either switching from the current OC to another or executing a set of abstract commands. The former is the imposition of a new allocation for the system services, with (possibly) a new configuration for them, whereas the latter is a minor change in the target system but without changing the running operational plan; for instance, by executing a *ddns* command to add a new RR to the domain name server dynamically.

The detection engine constantly receives events from the lower down layers which are mapped against the current Detection Scenario. When a coincidence is detected an alarm is fired, thereby starting the reaction process. Since different reaction rules (defined in the Reaction Scenario) can be associated to the same Detection Scenario rule, the decision engine should determinate which of them is the most suitable one; that is, which is the best OC or set of abstract commands to be enforced in the target system to fix the detected problem.

This reaction process is driven by a *Policy-Based Network Management* (PBNM) approach [23], which is responsible for deploying, installing and enforcing policies in target devices; in our case either an *Operational Configuration* (OC) or a set of abstract commands, depending on the kind of reaction chosen by the decision engine, and defined in the Reaction Scenario.

An in-depth explanation of the DESEREC framework and its modules is provided in Section VII.

IV. Illustrative Example

This section introduces a complete running example, which represents usual problematic situations with the aim of clearly demonstrating the concepts and definitions that will be explained in the following sections. These situations capture typical dependability and security problems which can be managed and solved through the DESEREC framework.

A. Services

The physical testbed simulates a railway signaling and control system, where the network and services infrastructure is depicted in Figure 2. It is composed of a private network which is connected to the corporative Intranet through a firewall component. This network makes services accessible for railway administrators.

Let us suppose the following services:

- Railway Web service: it provides a Web service interface for the railway signaling and control system.
- DNS service: it defines a domain name for the IP address of the previous Web service.
- Firewall service: it keeps packet filtering rules to services and network components.
- Timing service: it provides time synchronization for all the components that need it.

The railway Web service is configured to listen on two ports, depending on the requested services: on port 5486 a Web service interface is offered for interaction with the signaling and control system; and on port 443 administrators can gain access to a management Web page which provides statistics and monitoring services for the system. Both interfaces are offered through a secure HTTPS communication by using, for example, SSL or TLS with X.509 certificates.

The DNS service will be configured with the IP address where the Web service will be placed and, at the beginning, the firewall will allow connections to both service ports. In order for the service to be accessed through its domain name by administrators, the firewall will also allow connections on port 53.

B. Testbed Description

The scenario used for this example is comprised by two molecules. Figure 2 depicts the molecules and the software distribution in the testbed.

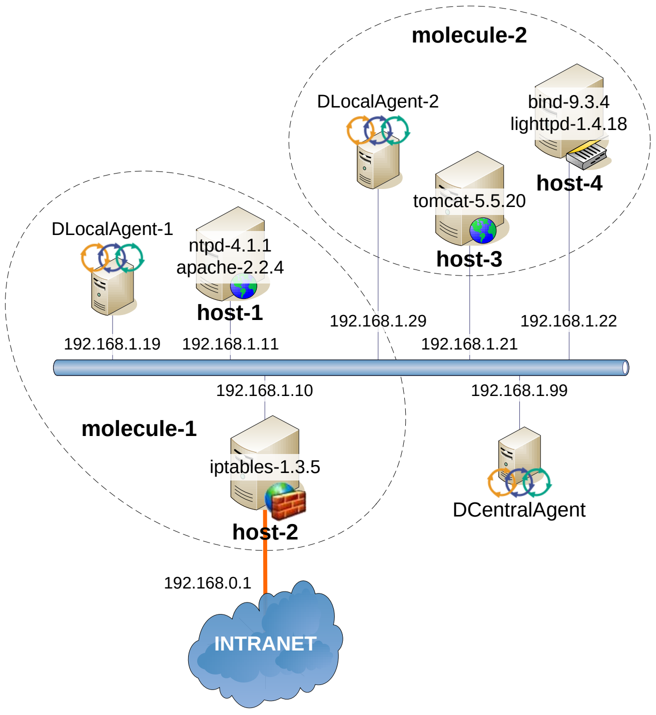


Figure 2. Molecules distribution and DESEREC components in the testbed

Table I summarizes the software requirements, along with their version numbers, needed to properly deploy this scenario.

Table I
Elements into the molecule breakdown

MOLECULE	COMPONENT	SPECIFIC SOFTWARE
molecule-1	Web server	apache-2.2.4
	Firewall	iptables-1.3.5
	Timing server	ntpd-4.1.1
molecule-2	Web server	lighttpd-1.4.18
	Web server	tomcat-5.5.20
	Name server	bind-9.3.4

It is worth noting that there are three Web servers available, installed in different elements. Two of them (Apache and Tomcat) support secure connections through SSL and TLS, whereas the other one (LigHTTPd) is not able to provide this feature. This last server has been included in the testbed, although it will be discarded during the allocation process since it does not provide the required features.

In this scenario, the corresponding DESEREC framework entities have also been included: one DLocalAgent per molecule, which will receive event notifications and will launch the detection and reaction processes; and a DCentralAgent in charge of supervising those molecules.

Three different situations are considered to show how the DESEREC framework works to confront dependability and security problems. Either of these situations could cause dependability or security problems that the DESEREC framework should correct. They will make the framework react in three different ways and will serve to illustrate the following three kinds of reactions:

1) *Reallocation*: The Web service goes down due to a DoS attack from a compromised host in the Intranet and it becomes unavailable. In this case, the framework reaction is to move (reallocate) that service to another place which is able to get it running again.
2) *Reconfiguration*: An unauthorized user gains access to the private Web page. The reaction of the framework in this situation is to reconfigure the firewall by changing its current configuration to another more restrictive. Only connections to the critical system through HTTPS will be permitted on port 5486, blocking accesses to the Web page.
3) *Abstract command execution*: Some important Web service files are removed from the server due to a temporal hard disk failure and the Web service becomes inconsistent. The reaction of the framework is to execute a command, specified using an abstract syntax, that will copy the removed files from a backup folder to the appropriate directory in the Web server.

V. High-level Configuration Constraints

The deployment of Operational Configurations, used to enhance the system dependability, requires the definition of the desired prerequisites that Business Services should accomplish. These prerequisites are comprised by a set of requirements or constraints defined during the business process and specified by the administrator using a set of high level configuration policies for the different Business Services.

Each Business Service in a DESEREC-managed system is comprised by a number of individual *Business Service Components* (BSC). These BSCs carry out specific tasks of their Business Service, which can be mapped to one or more technical services of some kind. For example, a Business Service that provides a Web portal could be comprised by one BSC for delivering the Web content, another BSC for data storage, and another for performing access control operations.

Typically, any technical service needs to have a proper configuration in order to implement a BSC. The system administrator must supply some configuration policies or constraints (ideally, vendor-independent ones) about how the BSCs should operate. These policies will serve as the bases for generating the configuration of the technical services that will finally implement the BSCs. These guidelines are referred to as *service configuration constraints*.

The service configuration constraints are modeled in DESEREC via the *Service Constraints Language* (SCL) [20]. This language specifies which are the service configuration constraints for a given DESEREC-managed system, and how the constraints defined in it are related to the BSCs of the system.

Figure 3 gives a basic view of the Business Services and their BSCs defined for the illustrative example introduced in Section IV, and how each of them has a set of one or more configuration constraints.

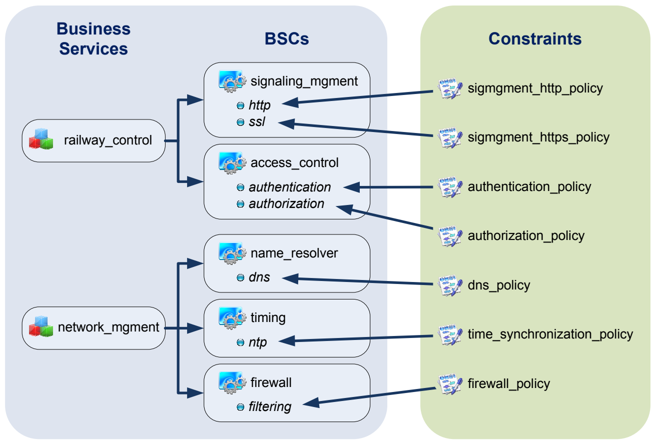


Figure 3. Constraints assigned to the BSCs

The relationship between BSCs and constraints lies on the concept of *capability*. A BSC is associated with a set of capabilities which define the type of service the BSC provides. For instance, the BSC providing access control has two capabilities: *authentication* and *authorization*.

The set of capabilities associated to one BSC determines the possible service configuration constraints that can be assigned to it. Every configuration policy defined in SCL belongs to a constraint type, being different constraint types able to configure specific capabilities. Thus, the BSC *access_control* has two constraints assigned to it: an *AuthenticationConstraint* specifying the authentication policy for the *authentication* capability; and an *AuthorizationConstraint* specifying the authorization policy for the *authorization* capability.

It is worth noting that configuration constraints may be reused. If more than one BSC requires exactly the same behavior for a given type of service, then they may share the same constraint. For example, there may be a new Business Service in the system containing another BSC for time synchronization which should have the same behavior and synchronize with the same time servers, as the one shown in Figure 3. That BSC may share the already defined *time_synchronization_policy* constraint, that is, the administrator can assign the same constraint to both BSCs.

This however does not mean that exactly the same technical configuration will be used on both, because they could be implemented by different software packages. Constraints are vendor-independent and they will have to be translated to final configurations only when the final software to which they will be applied is known.

Since configuration constraints are defined before the final software implementing the services is known, SCL is able to describe the configuration semantics in a vendor independent way. Moreover, dependability management implies service reallocation; e.g., a service that is running on a machine can be reallocated to another if the previous one fails. The first situation in the illustrative example of Section IV introduces this kind of reaction for the railway Web service, and it will be further described in Section VII-E (Situation 1). This requires the description of configuration constraints with a high level of abstraction, based on the BSC concept and avoiding the usage of allocation-dependent data.

The definition of domain name resolution constraints shows this problem about service reallocations. In the example, the managed system provides a website which should be accessed from the Intranet through a given URL, for instance `https://signaling.example.org`. The domain `example.org` is managed by the company and, therefore, the DNS service should be configured to map `signaling` to the company's railway signaling website. In this scenario, there are two different BSCs, one representing the website and another one representing the DNS service. The configuration constraint defining the behavior for the BSC *name_resolver* should specify a record with the IP address of the website as response for the `signaling` query by the DNS service. However, the IP address of the website is unknown since the service can be reallocated to a different machine in case of failure.

Because of this, SCL supports BSC references in policies. This allows administrators to specify BSC identifiers instead of IP addresses, or any other static information, which may vary depending on where the service is finally allocated by the dependability management system.

Other examples to illustrate these high level configurations may be the time synchronization policy specifying *Listen to the default NTP port* and *synchronize with another timing BSC named "corporative_timing"*. Or the filtering policy specifying *Allow traffic from the Intranet to the BSC "signaling_mgment"* and *Deny everything else*.

A further policy refinement process will resolve these references and will translate the high level configuration constraints defined in SCL into a lower level language. This language will contain all the specific and detailed data needed to generate the precise configuration for the current system.

To show the SCL structure and some of the aforementioned characteristics, Listing 1 presents the corresponding constraints for the illustrative example in Section IV.

```
<scl id="example_policies" plan="example_plan">
  <constraintAssignments>
    <bscConstraints bsc="example.access_control">
      <constraint ref="authentication_policy"
         requiredCapability="authentication"/>
      <constraint ref="authorization_policy"
         requiredCapability="authorization"/>
    </bscConstraints>
    ...
  </constraintAssignments>
  <constraintDefinitions>
    <authenticationConstraint id="authentication_policy">
      <authenticationRules>
        <authenticationRule name="user1_id">
          <identity>user1@example.org</identity>
          <credentials>
            <accountCredential>
              <accountId>user1</accountId>
              <accountContext>example.access_control</accountContext>
            </accountCredential>
          </credentials>
        </authenticationRule>
        ...
      </authenticationRules>
    </authenticationConstraint>
    <authorizationConstraint id="authorization_policy">
      <roles>
        <role name="signaling_admins">
          <identity>user1@example.org</identity>
          ...
        </role>
      </roles>
      <authorizationRules>
        <authorizationRule name="allow_signaling_mgment">
          <role>signaling_admins</role>
            <privileges>
              <privilege granted="true" name="signaling_mgment_access">
                <activities>
                  <activity>Read</activity>
                  <activity>Write</activity>
                </activities>
                <qualifiers>
                  <qualifier type="Packets"/>
                </qualifiers>
                <target>signaling.example.org</target>
              </privilege>
            </privileges>
        </authorizationRule>
      </authorizationRules>
    </authorizationConstraint>
    <httpConstraint id="sigmgment_http_policy"> ... </httpConstraint>
    <sslConstraint id="sigmgment_https_policy"> ... </sslConstraint>
    <dnsConstraint id="publicdns_policy">
      ...
      <zones>
        <zone domain="example.org" type="Master">
          ...
          <records>
            ...
            <record type="A">
              <query>signaling</query>
              <response type="BSC">example.signaling_mgment</response>
            </record>
          </records>
        </zone>
      </zones>
    </dnsConstraint>
    <filteringConstraint id="firewall_policy"> ... </filteringConstraint>
    <ntpConstraint id="time_synchronization_policy"> ... </ntpConstraint>
  </constraintDefinitions>
</scl>
```

Listing 1. Service Constraints Language

For clarity reasons, XML namespaces have been removed from the listing and some fragments have also been replaced by dots. It can be seen that this set of constraints and policies contains an identifier (*example_policies*) and it is defined in the scope of a concrete Operational Plan, named *example_plan* in this case.

An SCL description is formed by two main parts: constraint assignments and constraint definitions. The former defines the policy which is associated to each capability of each BSC defined in the operational plan. The latter contains the actual definitions of the policies. This separation allows reusing policies defined by different BSCs; i.e., a policy defined in the definitions section can be assigned to two different BSCs in the assignments section.

Listing 1 shows just one constraint assignment corresponding to the BSC *access_control*. In the full description there is a constraint assignment for every BSC. Moreover, all policies appear in the description section, but their content has been replaced by dots and only the two assigned to the BSC *access_control* appear almost complete, just to illustrate the aspect of policies in SCL. Moreover, some fragments of the DNS policy of the BSC *name_resolver* are also shown to illustrate the previous example of BSC references in constraints. It can be seen how the record for `signaling` within the domain `example.org` assigns that domain name to the BSC *signaling_mgment* without still knowing its actual IP.

VI. Policy Modeling in DESEREC

As stated in Section V, within the DESEREC framework the administrator defines the behavior of the Business Services by means of a set of configuration constraints or policies defined in SCL. This language operates in a high level fashion, making use of platform-independent semantics as well as being transparent to the allocation process from Business Services to technical services.

In this context, it is worth reaching a trade-off between a high level formal language able to express the administrator's abstract requirements, in a human readable way, and a formal language able to be parseable by an automatic translation process. This is exactly what the SCL language deals with, defining models as clear as possible to be later used by an intelligent software process.

However, a standard model with the proper level of detail to allow generating configurations for the real system is also needed in order for the enforcement phase to be done properly in a vendor independent way. DESEREC uses the *Common Information Model* (CIM) [24] as this final model since it is a very complete information model. It covers almost all the different aspects required in a networking scenario, including systems, services, networks, applications, policies, security, etc. Moreover, CIM is independent of the language used to represent it, free, open source and extensible. Additionally, this information model has been used in a wide variety of research works, such as [25] or [26] among others.

Both DMTF and DESEREC define an XML representation of this model following two different approaches. The DMTF, in its standard CIM-XML of WBEM, uses a metaschema mapping which defines an XML schema to describe CIM, where both classes and instances are valid documents to the CIM metaschema. On the other hand, DESEREC uses an XML schema to describe CIM classes as XML complex types. Thus, CIM instances are described in valid XML documents for that schema. DESEREC reuses and extends the xCIM language, originally defined in the POSITIF EU-IST project (*Policy-based Security Tools and Framework*, IST-2002-002314), to provide this last kind of XML format representation of CIM. Furthermore, and thanks to the usage of the xCIM language, a wide range of possibilities become available. For example, there are related standards and technologies grouped under the WBEM specifications [27], which allow dynamically gathering the current state of the system by means of CIM.

Therefore, although the xCIM language is a useful implementation of CIM (including extended classes), it does not fit perfectly in the DESEREC requirements due to the great amount of classes that compound the model, and the lack of some DESEREC-specific requirements. To solve this issue, DESEREC defines a sublanguage, called *xCIM Service Constraints Language* (xCIM-SCL), that allows the representation of service constraints and policies.

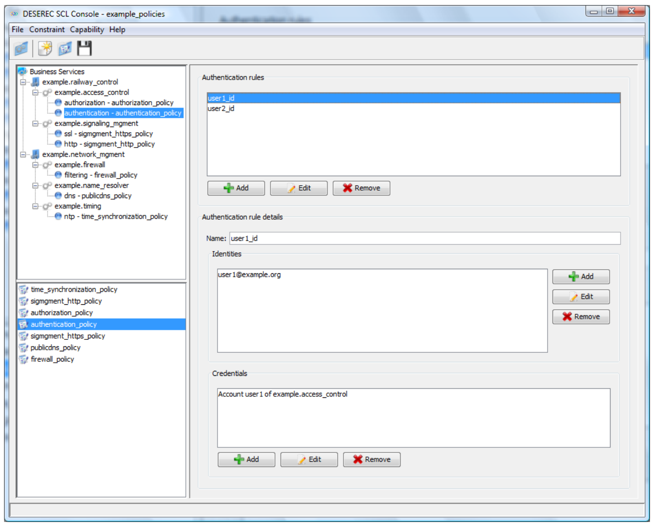
So far, in DESEREC, different kinds of policies have been modeled for both SCL and xCIM. Among them, we have security policies, such as authentication, authorization, filtering or SSL, and also common constraints to specify service configurations, such as HTTP, NTP, NAT, DNS, streaming, DHCP or load balancing. Anyway, both languages, SCL and xCIM, can be easily extended to hold any new kind of policy.

A. SCL Console

In order to assist the administrator with the task of defining service configuration constraints in SCL, DESEREC has developed an intuitive console that permits to generate and manage SCL models using a graphical interface.

Through the SCL Console [28], a system administrator can create a service constraints model in SCL which defines how the system is expected to operate. This is done by working on the system model (physical/logical infrastructure) and the service model (Business Services description, decomposition and interaction). This console generates the aforementioned constraints model (SCL), which contains the configuration constraints defined by the administrator, as well as the assignments between them and the present BSCs in the service model.

Figure 4 is a snapshot of the SCL Console showing the Business Services, BSCs, capabilities and service constraint policies defined for the illustrative example introduced in Section IV. The console presents three well-defined areas in the frame: one for browsing through the Business Services, BSCs and capabilities (top-left area); a bigger one for viewing and editing the SCL constraints (right area); and a last one for browsing through the constraints and assigning them to the BSCs (bottom-left area).


DESEREC SCL Console - example_policies
File Constraint Capability Help

Business Services
- example.railway_control
  - example.access_control
    - authorization - authorization_policy
    - authentication - authentication_policy
  - example.signaling_mgment
    - ssl - sigmgment_https_policy
    - http - sigmgment_http_policy
- example.network_mgment
  - example.firewall
    - filtering - firewall_policy
  - example.name_resolver
    - dns - publicdns_policy
  - example.timing
    - ntp - time_synchronization_policy

- time_synchronization_policy
- sigmgment_http_policy
- authorization_policy
- authentication_policy
- sigmgment_https_policy
- publicdns_policy
- firewall_policy

Authentication rules
- user1_id
- user2_id

Authentication rule details
Name: user1_id
Identities: user1@example.org
Credentials: Account user1 of example.access_control

Figure 4. Snapshot of the SCL Console

The SCL Console supports creation, edition and management of the whole set of configuration policy types, defined by DESEREC and mentioned in this section. Nevertheless, the constraints processing in the SCL Console is implemented by following a plug-in based approach. That is, each supported constraint is implemented as a separate plug-in which is loaded on-the-fly. This means that the set of supported constraints can be extended as desired, just by developing additional plug-ins. Moreover, the console is able to manage SCL models with constraint types for which there is no plug-in to support its edition or creation. In such a case, the SCL Console will show the constraint as an XML text, thereby allowing the administrator to work with it.

B. Service allocation

Once the system administrator has defined the service configuration constraints, the Planning block must automatically allocate, in a first step, each BSC onto the molecules at global level; that is, it will provide the set of the HOCs that will compose the final global plan. To this end, the allocation process maps the required capabilities that each BSC needs against the available capabilities that each molecule provides (defined previously in SDL). The next step of this process is to calculate all possible local allocations for each HOC generated previously, thus providing each of the LOCs that will compose each HOC. As before, this local allocation process is carried out by means of mapping the required capabilities that each BSC needs against the available capabilities that each technical service provides (also defined in SDL).

Note that if one allocation does not fulfill the requirements defined by the administrator, the service cannot be allocated and the corresponding Operational Configuration (HOC or LOC) is discarded.

Table II shows the output of this process for the running example. As can be seen, two HOCs have been generated, in which the only difference is the allocation of the Business Service *railway_control* (for both BSCs, *signaling_mgment* and *access_control*). In the first case (`HOC1`), both BSCs are allocated onto *molecule-1* and, in the second one (`HOC2`), they are allocated onto *molecule-2*. In both cases the Business Service *railway_control* can be allocated onto the two defined molecules since they provide the required capabilities needed by it.

At local level, each HOC generates one LOC: a first LOC belonging to `HOC1` (`HOC1.LOP`$_{1\&2}$`.LOC1`) in which the complete Business Service *railway_control* is allocated onto the *apache-2.2.4* software; and another first LOC belonging to `HOC2` (`HOC2.LOP`$_{1\&2}$`.LOC1`) in which the same Business Service is now allocated onto the *tomcat-5.5.20* software. Note that, for example, the BSC *signaling_mgment* cannot be allocated onto the *lighttpd-1.4.18* software since it does not provide one of the required capabilities (specifically the *ssl* one to provide secure connections), thus being discarded during this mapping process.

The LOCs introduced in Table II will be later split and packaged in LOPs depending on the molecule to which they are addressed. In this case, all the allocations to *molecule-1* will be packaged as LOP1, and those addressed to *molecule-2* will be packaged as LOP2.

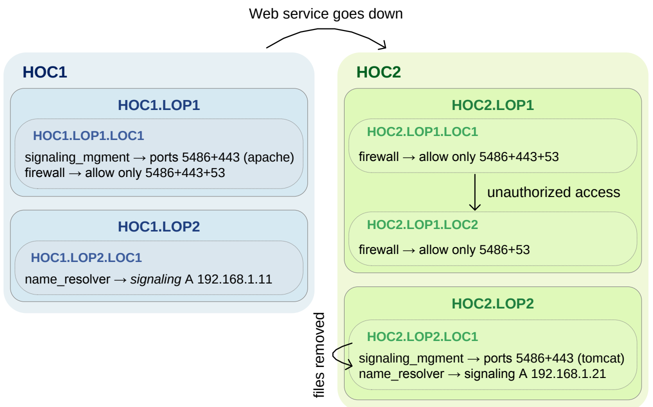

Web service goes down
HOC1
HOC1.LOP1
HOC1.LOP1.LOC1
signaling_mgment → ports 5486+443 (apache)
firewall → allow only 5486+443+53
HOC1.LOP2
HOC1.LOP2.LOC1
name_resolver → *signaling* A 192.168.1.11
HOC2
HOC2.LOP1
HOC2.LOP1.LOC1
firewall → allow only 5486+443+53
unauthorized access
HOC2.LOP1.LOC2
firewall → allow only 5486+53
HOC2.LOP2
HOC2.LOP2.LOC1
signaling_mgment → ports 5486+443 (tomcat)
name_resolver → signaling A 192.168.1.21
files removed

Figure 5. Complete allocation graph including detection/reaction logic

From these allocations, the Planning block generates the complete allocation graph, as the one shown in Figure 5 for the running example.

For clarity reasons, the BSCs *timing* and *access_control* have not been included since the former is always allocated in the same technical service (the only one that provides the required capability), and the latter is always allocated together with the BSC *signaling_mgment*. In this allocation graph, the allocation and configuration changes have also been included to follow how the DESEREC framework will work in runtime after detecting the aforementioned incidents.

Table II
ALL POSSIBLE ALLOCATIONS AT BOTH GLOBAL AND LOCAL LEVEL

GLOBAL LEVEL (mapping onto molecules)	LOCAL LEVEL (mapping onto technical services)
HOC1	**HOC1.LOP$_{1\&2}$.LOC1**
railway_control.signaling_mgment –> molecule-1	railway_control.signaling_mgment –> molecule-1.host-1.apache-2.2.4
railway_control.access_control –> molecule-1	railway_control.access_control –> molecule-1.host-1.apache-2.2.4
network_mgment.name_resolver –> molecule-2	network_mgment.name_resolver –> molecule-2.host-4.bind-9.3.4
network_mgment.timing –> molecule-1	network_mgment.timing –> molecule-1.host-1.ntpd-4.1.1
network_mgment.firewall –> molecule-1	network_mgment.firewall –> molecule-1.host-2.iptables-1.3.5
HOC2	**HOC2.LOP$_{1\&2}$.LOC1**
railway_control.signaling_mgment –> molecule-2	railway_control.signaling_mgment –> molecule-2.host-3.tomcat-5.5.20
railway_control.access_control –> molecule-2	railway_control.access_control –> molecule-2.host-3.tomcat-5.5.20
network_mgment.name_resolver –> molecule-2	network_mgment.name_resolver –> molecule-2.host-4.bind-9.3.4
network_mgment.timing –> molecule-1	network_mgment.timing –> molecule-1.host-1.ntpd-4.1.1
network_mgment.firewall –> molecule-1	network_mgment.firewall –> molecule-1.host-2.iptables-1.3.5

It is worth mentioning that in Figure 5 it has been added a second LOC in `HOC2.LOP1`, which represents the reaction thrown after detecting an unauthorized access (second problematic situation introduced in Section IV-B). This local reconfiguration is just a change in the firewall configuration but without implying changes in the service allocation; that is, the allocations are maintained exactly the same as the ones defined before the reaction.

C. Policy Refinement

Definition of high-level objectives is usually the way administrators work. To make these objectives a reality in terms of configuration, lots of information need to be provided to a refinement process, from high-level objectives to final configurations [29]. Afterwards, these configurations can then be deployed to the final devices and services, in order to maintain the system configured properly based on the administrator requirements. This avoids administrators to generate a wide range of different and specific configuration files for each device or service.

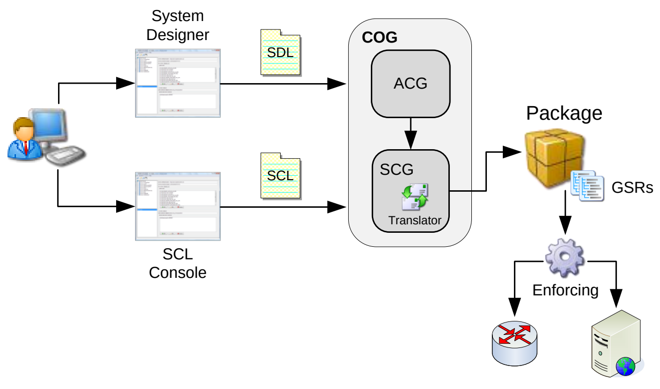


Figure 6. Refinement process workflow

In this context, a top-down engineering approach of this refinement process has been designed and implemented. Figure 6 depicts the refinement workflow followed in DESEREC. Firstly, the administrator defines both the Business Services and a complete system description using the *System Description Language* (SDL). The requirements, i.e., configuration constraints defining Business Services behavior, are also defined in SCL by means of the SCL Console as explained in Section VI-A. Then, all this information is used to generate the generic configuration model in xCIM format, which will be finally used by the reconfiguration framework to configure the system resources.

DESEREC relies on the *Configuration Generator* (COG) to perform the translation process (see Figure 6), which is the central part of the Planning block in the DESEREC framework. It is in charge of taking the requirements from the administrator in terms of a system description, Business Services specification and the desired behavior of services. Then, it produces as output an *Operational Plan* (OP) containing the configuration models in xCIM needed to enforce it. The OP will contain a list of alternative Operational Configurations, and the needed detection/reaction logic that will implement the dependability features, as explained in Section III-B.

There are two submodules in the COG, the *Automatic Configuration Generator* (ACG) and the *Services Configuration Generator* (SCG). The former generates a first version of the Operational Plan containing Operational Configurations; that is, allocations of Business Services onto infrastructure elements, supporting the molecule abstraction. The latter analyzes the allocations present in the Operational Configurations, adds semantics to the constraint model, and produces configuration packages as a *Generic Service Ruleset* (GSR) [20]. This is a generic model that will enable the DESEREC runtime framework to actually set up the software elements to operate as desired.

The SCG takes the Operational Configurations one by one and launches the process autonomously for each one. As a result, a versatile package is generated, which contains the GSRs as well as the whole information about the system where the GSR is assigned to. The SCG model has been developed as a Java application which allows loading and processing the Operational Plans in order to produce GSR packages in xCIM-SCL format. Indeed, the *Translator* is the SCG submodule in charge of translating from policies/constraints (defined in SCL) to xCIM-SCL.

In order to generate the GSRs properly, the *Translator* needs as input both the policy information to be refined and the information about the system that is being managed. For this second issue, DESEREC relies on the SDL language. Since the SDL model is also included as a part of the Operational Plan, this model is available to the translation process.

It is worth noting that the *Translator* module is composed of smaller and specific *translation submodules*, as many as different kinds of policies are supported by the DESEREC framework; i.e., HTTP, DNS, filtering, etc. Thus, each submodule is specialized on translating each kind of policy, taking into account its idiosyncrasy. This fact leads our approach to be easily extended with new kinds of policies. This refinement process is based on the one defined in [30].

```
<gsr:GSR xmlns:gsr="http://www.deserec.eu/xsd/gsr">
  <gsr:xDESEREC_DNSServerSettingData>
    <InstanceID>publicdns_policy</InstanceID>
    <Forwarders>198.41.0.4</Forwarders>
  </gsr:xDESEREC_DNSServerSettingData>
  <gsr:xDESEREC_DNSZoneSettingData>
    <InstanceID>publicdns_policy.zone1</InstanceID>
    <Domain>example.org</Domain>
    <Type>1</Type>
    <TimeToRefresh>7200</TimeToRefresh>
    ...
  </gsr:xDESEREC_DNSZoneSettingData>
  <gsr:xDESEREC_DNSRecordSettingData>
    <InstanceID>publicdns_policy.zone1.record3</InstanceID>
    <ElementName>publicdns_policy.zone1.record3</ElementName>
    <Query>signaling</Query>
    <Type>1</Type>
    <Response>192.168.1.11</Response>
  </gsr:xDESEREC_DNSRecordSettingData>
  <gsr:xCIM_ConcreteComponent>
    <GroupComponent>DESEREC_DNSZoneSettingData.InstanceID=
      'publicdns_policy.zone1'</GroupComponent>
    <PartComponent>DESEREC_DNSRecordSettingData.InstanceID=
      'publicdns_policy.zone1.record1'</PartComponent>
  </gsr:xCIM_ConcreteComponent>
  ...
  <gsr:xDESEREC_GSRHeader>
    <TransportationMethod>COPS−PR</TransportationMethod>
    <GSRTarget>system.netw.servers.M2.DNSServer.dnsd1</GSRTarget>
    <GSRTargetSoftware>
      system.netw.servers.M2.DNSServer.dnsd1.PublicDNSAGTsw
    </GSRTargetSoftware>
    <MoleculeID>system.netw.servers.M2</MoleculeID>
  </gsr:xDESEREC_GSRHeader>
</gsr:GSR>
```

Listing 2. xCIM representation of the DNS policy

Listing 2 shows how the translation process works, based on the running example defined in Section IV. It shows a fragment of a GSR that represents the aforementioned DNS policy, but now translated into xCIM format. The *Translator* generates this GSR document taking into account the SCL policy, the allocation information and the system description in SDL.

At first sight, it can be noticed that the xCIM language is more complex and, therefore, harder to understand than SCL. For instance, it codifies different kinds of SCL options by means of numbers. In order to understand this point, let us compare the Listing 2 with its equivalent specified in SCL, and shown in Listing 1. The DNS zone typed as *Master* in the SCL policy is now codified as $< Type > 1 < /Type >$ inside the xCIM class *xDESEREC_DNSZoneSettingData*.

Furthermore, it is important to note how the *Translator* resolves the BSC references in the DNS policy to IP addresses; e.g., the previous *A* record response reference called *example.signaling_mgment* has been replaced by its corresponding IP according to the allocations defined in `HOC1.LOP1.LOC1` of the running example. Note that it is possible since the allocation process is done before the translation process takes place.

Additionally, every GSR maintains some control information which is later used by the DESEREC framework to perform the enforcement and other operations. Thus, the xCIM class *xDESEREC_GSRHeader* contains some useful parameters, such as the reference to the target element where the GSR is going to be enforced or the transportation method used in such an enforcement.

Listing 3 shows another GSR, but now with respect to the access control policies that can be also found in SCL, and shown in Listing 1.

```
<gsr:GSR xmlns:gsr="http://www.deserec.eu/xsd/gsr">
  <gsr:xCIM_Role>
    <CreationClassName>CIM_Role</CreationClassName>
    <Name>authorization_policy.role.signaling_admins</Name>
    <CommonName>signaling_admins</CommonName>
    <ElementName>signaling_admins</ElementName>
  </gsr:xCIM_Role>
  <gsr:xCIM_Identity>
    <InstanceID>authorization_policy.role.signaling_admins.identity.
      user1@example.org</InstanceID>
    <ElementName>user1@deserec.org</ElementName>
  </gsr:xCIM_Identity>
  <gsr:xCIM_MemberOfCollection>
    <Collection>CIM_Role.CreationClassName='CIM_Role',
      Name='authorization_policy.role.signaling_admins'</Collection>
    <Member>CIM_Identity.InstanceID='authorization_policy.role.
      signaling_admins.identity.user1@example.org'</Member>
  </gsr:xCIM_MemberOfCollection>
  <gsr:xCIM_Privilege>
    <InstanceID>authorization_policy.allow_signaling_mgment.
      signaling_mgment_access</InstanceID>
    <PrivilegeGranted>true</PrivilegeGranted>
    <Activities>5</Activities>
    <Activities>6</Activities>
    <QualifierFormats>11</QualifierFormats>
  </gsr:xCIM_Privilege>
  <gsr:xCIM_AuthorizationRuleAppliesToPrivilege>
    <PolicySet>
     CIM_AuthorizationRule.SystemCreationClassName='CIM_AdminDomain',
     SystemName='system.netw.servers.M1',PolicyRuleName='
     authorization_policy.allow_signaling_mgment'</PolicySet>
    <ManagedElement>CIM_Privilege.InstanceID='authorization_policy.
      allow_signaling_mgment.signaling_mgment_access'</ManagedElement>
  </gsr:xCIM_AuthorizationRuleAppliesToPrivilege>
  <gsr:xCIM_AuthorizationRuleAppliesToTarget>
    <PolicySet>
     CIM_AuthorizationRule.SystemCreationClassName='CIM_AdminDomain',
     SystemName='system.netw.servers.M1',PolicyRuleName='
     authorization_policy.allow_signaling_mgment'</PolicySet>
    <ManagedElement>signaling.example.org</ManagedElement>
  </gsr:xCIM_AuthorizationRuleAppliesToTarget>
  ...
  <gsr:xCIM_AuthorizationRule>
    <SystemCreationClassName>
      CIM_AdminDomain
    </SystemCreationClassName>
    <SystemName>system.netw.servers.M1</SystemName>
    <CreationClassName>CIM_AuthenticationRule</CreationClassName>
    <PolicyRuleName>
      authorization_policy.allow_signaling_mgment
    </PolicyRuleName>
  </gsr:xCIM_AuthorizationRule>
  <gsr:xCIM_AuthorizationRuleAppliesToRole>
    <PolicySet>CIM_AuthorizationRule.SystemCreationClassName=
      'CIM_AdminDomain',SystemName='system.netw.servers.M1',
      PolicyRuleName='authorization_policy.
      allow_signaling_mgment'</PolicySet>
    <ManagedElement>CIM_Role.CreationClassName='CIM_Role',
      Name='authorization_policy.role.
      signaling_admins'</ManagedElement>
  </gsr:xCIM_AuthorizationRuleAppliesToRole>
  <gsr:xCIM_PolicyRuleInSystem>
    <Antecedent>CIM_AdminDomain.CreationClassName='CIM_AdminDomain',
      Name='system.netw.servers.M1'</Antecedent>
    <Dependent>CIM_AuthorizationRule.SystemCreationClassName=
      'CIM_AdminDomain',SystemName='system.netw.servers.M1',
      PolicyRuleName='authorization_policy.
      allow_signaling_mgment'</Dependent>
  </gsr:xCIM_PolicyRuleInSystem>

  <!-- Authentication Policy -->

  <gsr:xCIM_AuthenticationRule>
    <SystemCreationClassName>
      CIM_AdminDomain
    </SystemCreationClassName>
    <SystemName>system.netw.servers.M1</SystemName>
    <CreationClassName>CIM_AuthenticationRule</CreationClassName>
    <PolicyRuleName>authentication_policy.user1_id</PolicyRuleName>
  </gsr:xCIM_AuthenticationRule>
  ...

  <gsr:xDESEREC_GSRHeader>
    <TransportationMethod>COPS-PR</TransportationMethod>
    <GSRTarget>system.netw.servers.M1.Apache2</GSRTarget>
    <GSRTargetSoftware>
      system.netw.servers.M1.Apache2.AAAsw
    </GSRTargetSoftware>
    <MoleculeID>system.netw.servers.M1</MoleculeID>
  </gsr:xDESEREC_GSRHeader>
</gsr:GSR>
```

Listing 3. xCIM representation of the access control policy

At the sight of the above fragment of XML document, both authentication and authorization policies share the same target element in this GSR and, therefore, the GSR header is unique for both of them. It means that this configuration will be enforced in the same software of the same machine for a given molecule. Please bear in mind that the *Translator* module will generate some other GSRs, which are able to configure other target elements according to the allocations established before, always taking the same SCL access control policies as input. Moreover, as can be seen, the xCIM language is still defined as a generic configuration and it is not linked to any implementation or particular software. It will be the enforcing mechanisms, described in section VII-D, the ones in charge of generating the actual configuration files depending on the concrete software.

As before, and for the sake of clarity, the authentication policy has been nearly omitted from the GSR and it is not shown in Listing 3.

VII. Reconfiguration Framework

This section introduces an in-depth explanation of the reconfiguration framework, its modules and components. We also present a complete illustrative example as demonstration of the proposed framework.

A. General Requirements

In this subsection we summarize the general requirements that have been identified by DESEREC, and that our framework should fulfill. Among them, we include scalability, language interoperability, security assurance, autonomy and, finally, issues related to service continuity and reliability of reconfiguration. These requirements are summarized as follows:

- Only well-characterized incidents shall be treated, and their analysis needs to be very fast and non ambiguous for detecting an incident in runtime.
- The reaction process shall be automatically carried out as soon as possible after the detection of an incident. Therefore, human interaction cannot take place in this process, although the system administrator could be alerted with high priority.
- Strong mechanisms must be provided and supported to avoid intrusion.
- It is necessary to use as much standard languages as possible to exchange consistent information between heterogeneous managed components (target system components and the DESEREC framework).
- A distributed solution should be designed since large systems will produce a great amount of events that must be processed.
- The detection and reaction processes should take a maximum time interval of a few minutes in order to maintain the service continuity.
- This framework must provide mechanisms to guarantee the integrity of the requested reconfiguration since it could provoke a breakdown of the service if it is corrupted.

B. Workflow of the Reconfiguration Framework

Once the configuration information is released, by using the planning tools described above, the reconfiguration framework works autonomously without human intervention. This information must be deployed through the DESEREC architecture and applied both in the corresponding technical services, for configuring them properly, and in the different modules of the architecture for detection and decision purposes.

Figure 7 depicts the complete workflow inside this framework, labeled with the information exchanged between the modules. This exchange is done using SOAP-based Web service interfaces. Please note that only the local reconfiguration framework is shown in Figure 7, although it could be extended to the global one since both levels (global and

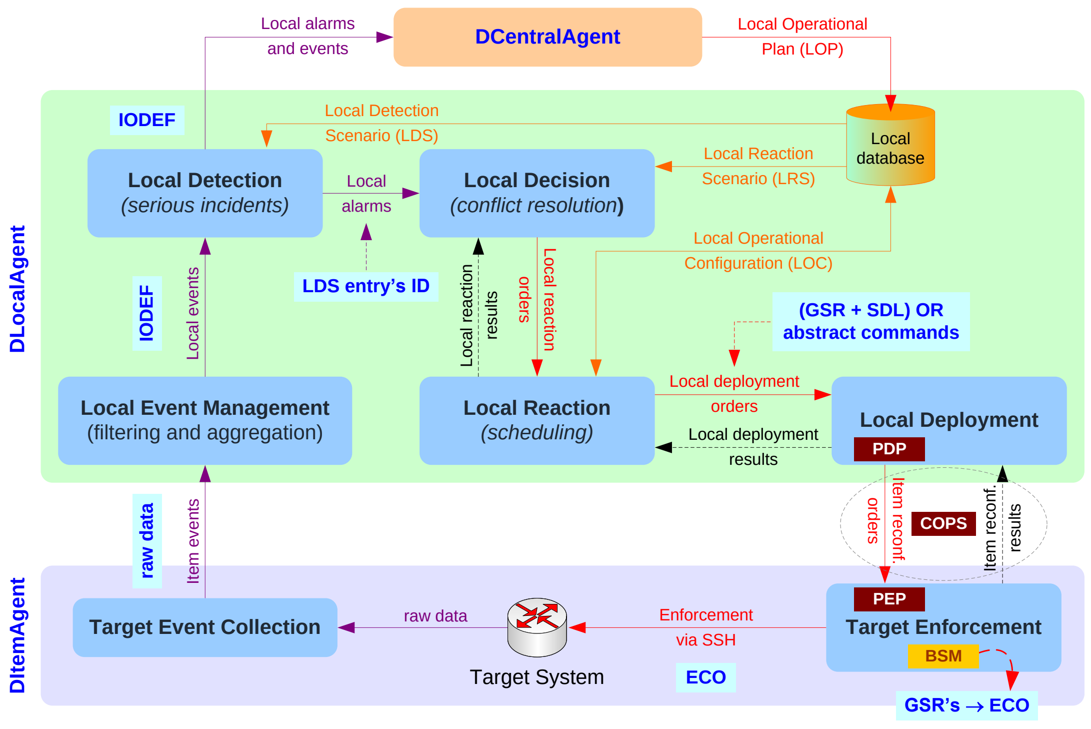


Figure 7. Local reconfiguration framework

local) work in a similar way to be composed of the same modules.

Through the *Target Event Collection*, which collects raw events from the target elements, the *Local Event Management* is continuously gathering item events, filtering and aggregating them to provide higher level events to the upper modules. This will avoid overloading these modules and will reduce the bandwidth occupation. Furthermore, the *Local Event Management* module also transforms the collected item events into a normalized format and sends them to the *Local Detection*. In our case, the chosen exchange format is IODEF [31] since it is a W3C standard format defined to represent and exchange operational and statistical information between components.

The Detection modules are constantly receiving events from the lower levels. DLocalAgents retrieve them from the target system through the DItemAgents, whereas the DCentralAgent retrieves them from the DLocalAgents. These events are matched against the possible problems defined in the *Detection Scenario* (GDS or LDS, depending on the level or agent) with the help of a set of signature-based rules included in that scenario. If there is a coincidence, it means that this module has detected a fault in the system and a response is required.

The Detection module notifies to the Decision one which alarm has occurred; that is, the problem that has been detected. In addition, if the problem has been detected at local level, the *Local Detection* module also forwards the appropriate alarm, i.e., the IODEF itself and some additional information about the problem, to the DCentralAgent.

The Decision module retrieves from the *Reaction Scenario* (GRS or LRS, depending on the level or agent) a list of possible reactions to solve the previously raised problem. Note that for each problem or alarm we have a list of $[1..n]$ reactions. This module will decide the most suitable reaction to carry out taking into account the statistical data harvested from the target system. The current system situation could also be taken into account for this decision-making process to choose the best reaction in a particular moment. When taking this decision it will always first try to apply a local reaction, which will be faster and less costly; otherwise, the DCentralAgent will be informed to take the corresponding global reconfiguration, if necessary.

At global level, the *Global Decision* module sends to the *Global Reaction* the HOC identifier with the new configuration to deploy in the system. On the other hand, at local level, the *Local Decision* module sends to the *Local Reaction* the XML-based reaction to apply, which includes either the LOC identifier with the new local configuration or a set of abstract commands to be executed to fix the problem.

The Reaction module retrieves the appropriate *Operational Configuration* (HOC or LOC) from its local database using the identifier (HOC ID or LOC ID) sent by the Decision module. Note that if the reaction is to apply a set of abstract commands, the Reaction module does not need to retrieve any information since these commands are already included in the XML-based reaction instance itself. Later on, this last module queues the set of deployment orders and sends them progressively to the Deployment module; until an order is not correctly deployed and enforced, the next one is not sent. In the *Global Reaction* these orders will be the LOPs contained in the new HOC, whereas in the *Local Reaction* they will be the GSRs contained in the LOC (one GSR per service to be configured).

The Deployment module delivers the reconfiguration orders to the lower agents which will process them in a different way, depending on the agent level involved:

- At global level, each LOP sent by the *Global Reaction* module (one per molecule) is forwarded to the corresponding molecule associated with that LOP. This process is repeated for each molecule, in which each LOP is stored in its own repository. Then, the *Local Decision* module launches the deployment of the first configuration for the new LOP with the best possible LOC. During this phase, the *Local Detection* and the *Local Decision* modules are automatically reconfigured with the new scenarios specified in the new LOP as well.
- At local level, the GSR received by the *Local Deployment* module is sent to the appropriate *Target Enforcement* module which manages the underlying software. Each GSR will be translated to *End COnfigurations* (ECOs), thanks to the *Block Service Module* (BSM) submodule, see Section VII-D, just before enforcing it in the target system. It is worth noting that a local deployment order could also be a set of abstract commands, which also have to be translated to target-specific commands for being enforced in the system.

It is worth mentioning that during all this process, as can be seen in Figure 7, feedback information is sent to upper modules and/or layers for reporting the sending and proper execution of the requested orders. If a deployment error is reported, e.g., an Operational Configuration becomes unfeasible in that moment, the upper layers will then have to decide which of the rest of available Operational Configurations could be deployed as valid, taking into account the current system situation.

C. PBNM Approach for Deployment

The last action in the reconfiguration framework is to distribute and enforce the GSRs into the final technical services. By this, the deployment phase is based on a *Policy-Based Network Management* (PBNM) approach [23], which allows deploying, installing and enforcing policies in the target devices.

This architecture is basically composed of the following four elements:

1) *Policy Decision Point* (PDP). The PDP processes the policies of the system, along with other data such as network state information, and takes policy decisions regarding which policies should be enforced, and how and when this will happen. These policies are sent as configuration data to the appropriate PEPs.
2) *Policy Enforcement Point* (PEP). The PEP is communicated to the managed devices and it is responsible for installing and enforcing the policies sent by the corresponding PDP.
3) Policy repository. This is a policy database which the PDP uses for its decision-making process.
4) Target system. The final target device or element in which the above policies will be enforced.

Regarding the communication protocol between the PDP and its PEPs, the IETF has been focused on the definition of the *Common Open Policy Service* (COPS) protocol [32]. COPS is a simple query and response protocol based on a client-server model that can be used to exchange policy information between a policy server (PDP) and its clients (PEPs).

The inclusion of such a policy-based framework in the DESEREC architecture has been performed as follows: one PDP is included in each DLocalAgent, i.e., one PDP per molecule, whereas one PEP is placed in each DItemAgent. The policy repository is a database (local to the PDP) which receives policy information from the *Local Reaction* module and caches it for both performance and autonomy purposes.

When a new configuration needs to be deployed, the PEP can get the appropriate policy information from its PDP, adapt it (if needed) to the particular device which is being managed and, lastly, enforce it. This model also supports enforcement feedback, via the COPS reports which PEPs can send back to their corresponding PDP.

D. Block Service Module

The policy data provided by a PDP to one of its PEPs may need to be tuned for a specific managed device. This may include not only a change in the notation, but also other specific information; for example, updating the configuration of a service might require different steps to be taken, depending on the particular implementation of that service.

In the DESEREC architecture, just before enforcing the GSRs by the PEP, they should be translated according to the final software installed on the system, e.g., product, version, etc., since the GSRs represent software-independent configurations. This translation is performed by the *Block Service Module* (BSM) [20], using specific translation templates which generate the final device-specific configurations, called *End COnfigurations* (ECO). Finally, these configurations are enforced in the target device by the PEP that manages it through enforcement protocols like SSH/SCP or SNMP. The framework also allows the usage of proprietary protocols, depending on the managed software.

E. Illustrative Example Scenario

This section shows how the DESEREC framework, explained above, is used as a proof of concept to react at runtime when a problem in the system turns up. It describes a set of problematic situations that could cause different dependability and security problems which the DESEREC framework should fix, taking the testbed description in Section IV-B as the design and lab implementation.

Situation 0: everything is fine

This situation shows the whole system working properly in a nominal case. That is, interactions with the railway signaling and control system are made through the provided Web service and administrators can access the management website to request statistics and monitoring information.

Initially, during the first configuration of the DESEREC-managed system, the Global Detection and Reaction Scenarios (GDS and GRS) are automatically stored in the Global Detection and Decision modules, respectively, in order to provide detection and reaction logic at global level. As can be seen in Figure 5, the GDS contains at this level a signature-based rule of the type "*Web service goes down*". The associated reaction, defined in the GRS, will be switching between high-level configurations; in this case, from `HOC1` to `HOC2`.

Then, the most suitable global configuration (`HOC1` in this case) is released and distributed through all the DESEREC components, until configuring the technical services as the configuration policies dictate. At global level, the DCentralAgent extracts from `HOC1` each LOP that will configure each of the defined molecules; that is, `HOC1.LOP1` will be addressed to *molecule-1* and `HOC1.LOP2` to *molecule-2*. In both cases, the corresponding LOP is stored into the local database of each DLocalAgent. The local detection and reaction logic (LDS and LRS) included in each LOP, is automatically stored in the Local Detection and Decision modules. In this example, both local scenarios are empty without defining any reaction capacity.

The most suitable configurations (`HOC1.LOP1.LOC1` in *molecule-1* and `HOC1.LOP2.LOC1` in *molecule-2*) at local level are then deployed to the corresponding target software, with the aim of configuring them properly. In this example, and according to the service allocations defined above, the `HOC1.LOP1.LOC1` contains the GSR belonging to the BSC *signaling_mgment*, which is delivered to the DItemAgent that manages the *apache-2.2.4* software, and the GSR of the BSC *firewall*, which is delivered to the DItemAgent that manages the *iptables-1.3.5* software. Note that each DItemAgent will translate these GSRs to ECOs (final configurations) just before enforcing them in the underlying technical service.

On the other hand, the `HOC1.LOP2.LOC1` contains the GSR associated to the BSC *name_resolver*, which is delivered to the DItemAgent that manages the *bind-9.3.4* software. That DItemAgent will translate it to BIND format before finally enforcing it. This last GSR can be seen in Listing 2, which contains the entire required DNS configuration at high level: information about the zone for the `example.org` domain; and an "A" record to resolve the *signaling* name to a specific IP address for pointing out that the railway control Web service is running on `192.168.1.11`.

Note that although the BSCs *timing* and *access_control* have not been included in Figure 5, their GSRs are also deployed together with the previous ones. The GSR for the BSC *timing* will be always delivered to the DItemAgent that manages the *ntpd-4.1.1* software, independently of the operational plan and configuration, since this is the only technical service that can provide it. On the other hand, the GSR for the BSC *access_control* (see Listing 3) will be delivered to the DItemAgent that manages the *apache-2.2.4* software as defined by the service allocation.

As can be seen, two GSRs are addressed to the same *apache-2.2.4* software for configuring, in the same technical service, the BSCs *signaling_mgment* and *access_control*.

After all this process, the final configuration of the system is as the one depicted in Figure 8a), labeled with the running OCs and scenarios for each agent.

Situation 1 (reallocation): the Web service has become unavailable

This situation shows a global reconfiguration process, i.e., the DCentralAgent is involved on it, when the Web service becomes down and needs to be reallocated. This could be due to the fact that a DoS attack has been performed from a compromised host in the Intranet, an internal error of the Web server itself, etc. As a consequence, the Web server goes down.

Through the left-hand side of the framework, i.e., the monitoring part, shown in Figure 7, different local events harvested from the target system are going up on both molecules until their *Local Detection* modules.

Suddenly, one of these events in DLocalAgent-1 carries a possible problem of the type "*the Web service has become unavailable*". Since the *Local Detection* module in DLocalAgent-1 has no rule in its LDS, as can be seen in Figure 8a), this event is forwarded directly to the DCentralAgent for being managed at global level if necessary. Once the DCentralAgent receives the above event including the actual problem, sent in this case by the DLocalAgent-1 where the service was running, it is capable of detecting that a possible problem has occurred by mapping the event against its GDS. In this example, the GDS includes a global detection rule of the type "*Web service goes down*" and the framework needs to react for fixing it. The associated reaction is to

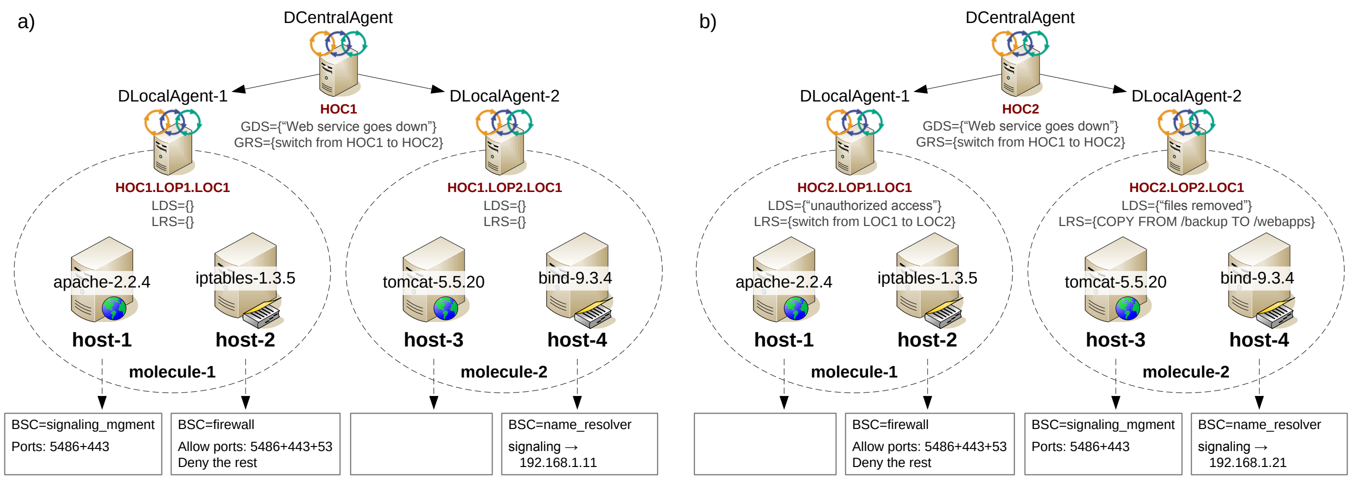


Figure 8. System status: a) initial configuration, situation 0; and b) after a reallocation, situation 1

move (reallocate) that service to another place where it is able to be provided again; in this case, by switching from the current global configuration (`HOC1`) to a new one (`HOC2`) as is defined in the GRS.

As in the previous situation, the new global configuration (`HOC2`) is distributed through all the DESEREC components. Because of this, two new LOPs (`HOC2.LOP1` for *molecule-1* and `HOC2.LOP2` for *molecule-2*) are sent to their corresponding DLocalAgents for deploying a new configuration. After being stored into their local databases, the local detection and reaction logic is updated as the new LOPs dictate. Looking again at Figure 5, the LDS of `HOC2.LOP1` contains a local detection rule of the type "*unauthorized access*" whose reaction (which is later used in Situation 2) is to switch from `HOC2.LOP1.LOC1` to `HOC2.LOP1.LOC2`. On the other hand, the LDS of `HOC2.LOP2` contains another local detection rule of the type "*files removed*" whose reaction (which is later used in Situation 3) is to execute an abstract command to *COPY* these files *FROM* a backup folder *TO* the appropriate place.

After this, the most suitable local configurations for both LOPs are then deployed (`HOC2.LOP1.LOC1` in *molecule-1* and `HOC2.LOP2.LOC1` in *molecule-2*) in a similar way as the one described in Situation 0, letting the system as shown in Figure 8b): the railway control Web service running again, but now in a different allocation (namely, in the *tomcat-5.5.20* software); the BSC *firewall* remains unchanged as before; and a new configuration has been applied in the *bind-9.3.4* software reflecting the reallocation change of the BSC *signaling_mgment* (this service is now running on `192.168.1.21`).

It is worth pointing out that, at global level, the GDS and GRS remain the same that before, although they will have no effect since the rules included in them are only applicable to `HOC1`, and the current running configuration after executing this situation is `HOC2`.

Situation 2 (reconfiguration): unauthorized access to a private resource

In this situation we show how the DESEREC framework is able to reconfigure a service. In this sense, we suppose that an unauthorized user gains access to the website. The response of the framework will be to put into quarantine the website by establishing a more restrictive configuration in the firewall component.

Only the DLocalAgent-1 will be into play in this case since, after locally receiving a possible problem of the type "*a user has made an unauthorized access to a resource*", the *Local Detection* module in DLocalAgent-1 has a location detection rule in its LDS, as can be seen in Figure 8b), that will throw a local alarm. The reaction associated to this incident, defined in the LRS, is to deploy a new local configuration (by switching between `HOC2.LOP1.LOC1` and `HOC2.LOP1.LOC2`) to reconfigure the firewall with a more restrictive filtering rules. In this new configuration, the firewall will only permit connections on ports 5486 and 53, blocking the access to the 443 (HTTPS) port.

Note that, despite reconfiguring a service, the allocations are maintained as before since the BSC *firewall* still continues to run on the same technical service, but now with a different configuration. As seen in this situation, the *molecule-2* remains the same since the reaction has been carried out locally in *molecule-1* without involving the rest.

Situation 3 (abstract command execution): some important files have been removed

In this last situation we show a third sort of reaction that the DESEREC framework is capable of managing by means of executing a set of abstract commands. Due to a temporal hard disk fail, important Web service files have been removed and the Web service becomes inconsistent. In this case, when a request is made to the Web service, it will trigger an internal error.

Once the DESEREC framework catches that internal error by means of the events received from the target system, a possible problem of the type "*some files have been either removed or modified*" is detected. In this last situation, the associated reaction to this alarm (defined in the LRS) is sending of an abstract command which will copy the removed files from a backup folder to the appropriate place in the Web server. This abstract command is generically defined as `COPY FROM <source> TO <destination>`, which will have to be translated to the specific command depending on the system where it will be enforced. For example, in Linux systems this abstract command will be translated to `cp -r /backup /webapps`.

As before, this situation does not suppose any change in the allocation map, although in this case neither in the configurations. The reaction is only a very slight adjustment in the target system without changing anything in the running allocations and configurations.

VIII. Deployment and Validation

In this section we summarize how to design a dependable system following the proposed framework, as well as our experience in its deployment and how it has been validated and tested in existing mission critical systems.

A. Deployment of the DESEREC framework

The administrator(s) of the system, where the proposed framework has to be installed, must initially provide the molecule breakdown of the system. This will usually be done from scratch, although it could be also done in a non-intrusive mode by starting from an existing CIS. In both cases, the system administrator knows the Business Services that should be provided, for making them secure and thereby ensuring a given QoS. These Business Services have to be decomposed into Business Service Components for which the system administrator should define a list of constraints or policies using SCL. On the other hand, a system description should be provided by means of SDL, containing subsets of the whole CIS that are able to support technical services; that is, molecules to provide the required Business Services. In turn, these molecules have to be decomposed into software and network components, up to the level where they can be monitored, configured and deployed.

In this process, the administrator defines a synthetic and graphical view of the technical services, as well as the linked molecules. With them, and thanks to the Planning tools, the administrator can simulate errors, crashes, security attacks, etc., which means that few molecules could become unavailable. In each case, our framework is able to compute the best allocation of available molecules (according to the defined high-level policies), with the minimum number of reconfiguration steps involving molecule instances.

From the above defined molecule type description, the system administrator can generate and deploy the global and local configurations, together with their reaction plans. From this moment, it starts the runtime execution of the reconfiguration framework, in an autonomous way, as presented in Section VII.

B. Validation in real environments

The fulfillment of the DESEREC objectives have been evaluated by taking three typical cases of critical systems, which were provided by three end-user partners belong to the DESEREC consortium; namely:

- Hellenic Telecommunications Organization (OTE) [33]. OTE is the leading telecommunication operator in Greece and the Balkan area and, as such, operates most of the critical telecom infrastructures installed in that country. Therefore, securing its telecom infrastructure is a critical issue. The exploitation of the DESEREC project results has been very interesting in its lab testbed through a *TV over IP* (IPTV) scenario. It is worth mentioning that on this testbed the DESEREC consortium presented its results to the European Commission through a final demonstration at OTE premises.
- EADS Defence & Security Systems (DS) [34]. This partner provided to the consortium its *Security Command and Control System*, as main provider for the French Army. The main goal in this testbed was to minimize the risk exposure through protecting people and territories. In this case, EADS proposed a scenario where a border guard checks the passport of a person which is detected as blacklisted. Then, the border guard creates an alarm in the Web application in order to signal the problem, which is dispatched to the *Command and Control* application thanks to the *Enterprise Service Bus* (ESB). However, the border security employee plugs a USB key on the computer that hosts the ESB and it executes a malicious code which is present on his key. The code triggers an anomaly in the ESB service, which goes down and is no more able to forward client requests to the access control system. The DESEREC framework was successfully deployed to compute and execute the appropriate reaction, by making the service available again.
- RENFE-Operadora [35]. RENFE is the national railway operator in Spain, providing public service of train transportation for both passengers and trade goods. In this case, the DESEREC partners used the RENFE testbed to test a first approach of the proposed framework. These tests were focused on the railway signaling and control system, presented along with this paper, as well as the management of the *Ticket Selling* service.

IX. Conclusion and Future Work

In this paper we have presented a specific framework for managing service dependability in a policy based fashion. The concept of policy based management has been around in the research scene for several years now, with proven validity as an intuitive and scalable way for administrators to keep large information systems under control, ensuring the continuous enforcement of domain directives. Here we have checked how building a dependability management framework on a policy based core has indeed achieved to leverage the potential of this paradigm, applying it to a novel field. The proposed framework allows using the same abstract approach inherent to policy based solutions for managing also the automatic, on-demand configuration of system services.

Lastly, the tools developed for administrator interaction have allowed putting this proposal into practice, serving as further validation of the claimed achievements. By this, a complete example has been developed throughout this paper to stage how the proposed framework works in a fully autonomous way without human intervention.

As future work, some extensions remain to be taken into account which would improve this framework considerably. The current configuration policies that govern the system should be extended to include also setting up in a similar manner our own modules belonging to the framework; for example, the collection of sensors needed for a concrete operational plan, indicating the configuration for each of them.

Acknowledgment

This work has been funded by the DESEREC EU IST Project (IST-2004-026600), within the EC Sixth Framework Programme (FP6). Thanks also to the Funding Program for Research Groups of Excellence granted by the Seneca Foundation with code 04552/GERM/06.

References

[1] J. Bernal Bernabé, J.M. Marín Pérez, D.J. Martínez Manzano, M. Gil Pérez, and A.F. Gómez Skarmeta. “Towards a Policy-driven Framework for Managing Service Dependability”. In *DEPEND ’09: Proceedings of the 2nd International Conference on Dependability*, pages 66–72, 2009.

[2] E. Amidi. “System and Method for Providing a Backup-Restore Solution for Active-Standby Service Management Systems”. U.S. Patent Application 20060078092, April 2006.

[3] A. Avizienis, J. Laprie, and B. Randell. “Fundamental Concepts of Dependability”. Research Report 1145, LAAS-CNRS, April 2001.

[4] The SERENITY EU-IST Project (System Engineering for Security & Dependability). http://www.serenity-project.org [22 February 2010].

[5] R. Sterritt and D. Bustard. “Towards an Autonomic Computing Environment”. In *DEXA ’03: Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, pages 699–703, 2003.

[6] R. Sterritt and D. Bustard. “Autonomic Computing-a Means of Achieving Dependability?”. In *ECBS ’03: Proceedings of IEEE International Conference on the Engineering of Computer Based Systems*, pages 247–251, 2003.

[7] D.C. Verma. “Simplifying Network Administration using Policy-based Management”. *IEEE Network Magazine*, 16(2):20–26, 2002.

[8] J. Schönwälder, A. Pras, and J.P. Martin-Flatin. “On the Future of Internet Management Technologies”. *IEEE Communications Magazine*, 41(10):90–97, 2003.

[9] The DESEREC EU-IST Project (Dependability and Security by Enhanced Reconfigurability). http://www.deserec.eu [22 February 2010].

[10] L. Ardissono, A. Felfernig, G. Friedrich, A. Goy, D. Jannach, G. Petrone, R. Schäfer, and M. Zanker. “A Framework for the Development of Personalized, Distributed Web-based Configuration Systems”. *AI Magazine*, 24(3):93–108, 2003.

[11] A. Felfernig, G.E. Friedrich, and D. Jannach. “UML as Domain-Specific Language for the Construction of Knowledge-based Configuration Systems”. *International Journal of Software Engineering and Knowledge Engineering*, 10:449–469, 2000.

[12] M. Caporuscio, A. Di Marco, and P. Inverardi. “Model-Based System Reconfiguration for Dynamic Performance Management”. *Journal of Systems and Software*, 80(4):455–473, 2007.

[13] M. Mikic-rakic, S. Malek, and N. Medvidovic. “Improving Availability in Large, Distributed, Component-Based Systems via Redeployment”. In *CD ’05: Proceedings of the 3rd International Working Conference on Component Deployment*, pages 83–98, 2005.

[14] G. Spanoudakis, A. Maa Gomez, and S. Kokolakis. *“Security and Dependability for Ambient Intelligence”*. Springer Publishing Company, Incorporated, 2009.

[15] “Willow Survivability Architecture”, Dependability Research Group, University of Virginia. http://dependability.cs.virginia.edu/research/willow [22 February 2010].

[16] Z. Hill, J. Rowanhill, A. Nguyen-Tuong, G. Wasson, J. Knight, J. Basney, and M. Humphrey. “Meeting Virtual Organization Performance Goals through Adaptive Grid Reconfiguration”. In *GRID ’07: Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, pages 177–184, 2007.

[17] Z. Hill and M. Humphrey. “Applicability of the Willow Architecture for Cloud Management”. In *ACDC ’09: Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds*, pages 31–36, 2009.

[18] D.J. Martínez Manzano, M. Gil Pérez, G. López Millán, and A.F. Gómez Skarmeta. "A Proposal for the Definition of Operational Plans to provide Dependability and Security". In *CRITIS '07: Proceedings of the 2nd International Workshop on Critical Information Infrastructures Security*, pages 223–234, 2007.

[19] A. Cotton, M. Israël, and J. Borgel. "Molecular Approach Paves the Way towards High Resilience for Large Mission-Critical Information Systems". In *SECURWARE '08: Proceedings of the 2nd International Conference on Emerging Security Information, Systems and Technologies*, pages 332–337, 2008.

[20] The DESEREC EU-IST Project. Deliverable D2.1, "Policy and System Models", March 2007.

[21] M.D. Aime, P.C. Pomi, and M. Vallini. "Policy-Driven System Configuration for Dependability". In *SECURWARE '08: Proceedings of the 2nd International Conference on Emerging Security Information, Systems and Technologies*, pages 420–425, 2008.

[22] "Web Services Choreography Description Language v1.0". W3C Candidate Recommendation 9, November 2005. http://www.w3.org/TR/ws-cdl-10 [22 February 2010].

[23] G. Martínez Pérez, A.F. Gómez Skarmeta, S. Zeber, J. Spagnolo, and T. Symchych. "Dynamic Policy-Based Network Management for a Secure Coalition Environment". *IEEE Communications Magazine*, 44(11):58–64, 2006.

[24] "Common Information Model (CIM) Standards", Distributed Management Task Force, Inc. http://www.dmtf.org/standards/cim [22 February 2010].

[25] M. Debusmann and A. Keller. "SLA-driven Management of Distributed Systems using the Common Information Model". In *IM '03: Proceeding of the 8th IFIP/IEEE International Symposium on Integrated Network Management*, pages 563–576, 2003.

[26] H. Mao, L. Huang, and M. Li. "Web Resource Monitoring Based on Common Information Model". In *APSCC '06: Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing*, pages 520–525, 2006.

[27] C. Hobbs. *"A Practical Approach to WBEM/CIM Management"*. CRC Press, April 2004.

[28] "SCL Console and User Manual v1.0", University of Murcia. http://deserec.inf.um.es/console [22 February 2010].

[29] J. Rubio Loyola. *"A Methodological Approach to Policy Refinement in Policy-based Management Systems"*. PhD thesis, Technical University of Catalonia, Spain, April 2007.

[30] J.M. Marín Pérez, J. Bernal Bernabé, J.D. Jiménez Re, G. Martínez Pérez, and A.F. Gómez Skarmeta. "A Proposal for Translating from High-Level Security Objectives to Low-Level Configurations". In *SVM '07: Proceedings of the 1st International DMTF Academic Alliance Workshop on Systems and Virtualization Management: Standards and New Technologies*, 2007.

[31] R. Danyliw, J. Meijer, and Y. Demchenko. "The Incident Object Description Exchange Format". IETF RFC 5070, December 2007.

[32] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. "The COPS (Common Open Policy Service) Protocol". IETF RFC 2748, January 2000.

[33] Hellenic Telecommunications Organization (OTE). http://www.ote.gr [22 February 2010].

[34] EADS Defence & Security Systems (DS). http://www.eads.com [22 February 2010].

[35] RENFE-Operadora. http://www.renfe.es [22 February 2010].