Automated Web Service Reconfiguration

Sander van Splunter
Doctoral committee:
prof.dr. F.M.T. Brazier (TU Delft, promotor)
dr. P.H.G. van Langen (TU Delft, copromotor)
prof.dr. F.A.H. van Harmelen (VU University Amsterdam)
prof.dr. M.N. Huhns (University of South Carolina)
dr. C.H.M. Nieuwenhuis (D-CIS Lab Delft)
prof.dr. O.F. Rana (Cardiff University)
prof.dr. C. Witteveen (TU Delft)

SIKS Dissertation Series No. 2010-14
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems, and supported by Stichting NLnet and the Interactive Collaborative Information Systems (ICIS) project [http://www.icis.decis.nl]. The ICIS project is supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024.

Copyright © 2010 Sander van Splunter
All rights reserved.
Automated Web Service Reconfiguration

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
on gezag van de rector magnificus
prof.dr. L.M. Bouter,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen
op maandag 29 maart 2010 om 15.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Sander van Splunter

egenomen te Alkmaar
promotor: prof.dr. F.M.T. Brazier

copromotor: dr. P.H.G. van Langen
Acknowledgements

This thesis could not have been written without the support and company of others. I thank them for enabling me to create this thesis, both from a professional and personal perspective.

This thesis has been written while I was part of the research group headed by Frances Brazier. Within this group research focussed on interactive intelligent distributed systems and autonomic self-management.

First of all, I would like to thank Frances Brazier, Pieter van Langen, and Niek Wijngaards for their active involvement in my research. Your support, attention, inspiration, belief, vision, and guidance enabled me to develop as a researcher. Your doors (and telephone-lines and email-boxes) were always open, and many cups of tea and coffee have been consumed.

Furthermore I would like to thank the reading committee, Frank van Harmelen, Michael Huhns, Kees Nieuwenhuis, Omer Rana, and Cees Witteveen, for reading and commenting on earlier versions of this thesis.

For the time I worked in the Artificial Intelligence department of the Vrije Universiteit I would like to specially thank Debbie Richards from Macquarie University in Sydney and Marta Sabou for our cooperation on pioneering in applying OWL-S on existing web services.

For the time I worked in the Computer Systems department of Andy Tanenbaum of the Vrije Universiteit, I have greatly enjoyed the active involvement of the ’tea at three’ every Tuesday. This initiative (thanks to Elth Ogston for setting it up) could only keep its momentum with all of those who kept on bringing in teas and cookies from all over the world.

For the time I worked at D-CIS lab in Delft, applying my research in a crisis management context, I would like to especially thank Bernard van Veenen, Niek Wijngaards, and Kees Nieuwenhuis for making the research fun, challenging, and fruitful. In addition I would like to thank the researchers at D-CIS lab for making it such an open, multi-disciplinary, and social environment for research.

Currently I am looking forward to applying the research presented in this thesis even further at the Faculty of Technology, Policies and Management.
of the TU Delft, and I would like to thank Alexander Verbraeck, Wim Veen, and members of their research groups for the open environment.

However, as mentioned, a constant factor has been my membership of the research group of Frances Brazier. I have really enjoyed working in this group. I would like to thank all who have been part of this group, and I would especially like to thank to David Mobach, Thomas Quillian, Michel Oey, Kassidy Clark, Elth Ogston, Reinier Timmer, Martijn Warnier, Reza Haydarlou, Evangelos Pournaras, Etienne Posthumus, and Mark van Assem for making working in this group both fun and interesting.

Last I would like to thank my friends and family. Specifically I would like to thank my father and mother for their support, you are both part of this work. Most of my thanks go to my wife, Irene van Splunter, without your support, belief, patience, and love I would not have been able to complete this part of research. You and our daughters Julia and Meike are the main inspiration and foundation of my life.

Sander van Splunter, 5th of February 2010, Hoorn.
Contents

Acknowledgements i

1 Introduction 1
  1.1 Research Hypothesis and Questions 1
  1.2 Scope and Assumptions 2
  1.3 Thesis Outline 3

2 Related Work 5
  2.1 Single Component Replacement 6
  2.2 Single-level Reconfiguration 7
  2.3 Multi-level Reconfiguration 10
  2.4 Reconfiguration as Configuration 12
  2.5 Discussion 13

3 Web Services, Templates & Configurations 15
  3.1 Web Services 15
  3.2 Templates 20
  3.3 Configurations 25
  3.4 Propagation 27

4 Automated Reconfiguration Process 31
  4.1 Overall Model of Reconfiguration 32
  4.2 Reconfiguration in More Detail 34
    4.2.1 Focus Determination 34
    4.2.2 Requirement Determination 35
    4.2.3 Template-based Configuration 35
    4.2.4 Integration 38
  4.3 Prototype Implementation 38
## Contents

**J Trace of Multi-Level Reconfiguration Example**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>J.1 Focus Determination (1st)</td>
<td>200</td>
</tr>
<tr>
<td>J.2 Requirement Determination (1st)</td>
<td>200</td>
</tr>
<tr>
<td>J.3 Template-based Configuration (1st)</td>
<td>200</td>
</tr>
<tr>
<td>J.4 Focus Determination (2nd)</td>
<td>207</td>
</tr>
<tr>
<td>J.5 Requirement Determination (2nd)</td>
<td>207</td>
</tr>
<tr>
<td>J.6 Template-based Configuration (2nd)</td>
<td>208</td>
</tr>
<tr>
<td>J.7 Integration</td>
<td>231</td>
</tr>
</tbody>
</table>

**Samenvatting**

233

**SIKS Dissertation Series**

235
Chapter 1

Introduction

Most of today’s systems are complex, distributed and networked in dynamic environments. Change is characteristic, failure a symptom often encountered. Managing such systems is a challenge for both human managers and autonomic managers. Designing systems capable of managing themselves is the goal of autonomic computing [74]: self-configuration, self-optimisation, self-healing, and self-protection. This thesis focuses on self-reconfiguration, one of the essential elements of self-healing.

This thesis introduces a new multi-level approach to automated reconfiguration of complex web services, due to failure of one or more web services. The approach consists of a generic model for complex web services and a generic process for automated reconfiguration. This approach is more powerful than any of the existing approaches as it maintains both functionality and Quality of Service at all levels within a configuration. Reasoning about dependencies between services and Quality of Service at each level, with local knowledge specific to the level, determines (1) the focus of adaptation and (2) the impact at the global level of Quality of Service. This chapter lists the research questions related to this new approach and outlines the remainder of this thesis.

1.1 Research Hypothesis and Questions

Our conjecture is that the current approaches to automated adaptation due to failure are limited in scope. Most of these approaches focus on component replacement for the failing service, often limited by the assumption that an exact match for this service is available. As web service compositions become more complex and further distributed, this assumption becomes less feasible. A more flexible approach, using knowledge to assess local changes,
and allowing a gradual increase of the part of the configuration to be replaced would offer advanced reconfiguration and enables more possible adaptations.

A local change to the structure of a configuration requires knowledge of the function of each component within the structure in the context of the whole configuration (i.e., with respect to overall requirements and dependencies, both functional and non-functional, such as Quality of Service requirements). In this thesis, this type of knowledge is called local knowledge.

This thesis has as its hypothesis that automated adaptation of complex services is feasible using local knowledge. Local knowledge allows changes to be local, reducing structural change to a minimum. The power of this approach is that it supports replacement of a single component (the one that failed) to reconfiguration of the complete configuration if necessary.

To create a system able to autonomously perform reconfiguration, the following research questions need to be answered:

Q1 How can local knowledge be represented and included within a configuration?

Q2 What process is needed for automated re-configuration using local knowledge?

1.2 Scope and Assumptions

The scope of this thesis is to address the following situation: given is (1) a web service configuration of semantic annotated components, and one of the services of the configuration has been identified as failing, and (2) instance replacement of the failing component is not possible.

Is it possible to define an automated generic reconfiguration process, that can reconfigure the given configuration such that it is functionally equivalent, and respects global QoS requirements. Enactment, monitoring and identification of the failing service are outside the scope of this thesis.

The first assumption for this thesis is systems are designed to be reconfigured. Secondly, a configuration that is reconfigured must be annotated and include local knowledge. Thirdly, that a repository of with components is available to the reconfiguration process, and that these components are annotated, including local knowledge.
1.3 Thesis Outline

Figure 1.1 shows a schematic overview of the chapters in this thesis, and their relations. This chapter and Chapter 2 on related work introduce the domain and the background of the research. Chapter 3 describes the concepts for template-based web service configurations. This chapter also introduces a template-based structure to contain local knowledge. Chapter 4 presents a generic model for the reconfiguration process, and an implementation in a prototype. Chapters 5 and 6 apply the framework to illustrate the approach, presented in two use cases. The applications in these two chapters are based on existing sets of services, and use the implemented prototype for reconfiguration. The purpose of these experiments is to evaluate the genericness of our approach. Chapter 7 presents the conclusions and a discussion on future work.
Chapter 2

Related Work

This thesis has as its hypothesis that automated structure adaptation is feasible by the inclusion of local knowledge. The presence of local knowledge allows changes to be performed locally, reducing the structural change to a minimum. In related research, as described in this chapter, different approaches exist on adaptation of complex configurations. According to these approaches, adaptation can be based on different policies. Basically four types of complex reconfiguration are distinguished, of which each differs in the preservation of the initial configuration, the availability of knowledge, and the locality of the proposed adaptation:

- Single component replacement
- Single-level reconfiguration
- Multi-level reconfiguration
- Reconfiguration as configuration

In Single component replacement the reconfiguration is based solely on replacing the failing component with an exact match. An exact match can be defined on 1) an instance level, where a match is determined by instance equality; 2) a syntactic level, where a match is determined by interface equality; as well as 3) a semantic level, where a match is determined by reasoning in a logic framework. Single component replacement assumes that an exact match is always possible. The initial configuration is completely preserved except for the failing component. A match based on the specifications of the failing component is assumed to return a component that is functionally equivalent to the failing component. After integration of the new component, the resulting configuration is assumed to be functionally equivalent to the initial configuration.
In *Single-level reconfiguration*, reconfiguration is based on the description of the overall configuration. A configuration is assumed to be constructed of one or more components, i.e., services. It is possible to resolve a failure of a single component by the replacement of multiple components. This approach has a number of advantages over single component replacement, as dependencies between components are identified, specific for this configuration, as well as relations between properties of the individual components with properties of the overall configuration. Reasoning on locality is limited, as reasoning about the full internal structure of the initial configuration is not used within this form of reconfiguration.

In *Multi-level reconfiguration*, the reconfiguration process assumes the initial configuration to have an explicitly represented internal structure, which can be used within the reconfiguration process. For automation, this approach requires the internal structure to be annotated, and knowledge must be available on how properties of a component relate to the properties of a larger part of the structure. The concept of locality is fine-grained and adaptable in this approach, as the full internal structure is known. Therefore, the scope of the reconfiguration is adaptable.

In *Reconfiguration as configuration*, the reconfiguration process is solely based on the requirements of the initial configuration. In the main form, reconfiguration from scratch, the initial configuration is discarded, as the reconfiguration starts by the construction of a completely new configuration. This approach requires knowledge on how a configuration relates to a set of requirements, and knowledge on how components can be combined to form more complex configurations. Locality is completely disregarded in this approach, as the initial configuration is ignored.

This chapter is organised as follows. Section 2.1 discusses single component replacement approaches. Section 2.2 discusses approaches that enable single-level reconfiguration. Section 2.3 discusses research for multi-level reconfiguration. Section 2.4 discusses reconfiguration as configuration, and Section 2.5 offers a small discussion.

### 2.1 Single Component Replacement

In single component replacement, adaptation is based solely on direct replacement of the failing component. If no direct match can be found as a replacement, this approach fails.

Single component replacement assumes that if two components are deemed equal, then they are exchangeable within any web service configuration, without negatively affecting the function or behaviour of the overall configuration.
2.2 Single-level Reconfiguration

The search for an exact replacement is performed based on the description of the failing component. There are different ways of describing components: instance-based \[76\] \[17\] \[81\] \[51\] \[130\]; syntax descriptions, e.g., WDSL \[39\]; or semantic descriptions e.g., OWL-S \[90\], or \[12\] \[162\]. These variations support an increased abstraction of the failing component, allowing more components to be considered as a substitute for the failing service, and thus enabling an increased flexibility for reconfiguration. The reconfiguration process for single component replacement can be minimised mainly to matching and to deploying the new configuration.

A number of approaches introduce separate concepts to identify a replaceable component and its description (or a preselected list of allowed replacements), e.g., virtual web services \[153\], and abstract web services \[136\]. A number of approaches define concepts to allow cooperation and information hiding, e.g., abstract web services \[5\] (BPEL), container services \[14\], or reference processes \[124\]. Though these descriptions are not complete, they do offer a description or placeholder, and assume that the services inserted at these places are exchangeable.

The main limitation of these single component replacement approaches is that only direct substitutions for the failing component are considered. Although syntactic and semantic descriptions of a failed service allow more flexibility, they only describe the properties of that specific service, and not the desired properties for which this service is needed within a specific configuration. The descriptions have no knowledge of the specific relevancy of properties of a service for a specific configuration. As a result, solutions using services with slightly different descriptions are not discovered, even though these would allow satisfactory solutions.

Its limitation also induces the strength of these single component replacement approaches: they are generic, as a configuration of web services often needs no further preparation for reconfiguration than to having the descriptions of the web services available. Automation of these approaches is highly feasible. If multiple services fail, then the policy can be applied repeatedly, as newly introduced dependencies or altered dependencies caused by replacing components are assumed not to exist.

2.2 Single-level Reconfiguration

Single-level reconfiguration is a more knowledgeable approach, considering a failure of a component in the context of the overall configuration. The basic principle is that each configuration can be prepared for reconfiguration, by the
Related Work

addition of knowledge specifically tailored to the configuration. This can be knowledge on:

– desired properties of components,

– dependencies between components,

– relation of properties of an individual service to the properties of the overall configuration,

– listing of possible fixes, optionally related to a specific problem diagnosis.

The complexity of the reconfiguration process can vary. If the annotations contain desired properties, then reconfiguration will mainly be a matching process (very similar to component replacement, the difference being that the description of the desired component is retrieved not from the failing component, but from the description of the configuration as a whole). If dependencies between components are also annotated, then a mechanism needs to be included to check whether the dependencies are satisfied. The same holds if knowledge is available on the properties of individual components in relation to properties of the overall configuration. Most of the presented approaches for single level reconfiguration, however, focus on reconfiguration to achieve a change in functionality or behaviour, and not on reconfiguration to deal with failure of a service. Two types of single-level reconfiguration are distinguished to this purpose: single-level fixes, and single-level schemas.

Single-level fixes. In single-level fixes, a definition of a single-level configuration is given, including one or more fixes. A fix specifically defines a way to deal with a need to change functionality or behaviour, or to deal with a given failure (e.g., failure of a service, failure to meet predefined QoS specifications). Fixes are defined independently of each other, to deal with unique conditions.

For example, variability points [142] are used to structure adaptations of existing configurations of services defined by patterns. Multiple fixes can be related to a single variability point. These variability points, and fixes are defined parallel to the design of a service. The use of variation points is similar to the self-adaptation achieved in DySao [127] and in the work described in [84, 96, 162]. Note that approaches using variability points often only focus on changes of a complex service as a whole, and not on specific characteristics of a service such as internal changes of pre-conditions and post-conditions. Comparably MobiPads [35] is an approach in which
possible adaptations are defined beforehand. Another approach specifically
targeted to handling fixes for failing instances of services is presented in [143],
tightly coupled to a monitoring and deployment architecture.

A special case of single-level fixes is when knowledge to reason on inter-
dependencies between the fixes is also included. For example, \textit{Parameterised}
\textit{skeletons} [139] have been developed in the context of Problem Solving Meth-
ods within IBROW and ported to OWL-S based web services [141]. This
approach is broker-centric, in which interdependency relationships between
both functional and non-functional aspects of individual and complex services
are pre-defined, in addition to explicit knowledge of their pre-conditions and
post-conditions.

\textbf{Single-level schemas.} In \textit{single-level schemas}, generic service composi-
tions are defined, using descriptions of abstract services. Each description
contains requirements, used to find (only) a service to instantiate the abstract
service. Dependencies between requirements of different abstract services can
be modelled, as well as dependencies between the abstract services and the
single-level configuration.

For example, \textit{Eflow} [33] uses process schemas in [34]. Process schemas de-
fine an abstraction over complex service instances, enabling single-component
replacement. In addition process schemas allow modelling of additional con-
straints representing dependencies between services.

In \textit{METEOR-S}\footnote{METEOR-S project page: \url{http://lsdis.cs.uga.edu/projects/METEOR-S/}} process templates [129, 101] are workflow-based. The
METEOR-S project also investigates automated web service composition,
with a focus on Quality of Service (QoS). Semantic process templates de-
fine configurations of web services with constraints for each individual web
service related to a single level of configuration [31]: properties of the pro-
file of a service (name, textual description, QoS metrics, input and output
parameters). The need for composable structures is identified but has, as
yet, not been addressed: without this the approach is limited to single-level
reconfiguration.

In \textit{Gronmo et al.} [57] the services in a configuration are linked to a con-
ceptual task description, to support replacement of individual services. A
conceptual task description of a configuration describes the (QoS) require-
ments and constraints of the services used. Mechanisms described in [66]
reason about QoS properties in the context of configurations as a whole.
Single level reconfiguration supports automated replacement by a single ser-
vice, replacement by a composite service depends on human evaluation.

In \textit{Chen et al.} [38] a generic schema, a multi-tier service, is defined for
single level configurations with a sequential activation of multiple abstract services. The generic schema in this approach is defined to contain knowledge on the QoS relations, both between the abstract services as between the abstract services and the overall configuration.

In conclusion, the main strength of the above single-level reconfiguration approaches is that they support reasoning on the effects of a proposed adaptation to the function and behaviour of an overall configuration. The other strong point of these approaches is that the reconfiguration process is not limited to replacement of the failing service only. More complex adaptations are enabled, such as the replacement of multiple components.

The concept of locality in single-level reconfiguration is limited, as the scope of reconfiguration is fixed in the configuration description, and often reconfiguration is performed by changing a limited set of components only. In comparison to single component replacement, this single-level reconfiguration offers more flexibility, as more complex adaptations are possible. As there is no variation in scope of the reconfiguration, however, the possibilities are specifically bound to the structure of the initial configuration. Automated adaptation is possible, but requires either extensive annotation and reasoning capabilities, or the adaptation to be limited and restricted to specific fixes. Both require extensive analysis and preparation.

2.3 Multi-level Reconfiguration

Multi-level reconfiguration is an advanced approach, that utilises the compositional structure of complex service configurations, and allows adaptation at each level: each complex service within a complex service defines a level. Per level the following knowledge can be defined:

- desired properties of components,
- dependencies between components,
- relation of properties of an individual service to the properties of the complex service as a whole,
- listing of possible fixes (in parameterised approaches), optionally related to a specific problem diagnosis.

This knowledge is similar to the knowledge defined in single-level configuration, except that in multi-level reconfiguration it is defined at a finer granularity. The current approaches for multi-level reconfiguration focus mainly
on extending the flexibility of the single-level composition approaches. Reasoning about the multi-level structure itself is not included.

Different approaches are based on different compositional structures. Hierarchical Task Networks (HTN) allow recursive decomposition of tasks, and specification of constraints and dependencies between the tasks. Constraints also include the activation order, which in the case of web services would be captured in coordination patterns. Dependencies between tasks can be expressed as relations between pre-conditions and post-conditions. Examples of HTN planners are SHOP2 [102], and UMCP [46]. Examples of usage of HTN for automated Web service configuration are [128, 78]. HTN planning based on SHOP2 in [128] defines a compositional template definition for OWL-S. These templates contain requirements for abstract processes, and allow quality-able requirements for Quality of Service attributes, enabling a partial ordering of requirements. Planning is extended to encompass qualifiable user preferences in [85]. The Dynamic Composer system in [63] allows a level of QoS awareness, however the same heuristic for selection is used for every individual component, irrespective of its effect on the global QoS. OWLS-XPLAN [78] has an extension for redesign [79]. Alternatively a conditional constraint network can be used, as in [4] in which templates are defined through which uniform constraints can be propagated, to allow the proposal of composed incremental solutions. All mentioned planning approaches, however, do provide no to very limited support for reasoning about the effect of local Quality of Service attributes on the Quality of Service attributes of the overall configuration.

In [126] a workflow-based approach is presented. This approach defines workflow schemas or workflow scripts as structures, in terms of tasks and temporal dependencies between tasks. It allows compound tasks, that is tasks composed of other tasks. After deployment the schemas are preserved, to allow modifications. Another feature of this approach is the distributed deployment of the complex service, and distributed adaptations. This work evolved into the ADAPT² framework [10], in which compound tasks reoccur [159], allowing a multi-level composition. However reasoning over multiple levels is not supported.

In [54] an organisational view is offered based on collaboration management. Tasks are defined for enabling dynamic runtime execution, in which tasks can be decomposed, based on task escalation points. The focus is on on-the-fly management of the coordination, and on semi-automated support, and the approach does not cover reasoning about the relation between properties of a single task and properties of the overall task.

²ADAPT project page: [http://adapt.ls.fi.upm.es/adapt.htm](http://adapt.ls.fi.upm.es/adapt.htm)
ReFFlow\cite{70} uses templates for automated creation of complex workflows\cite{69}. Adaptation varies from instance replacement, parameterised adaptation (skeletons), to dynamic structure-based redesign. This approach is one of the more advanced approaches that actively targets reconfiguration, and that offers a wide choice of ways to handle reconfiguration. However, reasoning on the effect of the properties of a single service on the overall configuration is not supported. This limits the possible changes available for reconfiguration.

In conclusion, the strength of these multi-level reconfiguration approaches is that these have the potential not to be limited to a predefined set of adaptations, as dependencies are modelled per level of composition, and not bound to a predefined set of components. The compositional structure is utilised both in determining which part to be replace, as in searching for a replacement component: a compositional configuration can be built to create a complex service useable as a replacement for the failed service. In addition, the part of the initial structure to be replaced by reconfiguration can be incrementally enlarged.

The weak point of these multi-level reconfiguration approaches, is that the preparation of the set of components requires a significant investment. It is difficult to express dependencies over multiple levels, as most approaches limit themselves to functional decompositions (e.g., Hierarchical Task Networks). More advanced dependencies, for example regarding Quality of Service properties, are not decomposed in most approaches. In addition, annotations for defining multi-level configurations are still limited (for example OWL-S does allow the specification of complex services, but is limited in specifying dependencies between components in a specific configuration). Lastly, automated reconfiguration requires a large set of components which can be used. This is still limited in most approaches.

\subsection{Reconfiguration as Configuration}

When reconfiguration is approached as configuration, a new configuration is created, instead of adapting the initial configuration. Configuration\cite{97,156} itself can be considered a problem close to design\cite{26}, and is often restricted to very particular domains.

In the literature, configuration is approached from different perspectives\cite{134,59}. Examples are configuration as constraint satisfaction\cite{8,3}, genetic

\footnote{ReFFlow project page: http://www.dvs1.informatik.tu-darmstadt.de/research/refflow/index.html}
algorithms (such as A-Design [28]), problem solving methods (such as Pro-
pose and Revise in CommonKads [123]), generic models for design (such as
GDM [145]), Case-based reasoning (such as WorkBrain [154]), and planning
(for example [93, 110, 115, 95, 94]).

To extend configuration approaches to reconfiguration as configuration,
the requirements of the initial configuration need to be available. This is, in
many cases, an issue as the only information available is mostly the structure
of the initial configuration.

Ideally, reconfiguration as configuration is more than selection of a lim-
ited set of pre-selected fixes. Configuration allows the creation of a new
configuration that fits the requirements best. This, however, requires con-
siderable knowledge, often solely available as human experience. Whereas
semantic annotation provides knowledge about the functionality, behaviour
and structure of a single service, it is often limited in describing how and why
a particular combination of services results in the derived complex function-
ality, behaviour and structure. This limits the possibilities for automation of
a configuration process considerably.

2.5 Discussion

The approaches discussed in this chapter all have their strengths and weak-
nesses. Component based replacement is an option that remains closest to
the initial configuration, but depends heavily on the availability of an exact
match. Single-level reconfiguration supports more advanced adaptations, but
this requires each configuration to be specifically annotated. Multi-level con-
figuration allows more variation in adapting the part afflicted by the reconfig-
uration, but requires availability of more compositional annotated structures.
Configuration as reconfiguration requires explicit requirements for reconfig-
uration.

Ideally an hybrid approach is constructed, which can vary the scope of
the adaptation, and which allows the most efficient approach for the selected
scope. Chapter 4 presents such a hybrid approach.
Chapter 3

Web Services, Templates & Configurations

This chapter describes the basic compositional units for enabling reconfiguration of a complex service consisting of web services. Section 3.1 provides an introduction to web services, with a description of the web service architecture, and a discussion of ontologies to describe web services and their interactions. Section 3.2 introduces the notion of template for web service reconfiguration, to capture information on partial service compositions. Section 3.3 presents template-based web service configurations. Lastly, Section 3.4 discusses propagation to allow reasoning on relations between properties over multiple levels of composition, such as e.g., Quality of Service properties, and how to represent such local knowledge.

3.1 Web Services

A web service is self-contained module - deployed over standard middleware platforms - that can be described, published, located, orchestrated and programmed using XML-based technologies over a network [109]. Figure 3.1 depicts a web service architecture, based on [133]. From bottom to top, the architecture reads as follows.

The lowest layer, layer 1, contain standard protocols to transport messages, to access information from a web service. The standard Internet protocols available in this layer are also found in the top-layer (application-layer) of the OSI-model [43], e.g., HyperText Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), and File Transfer Protocol (FTP).

4Part of this chapter is an extended version of [116].
The second layer adds structure to the content of the messages, providing a uniform data exchange format. Protocols in this layer are XML-based, e.g., Simple Object Access Protocol (SOAP). SOAP allows the exchange of structured and typed information between web-based applications.

The middle layer, the third, has languages to describe the web services, e.g., their input and output, and functionality: Web Service Definition Language (WSDL), and OWL-S. WSDL describes mainly the interface of a service, expressing details such as the location and port numbers of the service, and specifying the types for the input and output. WSDL can be defined on top of SOAP. OWL-S extends the WSDL description with a semantic description of the service. Functionality defined in the higher layers, such as service discovery, service composition, and service reconfiguration, rely on information defined at this level. For this reason WSDL and OWL-S will be discussed in more detail below.

Specialised tools and languages, in layer 4a, such as Triana [89, 137], BPEL4WS [5, 68] and the reconfiguration process discussed in this thesis, are used to compose and adapt complex services. Service descriptions are published and retrieved, in layer 4b, using service repositories, such as UDDI [13] [9, 48] or alternative approaches such as [86] [122].

The fifth layer, the Applications layer, consists of web services (both complex and individual services), and applications built on top of these services.

![Figure 3.1: Web service architecture (adapted from [133]).](image)

As stated above, the service description languages WSDL, and OWL-S are the bearing for the functionality offered in layers 4 and 5.

**WSDL** is the Web Services Description Language [39] and describes the interface of a web service, at the syntactic level, to enable access to the ser-
3.1 Web Services

In an XML format WSDL describes web services as a set of endpoints with associated messages. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. A WSDL description of a web service can contain the following elements:

- **Types**: a container for data type definitions using some type system (such as the W3C proposed XML Schema Definition Language (XSD) [47]).
- **Message**: an abstract, typed definition of the data being communicated.
- **Operation**: an abstract description of an action supported by the service.
- **Port Type**: an abstract set of operations supported by one or more endpoints.
- **Binding**: a concrete protocol and data format specification for a particular port type.
- **Port**: a single endpoint defined as a combination of a binding and a network address.
- **Service**: a collection of related endpoints.

The elements most commonly used in web services are Message, Port, Port Type, Binding and Service. The Message part describes the data that is communicated, and can be used to link to the corresponding OWL-S parameters. The part that describes the Port and Port Types defines names for the input and output of the web service and creates a Port for the input or output. These are then linked by the Binding to the SOAP level to ensure communication between different web services. Finally the Service combines the Port Types and Bindings and links them to a SOAP location on the Internet.

**OWL-S** [90] is an ontology for annotating web services, based on the Web mark-up language OWL [92]. This thesis uses OWL-S version 1.0. As depicted in Figure 3.2, OWL-S describes a main service (Service) using three different sub-ontologies: the ServiceProfile, ServiceModel, and Service-Grounding ontology. The Service Ontology is used to specify the highest level OWL-S concept: a Service. Further it specifies the relation of the Service class to other classes (see Figure 3.2). Each of these describes the service from
a certain point of view using concepts defined in the corresponding OWL-S ontology. The Profile describes what the service does, the Model describes how the service works, and the Grounding maps the processes defined in the previous ontology onto the actual calls and messages, defined in WSDL specifications (version 1.1).

![Diagram of OWL-S ontology]

Figure 3.2: The OWL-S ontology

A ServiceProfile describes the capabilities of the service for discovery purposes while a ServiceModel details its internal workings. A ServiceGrounding links the abstract description of the service to actual implementation details, such as message exchange formats and network protocols so that automatic invocation of the service is possible.

**Partial Configuration Descriptions**

This section discusses the need to describe partial configurations of services for reconfiguration. This is not inherently supported by OWL-S. The creation of partial configuration descriptions, and making these descriptions available in repositories, allows more complex reconfiguration.

In web service configuration, services are combined to satisfy a set of requirements. After configuration the end result is deployed: an OWL-S description of a complex service. During configuration the complex service is annotated (a profile is created), and the composition is defined in its process model and grounding. These annotations are interpretable by an automated process, as the annotations use OWL-S and domain ontologies. These annotations support reasoning about the current functionality and composition of web services. These descriptions do not capture the requirements for which the complex service has been created, the reasons specific services have been chosen to be part of the complex service and to be connected as such, etc. In other words, a deployed configuration does not explicitly define statements on the knowledge used during the creation of the configuration: the design
rationale [27]. Figure 3.3 illustrates different aspects of design rationale for web service configurations.

When relating partial configurations to the web service architecture presented above, only the service description layer is available, and OWL-S offers limited support to describe partial configurations. OWL-S is designed to describe web services, either atomic (not composed), simple (a grey-box description of the internal process), or complex (composed of multiple sub-processes). Web services in OWL-S are always complete. OWL-S has no concepts to support abstract processes to describe a partial configuration.

OWL-S offers support to define pre-conditions and post-conditions for a service, but not to express dependencies within a partial configuration. OWL-S does not support specification of pre-conditions and post-conditions within the context of a specific application, nor relations between properties of a complex service and properties of the services of which it is composed.

Specifications of partial service configurations to define abstract service configurations, and annotated interdependencies between abstract services related to the contexts of the abstract configuration, would make automated multi-level reconfiguration feasible. To this end, templates are introduced in Section 3.2.
3.2 Templates

To support automated reconfiguration, the web service architecture is extended with an additional layer (3b) for the description of partial service configurations, as illustrated in Figure 3.4. This section describes templates to represent local knowledge within web service configurations.

![Diagram of Web Service Architecture]

Figure 3.4: Web service architecture (adapted from [133]), extended with a 'partial service configuration description' layer (3b).

There is a multitude of approaches to define structures of partial configurations, as shown in Chapter 2, but most of these structures are specific to a particular composition method and do not support reasoning over multiple composition levels. This section describes our open approach, which is independent of the composition approach. It defines hierarchical compositional structures, including partial configurations for the domain of web services.

Common reconfiguration approaches base their reconfiguration efforts on the information available on the failed service by finding another service instance, matching on syntax, or matching on the semantic description. Instead of reconfiguration centred around the description of the failed service, it is also possible to start reconfiguration based on the remaining (still correctly working) part of the configuration: to find a service that enables the remainder of the configuration to offer the same functionality. The requirements for the replacement result from dependencies between services present
in the remainder of the configuration: they form the context for the substitute service. The context needs to be structured: if a composition is large, so is the configuration space, and dependencies are more difficult to express. Local knowledge addresses this challenge using the hierarchical structure of a service composition.

Local knowledge is knowledge within a complex service (composed of several sub-processes) about its sub-processes and present dependencies, specified at one composition level. In other words, local knowledge includes: knowledge about the immediate structure of the process and knowledge about the process’ immediate constituents and their dependencies.

Templates are used to express local knowledge, structure complex services, provide a means to represent configurations of services and knowledge about the configuration. Consequently, templates provide a means to represent local knowledge for web services. Both templates and web services are used as compositional structures: they can be combined to form a template-based web service configuration. Both templates and web services have properties, which are annotated. To regulate the creation of a configuration, templates have slots. Slots are associated with requirements that can be matched with the properties of web services and templates.

The remainder of this section explains the basic concepts for template-based configuration: template, slot, and control structure. These concepts are also defined in OWL, to support the use of OWL-S descriptions for web services and for templates.

A **template** describes a single-level composition of web services. This is defined as a control structure with one or more slots, and an associated template description, and dependencies between slots.

A **slot** is a placeholder. A slot defines the requirements for a desired (single) service or template. A requirement is an explicit expression about properties representing characteristics about function, behaviour or structure.

A **control structure** defines the conditions for activation of the slots. These are specified using control constructs over a set of slots. Examples of control constructs are sequential/parallel activation, and conditional constructs (if-then-else).

The template description specifies the properties of the template and the associated values, describing the function and behaviour. The template can also contain additional dependencies.
Describing templates in OWL-S

OWL-S is used to specify web service descriptions. A simple extension of OWL-S is used to specify templates, as illustrated in Figure 3.5 (see also [116, 120]). In OWL-S a Service Profile defines what the service does, as a higher level description of, for example, service type, input, output, pre-conditions and post-conditions. In our approach an OWL-S Service Profile is used to define a template, describing what this combination of slots would do, if used and refined. In OWL-S a Service Model offers insight in the internal workings of a service. A Composite Process in a Service Model defines a process composition of subprocesses to which references are defined, together with the control flow specified with pre-defined control constructs. In our approach for a template the OWL-S Service Model is used to define a CompositeProcess with slots defining abstract subprocesses. Slots are defined as a subclass of Process. Requirements defined on slots are included in this specification. Note that as the slots define abstract processes, a template description does not have a Grounding Model.

The remainder of this section describes the Template Profile Model, the Template Process Model, and the Slot Profile Model. Two examples of two complete template specifications in OWL-S are included in Appendices A and B.

Defining what a template does: Template Profile Model

A template profile model, which is similar to an OWL-S profileModel, contains information about the service publisher, the name of the service, a textual description, the inputs and outputs, and a representation of the capabilities of the service. The capabilities can be represented using (1) pre-
3.2 Templates

Figure 3.5: Example of a template description expressed in OWL-S.

conditions and post-conditions, or (2) a categorisation of the service. The
profile can be extended to represent Quality of Service attributes, of which
the values are dependent on the services and templates used to fill the slots
of this template.

```xml
<service:Service rdf:ID="Template1Service">
<!-- Reference to the Profile -->
<service:profers df:resource="#Template1Profile"/>
<!-- Reference to the Process Model -->
<service:describedBy df:resource="#Template1Process"/>
</service:Service>
<ontology:DisplayCreator rdf:ID="Template1Profile">
<profile:serviceName>
  ti_Determine_area_triangle_unknown_height
</profile:serviceName>
<profile:textDescription>
  This template is defined for determining the area of a triangle with one square corner.
  The length of all sides is known, except the height of the triangle.
  The template contains two slots:
  1. for determining the height of the triangle, based on the length of the other two sides.
  2. for determining the area of the triangle, based on the length of the height and the base.
</profile:textDescription>
<template:hasSlot df:resource="#DetermineHeight"/>
<template:hasSlot df:resource="#DetermineArea"/>
```

Defining how a template works: Template Process Model

The template process model is based on the OWL-S service model for a composite process. It is similarly defined as a composite process, but instead of being composed of processes, it is composed of slots. The following example
defines the process model of a composite process $P_1$. Process $P_1$ requires
the sequential activation of two slots DetermineHeight and DetermineArea.
The slots are specified in the next paragraph. The specification of inputs
and outputs of the process is omitted in this example.

---

// example template in OWL-S of processModel
<process:ProcessModel rdf:ID="template1Process"/>
<process:hasProcess rdf:resource="#P1"/>
</process:ProcessModel>

<!-- Definition of top level Process as a composed process -->
<owl:Class rdf:ID="P1">  
<owl:intersectionOf rdf:parseType="Collection">  
<owl:Restriction>  
<owl:onProperty rdf:resource="#composedOf"/>
<owl:Class rdf:about="#CompositeProcess"/>
</owl:Restriction>
</owl:Intersection>
<owl:toClass/>
</owl:Class>

---

Defining an abstract process: Slot Profile Model

Slots are placeholders specifying the requirements for a template or web service. Requirements can define a desired functionality, structure or behaviour, for example, by specifying ranges for a property. The values are related to annotations in profiles of templates and web services, expressed as constraints for properties defined in their OWL-S ServiceProfiles.

The slot specification for DetermineHeightSlot, depicted below, is part of
the template specified above. It specifies the requirement (rq1) that the template or web service needs to be of a specific serviceCategory TriangleCalculations: DetermineHeight, and the requirement (rq2) that the resulting precision of the template is at least 8. The second requirement is for a domain specific property for services for mathematic operations.

---
3.3 Configurations

The previous sections of this chapter described web services and templates. This section describes how templates and web services are combined. The structure of the resulting configuration is defined below, followed by the properties associated with (partial) configurations.

A configuration is composed of a set of web services, a set of templates, and a set of relations between web services and templates.

A configuration defines all relations between slots and services or templates, recursively. Each recursion defines either a leaf of the configuration (a single web service), or a branch (a single template, with reference to the configurations that have been inserted into its slots). The properties of a configuration are equal to the properties of its root (a web service or template).
The template-based structure of a configuration allows completeness and degrees of correctness within a template-based configuration to be defined. These concepts can be used in a configuration process to determine whether a satisfying (intermediary) result is found. For each component that is part of a configuration, both the completeness and correctness can be assessed.

The completeness of component is assessed by examining whether the slots within this component have been filled by components. A component is strongly complete if and only if none of the slots that are directly part, and recursively part of the component is empty. Thus, (1) a web service is strongly complete by default (it has no slots); (2) A template is strongly complete if and only if each of its slots is filled with a strongly complete component; (3) A configuration is strongly complete if and only if its top component is strongly complete. In other words, a strongly complete component does recursively not contain any open slot. In Figure 3.6 an example configuration is presented in which all strongly complete components are highlighted, and all not strongly complete components are faded out. The configuration as a
whole is not considered strongly complete as its top component, template t1, is also not strongly complete.

The correctness of a configuration is assessed by examining whether the requirements for the templates are fully satisfied. This implies that the requirements for each of the slots have been fully satisfied by the components with which they are filled. Note that the term ‘slot-filling’ of a template refers to the combination of a slot of the template and the component which has been inserted into that slot. Three notions of correctness are defined: correct, incorrect and partially correct.

A component is correct if all of its direct slots-fillings are correct. A slot-filling is correct if and only if the component inserted in the slot satisfies all requirements of that slot, and the inserted component is correct itself. Thus, (1) a web service is correct (as it has no slots); (2) a template is correct if and only if all of its slot-fillings have been filled with correct components; (3) A configuration is correct, if and only if its main component is correct.

A component is incorrect if and only if one or more of its direct slots-fillings is incorrect. A slot-filling is incorrect if and only if at least one requirement is violated by the inserted component, or if inserted component itself is incorrect. Thus, (1) a web service is never incorrect (as it has no slots); (2) a template is incorrect if and only if at least one of its slot-fillings violates at least one of the requirements of that slot, or when at least one of its slots is filled with an incorrect component; (3) A configuration is incorrect, if and only if its main component is incorrect.

A component is partially correct if and only if none of its direct slots-fillings are incorrect, and at least one of its slot-fillings is partially correct. A slot-filling is partial incorrect if and only if it is not correct and not incorrect, or when the slot-filling is empty. Thus, (1) a web service is never partially correct (as it has no slots); (2) a template is partially correct if and only if none of its slot-fillings are incorrect, and at least one of its slot-fillings is either empty, not correct, or partially correct; (3) A configuration is partially correct, if and only if its main component is partially correct.

The concepts on completeness and correctness, as defined in this section, are useful for the automated construction of template-based configurations, and for template-based reconfiguration, which is discussed in Chapter 4.

### 3.4 Propagation

To determine the implications of changing one or more components in a template-based configuration, reasoning about the effect of a local adaptation with respect to the overall configuration is required. For some properties,
e.g., Quality of Service properties, local adaptation has implications for multiple levels in a configuration. Templates include requirements on properties such as QoS properties. Handling requirements on such properties in multi-level configurations requires additional local knowledge on the dependencies between the properties of components in a template. Note that propagation requires a configuration to be strongly complete and partially correct.

The principle of propagation relies on the principle of compositionality that specifies that the properties of a complex system are determined by its structure and the properties of its constituents [135]. A strengthened and often presupposed notion, the principle of local compositionality, is that the properties of a complex system are determined by its immediate structure and the properties of its immediate constituents [135].

Local compositionality within a multi-level system must also take dependencies between different levels of the system into account. Propagation resolves such dependencies between the different levels of a system. Template-based configurations are hierarchical compositional structures, where the dependencies for reasoning on local compositionality follow the hierarchy. Propagation for template-based configurations resolves reasoning on local compositionality depth-first: starting at the templates containing only leaves, working upwards. Reasoning at each level using the local knowledge contained in the requirements results in knowledge on the impact of the configuration at that level.

![Diagram](image)

(a) First propagation step, determining the impact of the template with sequential activation.

(b) Second propagation step, determining the impact of the template with parallel activation.

Figure 3.7: Propagation of average execution time in two steps.
As an example consider a template-based configuration of two templates and three services, as displayed in Figure 3.7. The templates each have two slots, the first template has parallel activation of its slots, and the second template has sequential activation of its slots. The parallel template is the top component of the configuration, and the sequential templates is inserted in the second slot of the parallel template. The average execution time is known for each of the three services.

For the parallel template, the local knowledge contained in the requirements for local compositionality of the average execution time is to take the maximum of the average execution time of the components inserted in its slot. For the sequential template, the local knowledge contained in the requirements for local compositionality of the average execution time of the template is the sum of the average execution time of each of its slots.

Propagation is needed to determine whether the requirement on average execution time is satisfied. The first propagation step is to determine the impact of the sequential template using its requirements on the average execution time: the average execution time for the sequential template equals the sum of the average execution times of the configurations inserted into slot 1 and slot 2, resulting in 5ms, depicted in Figure 3.7(a). The second propagation step, dependent on the first, is to determine the impact of the parallel template on the average execution time using its requirements on the average execution time: the average execution time of the parallel template is the maximum of the average execution times of the configuration inserted into slot 1 and slot 2, resulting in 5 ms, depicted in Figure 3.7(b).

The OWL-S based template description has been extended to included
evaluations of mathematical expressions using an approach similar to \cite{80}. The example above defines local knowledge for local compositionality for the average execution time for the sequential template, \texttt{sequentialExecution-TimeFunction} expresses the sum of the average execution times of the slots.

Propagation using the OWL-S template-based description through a strongly complete configuration consists of: (1) determining the order of templates for reasoning on local compositionality (2) selecting a template for each propagation step to reason on local compositionality, (2a) determining the values of the properties of each component inserted into the slots of the template, (2b) deriving the impact of the template for the properties using the mathematical expressions.
Chapter 4

Automated Reconfiguration Process

This thesis focuses on template-based web service reconfiguration to cope with failure of one or more constituents of a complex web service. Figure 4.1 depicts the five main processes involved: Web service & template retrieval to discover and match requests in a repository; Template-based reconfiguration for web services to perform reconfiguration design; Assembly of web services to create a deployable (complex) web service configuration from a description created by the reconfiguration process; Execution to activate the deployable web service configuration; and Monitoring to observe system behaviour (i.e., failures). This thesis focuses on the reconfiguration process, using very basic implementations of the other processes of complex service adaptation.

For the purpose of clarifying, the reconfiguration process described in this chapter is a single machine process. Note that extending the process to a distributed process is possible and in itself is relatively straightforward.

Figure 4.1: Overview of processes for the adaptation of web service configurations.

This chapter explains the template-based reconfiguration for web services
process, introducing the overall model for reconfiguration, followed by more detailed explanations of subprocesses.

4.1 Overall Model of Reconfiguration

This section provides an overview of a generic template-based reconfiguration process. The subprocesses are described in more details in separate sections. Figure 4.2 depicts an overview of the reconfiguration process.

![Activity diagram of the Reconfiguration Process](image)

Figure 4.2: Activity diagram of the Reconfiguration Process

The reconfiguration process has three external states: started, failure, and success. In the started state, the reconfiguration process has been initialised. The following information is required at the start of the reconfiguration process:

- the initial service configuration,
- the failing web service,
- the available repositories for web services and templates.

In the end the reconfiguration process results either in the state success or in the state failure. The result of success indicates that a suitable service configuration has been found to replace the initial configuration, and this new web service configuration is returned by the reconfiguration process.
The result of failure indicates that no suitable service configuration could be found, using the available repositories. Both success and failure indicate the end of the reconfiguration process.

The following subprocesses are part of the reconfiguration process, and are displayed in Figure 4.2:

**Focus Determination** Determines the part of the initial configuration for which a replacement is to be created, this sets the scope of reconfiguration.

**Requirement Determination** Determines the requirements applicable to the part in focus.

**Template-based Configuration** Creates a new configuration that can replace the part in focus, satisfying the determined requirements.

**Integration** Removes the failing part of the configuration and replaces it with the created configuration, resulting in a new web service configuration.

In the following section these processes are described in more detail. Information is exchanged between the subprocesses of the reconfiguration process. The following information items are used by the reconfiguration process:

**Initial configuration** The template-based web service configuration to be reconfigured.

**Failing web service** A pointer to a service in the initial service configuration, that needs to be replaced.

**Repositories** A pointer to repositories containing web services and templates that can be used by the reconfiguration process.

**Focus** A pointer to a slot containing a service or a template in the initial service configuration, representing the part for which the reconfiguration process is currently determining a replacement. If the value of a focus is null, this implies that no focus could be determined.

**Requirement set** A collection of requirements, related to a specific focus.

**Created configuration** A template-based web service configuration that satisfies the determined requirement set, created by the configuration process. If the value of the created configuration is null, this implies that no web service configuration could be found that satisfies the given requirement set, based on the available repositories.
New configuration The resulting configuration that will act as a replacement for the initial service configuration, without the failing web service. This configuration is a combination of the initial configuration with the created configuration.

4.2 Reconfiguration in More Detail

In this section the subprocesses of reconfiguration are discussed in more detail. As the presented model is a generic model, and generic policies are described for the subcomponents. Combining the generic model and policies allows for a wide range of instantiations of reconfiguration processes, of which the implementation described in Section 4.3 is one example. Section 4.2.1 describes Focus Determination, Section 4.2.2 describes Requirement Determination, Section 4.2.3 describes Template-based Configuration, and Section 4.2.4 describes Integration.

4.2.1 Focus Determination

Focus Determination determines the scope of the reconfiguration process. The focus is a single part of the given initial service configuration to be replaced by a (newly configured) part. A focus always refers to a slot into which a single service or a template has been inserted within the initial configuration. After a focus is set and no replacement has been found, the focus needs to be adapted. When it is not possible to determine a new focus, an empty focus is returned.

Focus Determination uses the template-based structure, as the templates define the granularity of the possible focusses that can be selected. Note that the presented approach can be used in combination with other kinds of modelling approaches or techniques (e.g., a web service can be a complex service itself, but modelled without using templates). Our template-based approach handles a complex service without templates as a monolithic service. If a part of this service breaks, then the whole complex service is marked as failing. This service does not offer handles to perform changes using our reconfiguration approach on a finer scale.

Different generic policies are possible for (re-)determining the focus, for example: (1) Prioritise the locality of adaptation, by keeping the focus on the slot closest to the failing component; (2) Policies based on heuristics to estimate least complexity of the configuration process, for example prefer focus for slots with a high availability of replacement components, which needs interaction with both requirement determination and Template-Based
Configuration; (3) Based on history of previous adaptations, preferring a focus of a previous successful reconfiguration; (4) Based on domain knowledge specific for the configuration.

In case of multiple failing components, focus determination can handle them as separate reconfiguration problems, or can select a focus based on an estimate of the restrictiveness of the potential fillings of a slot. Alternatively, a focus encompassing multiple failing components could be selected.

### 4.2.2 Requirement Determination

Requirement Determination determines the set of requirements the part in focus has to satisfy. A created configuration is considered a valid replacement if it satisfies all these requirements. Requirement Determination requires as input the initial configuration, the failed web service, and the focus. Requirement Determination is only activated if the focus is not null. There are two aspects to the determination of requirements:

1. direct slot requirements,
2. an exclusion requirement.

**Direct slot requirements** are requirements directly derived from the slot description of the slot in focus. These are the requirements used to define the slot, as described in Section 3.2. The exclusion requirement is based on the failed web service, simply stating a structural requirement; that the failed service should not reoccur in the newly created configuration. Using this exclusion requirement, the reconfiguration process avoids reproducing exactly the same web service configuration, which includes the failed service.

Within the model for reconfiguration described in this chapter, requirement determination is a simple look-up process, which inspects the initial configuration and extracts the set of requirements, based on the focus.

### 4.2.3 Template-based Configuration

The subprocess Template-based Configuration tries to create a new configuration satisfying the given requirement set, based on the services and templates available in the given repositories. If a configuration is found, this is returned as a result; if a configuration is not found, then null is returned as a result, which will be interpreted as a failure of the configuration process.

For configuration different generic policies can be defined, for example:

1. Prefer simple solutions over complex: if a single service satisfies the set of requirements, then this service is preferred over a multi-level configuration;
(2) Optimise on a specific property, e.g., Quality of Service; (3) Prefer minimal time used for finding a configuration, accepting the first solution that can be found; (4) Prefer the use of components from the same repository or made by the same company.

In its current implementation Template-based Configuration has two main approaches to find a service configuration that satisfies the given set of requirements:

- direct matching,

- configuration construction.

The main policy for Template-based Configuration implemented for this thesis is to prefer simple solutions over more complex: if a single service satisfies the set of requirements, then this service is preferred over a complex configuration. Based on this policy, template-based configuration will always first use direct matching. Only when this fails is configuration construction initiated, constructing a template-based configuration that satisfies the given requirement set. If configuration construction also fails, then the template-based configuration process fails, as it concludes that it cannot find a service or configuration that satisfies the given requirement set, and the current available repositories.

**Direct matching** is a matching mechanism to search for a description of a web service that satisfies the requirements. A complete match must be made if direct matching is to succeed, in other words direct matching only returns a web service if the service satisfies all requirements given to the Template-based Configuration process. Direct matching is enabled by one or more repositories containing web service descriptions, and a process that compares the requirements to properties in the service descriptions. Within the model for reconfiguration described in this chapter, the exclusion requirements are handled by the repository, so that excluded components are not returned by the matching process. As matching is not the focus of this thesis, a simple matching process is assumed, focussing on simple properties. This could be extended with more reasoning capabilities, and it could also be replaced by matchers from other approaches to reconfiguration. This will be further discussed in Chapter 7.

**Configuration construction** requires a more extensive process capable of constructing a complex template-based web service, based on the given set of requirements. As the configuration process itself is not the focus of this thesis, only some highlights are described in the remainder of this section, specifically related to the template-based structure. Appendix C contains a more extended description of the model of the configuration process used.
in the implementation. Note this process can be exchanged with any other configuration approach, as long as it assumes a template-based configuration structure.

**Subprocesses for template-based configuration construction**  As template-based configuration is not the main focus of this thesis, only a few aspects specific to template-based configuration are highlighted: the selection of a top template, how slots are filled, how propagated is evaluated, and deciding when the template-based construction is finished.

**Selecting a top template**  The principle of configuration construction is simple, as it first searches for a component that directly satisfies each requirement, or has property propagation functions with which the requirements can be evaluated. If no template is found, then configuration construction fails. If multiple templates are found selection can be done based on:

- least number of slots,
- greatest number of requirements directly satisfied,
- at random.

After a template has been selected, its slots need to be filled. If the process that attempts to fill the slots fails, then apparently the template cannot be made correct and strongly complete, and is discarded. Subsequently, another top component is sought, with the exclusion of the discarded template(s).

**Filling slots**  After a top template has been found, the slots need to be filled to create a strongly complete configuration. Each slot is described by a set of requirements, and can be considered as a separate configuration problem. The requirements are determined, and first a direct match is attempted, if this fails, a top template is determined, and the slots of this template are filled using the same approach recursively.

**Propagation evaluation**  Requirements depending on propagation are evaluated after all slots of a configuration have been filled (i.e., the configuration is strongly complete). If an evaluation results in the violation of a requirement, then backtracking identifies the last slot that was filled, and the component is removed from the slot. The configuration process returns to filling slots for the (again) empty slot.
Terminating configuration construction  The configuration process ends if a configuration is created successfully, or no configuration could be determined given the requirement set, and the available components. A configuration is created successfully, if a resulting configuration is returned that is 1) strongly complete, and 2) correct 3) satisfies the given set of requirements.

4.2.4 Integration

The Integration subprocess integrates a newly created configuration with the initial configuration, at the point of the focus. Properties of the resulting configuration are derived.

4.3 Prototype Implementation

This section describes a prototype created for the implementation of the model of reconfiguration, as described in this chapter. The prototype is developed in Java. A number of generic policies have been implemented for the reconfiguration of web services. The reconfiguration process is initiated manually: automated initialisation, automated detection of failures by monitoring (e.g., as in [91]) is outside of the scope of this thesis.

The type of scenario for which the prototype has been developed has the following characteristics:

- a template-based configuration of services exists,
- one of the services within this configuration has been marked as failing,
- a repository of services and templates is available and accessible for reconfiguration.

Figure 4.3 depicts the reconfiguration process, as previously presented in Section 4.1. The following paragraphs describe the prototype implementation for each of the processes Focus Determination, Requirement Determination, Template-based Configuration, and Integration.

Focus Determination  Focus Determination supports a policy specified in the model in section 4.2.1 to minimise the structural impact of an adaptation. To determine the initial focus the single-level focus determination sub-policy is used, and if no appropriate configuration can be found for this focus, the multi-level focus determination sub-policy is used.
4.3 Prototype Implementation

In single-level focus determination the focus is set on the slot with the failing web service, to affect the least amount of change to the initial configuration. In multi-level focus determination, the new focus is set on the slot with the template of which one of its slots was previously in focus, after focussing on the slot containing the failing service. In this chapter focus determination prioritises local reconfiguration: single-level focus determination is preferred over multi-level focus determination.

A reconfiguration process always starts with single-level reconfiguration: if a focus has not already been determined (in effect null), the new focus is by default the slot with the failing web service. If this focus does not result in a successful reconfiguration, then Focus Determination is activated again, and the policy is switched from single-level to multi-level focus determination. In the subsequent activations the focus will each time be shifted up one level of the hierarchy of the service configuration.

Focus Determination will fail if the previous focus refers to a slot in the top template of the initial configuration, as then the possibilities for reconfiguration are exhausted.

In Figure 4.3 an initial configuration is displayed, with a failing web service ws5. The subsequent activations of Focus Determination results in the following sequence of focusses.

- **Initial situation**: Focus is empty.

- **First activation**: Single-level focus determination: Focus is on the slot

![Figure 4.3: Overview of the Reconfiguration Process](image-url)
containing web service ws5.

- Second activation: Multi-level focus determination: Focus is on the slot containing template t4.

- Third activation: Multi-level focus determination: Focus is on the slot containing template t2.

- Fourth activation: Multi-level focus determination: Failure, no focus is selected, as there is no slot containing template t1.

Based on the initial configuration and the failing service a focus is determined. Both the initial data and the chosen focus are stored in an internal history, so that in future situations another focus will be selected, given the same configuration and failing web service. The focus contains a reference to a slot within the configuration, and is passed on to Requirement Determination.

Requirement Determination  Requirement Determination is kept simple, and retrieves the requirements from the slot definitions in the template. The structure requirement, to exclude the failing web service, is based on the pointer to the failing web service. The requirement determination process does not do any further reasoning on the requirements, this is left to the template-based configuration process.

Template-based Configuration  Template-based configuration consists of a component repository with matching functionality, and a configuration process. These two subparts are discussed separately.
4.3 Prototype Implementation

The component repository The repository contains both web service and template descriptions. The repository has a simple matchmaking component to select specific web services and templates. The implementation is pragmatic, in which the component descriptions are stored locally in a localised annotation, and matchmaking is limited to a pre-specified set of attributes. The repository can be constrained to exclude specific components from matching. This mechanism is used to exclude services marked as failing services from all returned results. At the end of each reconfiguration task, the list of failing services is reset, and cleared to avoid interference with other activations of the reconfiguration process.

The configuration process This process tries to construct a configuration which satisfies the requirements resulting from Requirement Determination. As described in Chapter 2, a wide variety of configuration approaches exists, and a wide variety could be used. There are a number of preferences to determine which configuration process to use for the implementation: first, a generic approach is preferred, such that the implementation could be used for different domains. Secondly, the approach should be able to handle template-based structures. Third, the approach needs to be able to include specific policies, such as preferring a simple solution. Fourth, an approach is preferred that allows reasoning on the requirements, to allow transformations of requirements and fine-grained reasoning on correctness, when handling requirements on properties that need to be propagated.

The resulting choice is based on a generic model of design [145], which has been adapted for our implementation to a specialisation for template-based configuration, similar as the specialisation done in [157]. This generic model is knowledge-based, allows fine-grained control [23], and distinguishes between reasoning about adapting a template-based configuration, adapting the requirements, and reflecting and managing the reconfiguration process itself.

Integration The prototype integration creates a new configuration based on the initial configuration, the focus and the new configuration. The support for annotation is simple, as references to propagation functions are included in the implementation of the structure of the template. The annotation subprocess uses the propagation functions for each slot to determine the specific values for the propagated properties. The resulting annotated configuration is returned by the integration process. The implementation of Integration is based on the following assumptions:

1. The initial configuration is already correct and strongly complete;
2. The created configuration satisfies the given set of requirements;

3. The requirements determined for the configuration process are sufficient to ensure that integration results in a correct configuration.
Chapter 5

Single-Level Adaptation: Reconfiguration using Classification Services

This chapter presents the first of two scenarios to illustrate template-based reconfiguration, namely single-level reconfiguration using templates. The domain of classification services is used within this example. Most reconfiguration approaches are similar to component replacement or single-level reconfiguration, as described in the related research in Chapter 2. These approaches usually base the adaptation on the description of the failed service, instead of focussing on the requirements that the remainder of the configuration poses to the service that failed, as in our approach. This scenario assumes that the failed service has been identified, and that instance-based replacement, which is part of component-based replacement, is not possible.

This chapter is structured as follows. Section 5.1 provides some background to the classification services used. Section 5.2 describes the structures available for reconfiguration, Section 5.3 describes the process of reconfiguration, followed by a discussion in Section 5.4.

5.1 Background

The scenario is focussed on reconfiguration of classification services. The classification services are based on a model described in [100], and on a set of services, initially developed [100] within the IBROW project [15]. The IBROW project studied the semi-automated reuse of libraries of components, based on Problem Solving Methods. The service for classification used
in this chapter are for classification of publications into categories based on their keywords. The keywords presented by the authors are used as input. A generic task model for this type of classification can be found in [140], and an instantiation of this model in [139] is used for classification of publications used in an IBROW context. In our implementation classification, classification entails:

1. Micromatch: determining the features of an object,
2. Aggregate: gather all features,
3. Admit: select the possible candidate classes,
4. Select: select the best class(es) out of the possible candidates.

Different IBROW services are used to implement parts of this functionality. To be used for template-based reconfiguration within our scenario the IBROW services have been reused as follows: (1) these services needed to be translated into web services with implementations and WSDL descriptions; (2) the services needed to be annotated with semantic information, i.e., OWL-S descriptions of the services, based on the explicitly defined domain knowledge in the IBROW component library [100]; (3) templates needed to be defined. Two different approaches to template creation are shown: construction based on an existing service, and construction based on a generic task composition. The initial configuration used is based on the newly created templates and web services.

The implementation of the reconfiguration process is not related to the IBROW project, and based on the model as described in Chapter 4.

5.2 Structures

This section describes the web services, templates, and configuration involved in this scenario. Section 5.2.1 describes the web services, Section 5.2.2 the templates. Section 5.2.3 describes the initial configuration, based on these web services and templates.

5.2.1 Web Services

The web services for classification (based on [100, 139]) used in the context of this chapter are:

1. EvaluatePublicationFeatures
2. BundleFeatureEvaluations

3. Four variations of Filter services:
   - FilterConsistentClasses
   - FilterKnownClasses
   - FilterPresentClasses
   - FilterStrictClasses

4. SelectHighestRatedClass

5. Intersect

These services are described in more detail below.

EvaluatePublicationFeatures This service is an instance of a MicroMatch service, which interprets performed Observations in Scored Observations on the basis of a set of class definitions. The class definitions are based on keywords for publications. This specific service scores all observations per class to feature statuses:

   - Inconsistent (I): The observed feature does not satisfy a feature condition of the class.
   - Unknown (U): The observed feature is not part of a feature condition of the class.
   - Missing (M): The feature appears in a feature condition of the class, but has not been observed.

BundleFeatureEvaluations This service is an instance of an Aggregate service, which determines an Aggregated Score per class over the Scored Observations. This specific service calculates per class the number of IUM features, and represents the resulting score in a triple.

Filter There are four variations of this service. These services are all instances of an Admit service, which checks whether an object has enough features to adhere to the set strictness of the class definitions. An Admit service has as input Aggregated scores and as output Candidate Classes.

   1. FilterConsistentClasses
   2. FilterKnownClasses
   3. FilterPresentClasses
4. FilterStrictClasses

The service \textit{FilterConsistentClasses} selects classes on a subset of features which must be consistent (not I). The service \textit{FilterKnownClasses} selects classes on a subset of features which are known (not U). The service \textit{FilterPresentClasses} selects classes on a subset of features which are present (not M). The service \textit{FilterStrictClasses} is a combination of the previous three services, where the selection is based on restrictions on Inconsistent, Unknown, and Missing.

\textbf{SelectHighestRatedClass} This service is an instance of a Select service, which selects one or more classes out of the \textit{Candidate Classes}. This specific service selects a single class out of the candidate classes based on the highest aggregated score.

\textbf{Intersect} This is a generic service, able to handle multiple inputs, and returning as output the intersection of the separate inputs. This instance handles different inputs of \textit{Candidate Classes}, and its output will only contain the classes that appear in each of its inputs. Note that this service is not based on the IBROW services described in [100].

5.2.2 Templates

Two templates are described: \textit{Strict Admission template} and \textit{Strict Publication Classification template}. The first template is based on an existing service and the second on a generic task model. A description of the process of the development of the templates has been added for interested readers. Note that the process of development is not relevant for understanding the trace described in Section 5.3.

\textbf{Strict Admission template} The first template is an alternative to an existing service: FilterStrictClasses. Figure 5.1 depicts this template, and distinguishes separate slots for filters: one for classes having only consistent features, one for classes having only known features, and one for classes having only present features. All three have as input \textit{Aggregated Scores}, and as output \textit{Consistent Candidate Classes}, \textit{Known Candidate Classes} and \textit{Present Candidate Classes}, respectively. Note that the outputs produced by the first three slots are all subclasses of \textit{Candidate Classes}.

In addition, the template distinguishes a fourth slot to intersect the results, both input and output are \textit{Candidate Classes}. Three slots
Strict Publication Classification template The second template, depicted in Figure 5.2, is for classification of publications, based on a generic problem solving method [140]. Classification is performed by the sequential activation of four slots. The first slot, requiring MicroMatch for publications, determines on the basis of a set of observations on publication the status of the features present in all class definitions. The second slot, requiring Aggregate, organises the results per class. The third slot, requiring Strict Admit, filters the resulting classes to the selection that strictly meet all class requirements. The fourth slot, requiring Select Best, determines the best matching class out of the filtered classes candidate classes.

The information presented thus far on templates should suffice to understand the trace presented in Section 5.3, in which case the reader interested only in this trace can skip the following part and continue with Section 5.2.3 describing the initial template-based configuration. The remainder of this section elaborates on the construction of templates.
Single-Level Adaptation: 
Reconfiguration using Classification Services

48

Figure 5.2: Activity diagram for the Strict Publication Classification Template.

Template construction

The construction of templates is part of preparing a set of web services for template-based reconfiguration. A template defines, on one hand, a combination of services and, on the other hand, the degree of freedom for future adaptations. There are multiple ways to construct a template, and in this section two different approaches are illustrated: construction based on an existing service, and construction based on a generic task composition.

Construction of a template on the basis of an existing service

This part describes the construction of a template on the basis of a description of an existing service. This approach is used within top-down design of complex services, breaking down existing complex services into simpler services. In this approach, the semantic description of the resulting template is similar to that of the existing service: the template provides another way to achieve the same functionality. In the following example the description of the service FilterStrictClasses, which has been introduced in Section 5.2.1, is used as an example for defining a new template.

The left part of Figure 5.3 represents the information flow in the service FilterStrictClasses, in terms of input and output. The input of the service is Aggregated Scores, and the output is Candidate Classes. The new template needs to have the same input and output, and provide the same functional-
5.2 Structures

Figure 5.3: On the left the activity diagram of a single FilterStrictClasses service, and on the right the activity diagram of a configuration, using 4 services, offering an equivalent functionality.

The description of the template should contain the same attributes as FilterStrictClasses.

The internal structure of the template defines a composition of multiple services to this purpose. The right part of Figure 5.3 illustrates the internal structure of the template. The template contains four slots. Filtering the candidate classes is performed by three different slots, and the final result is created by taking the intersection over the candidate classes’ output provided by the three filter slots.

The rationale behind this structure is the availability of the following set of services:

- FilterConsistentClasses
- FilterKnownClasses
- FilterPresentClasses

These three services together provide the required filtering functionality, each returns only part of the requested result. The intersection service generates the desired result, as illustrated in Figure 5.4.
Figure 5.4: Venn-diagram of the sets of classes filtered by the three separate services. The intersection of the sets results in the desired class set.

Four slots are defined for these four separate services in the new template. The slot requirements are kept simple for this example, posing only restrictions on the functionality, input, and output.

As an example two slots are further described: the first slot responsible for filtering the classes that are consistent, and the fourth slot responsible for creating the intersection of the sets of filtered classes. The three requirements for the first slot concerns the functionality to filter consistent features, as input \textit{Scored Observations}, and as output \textit{Consistent Candidate Classes}. The three requirements for the fourth slot concern the functionality to intersect different inputs, as input \textit{Candidate Classes}, and as output \textit{Candidate Classes}.

After the definition of the slots, the control over these slots is defined as follows. The input of the template, \textit{Aggregated Scores}, is given to all three services, which are activated in parallel. After all three services are finished, the resulting outputs are given to the intersection service, which is then activated, resulting in Candidate Classes. The resulting template is the Strict Admission template.

**Construction of a template on the basis of a generic task composition**  This part describes the construction of a template on the basis of a generic task composition. This approach focuses on defining the internal structure of the template, based on the tasks identified in the generic task composition, whereas the other method of template construction focused on the external behaviour, as defined in the description.

To use a generic task composition for the construction of a template, four steps are performed:

1. Match
2. Reduce/Extend
3. Specialise

4. Instantiate

The first step is to match the desired functionality for the template with the available generic task models. This matching is difficult to automate as it requires broad domain knowledge (e.g., to determine whether a generic task model is defined at the desired level of detail, to assess whether the scope of the model is appropriate, and whether the concepts match). Therefore this matching is performed by hand. The matching results in at least one task model, which offers the desired functionality. A single task model is selected. The second step is to reduce or extend the generic task model, as it might offer more than is requested, or lack certain functionality. Thirdly the adapted generic task model is specialised to tailor it to a more specific task within the targeted domain. The result of this phase is a task model tailored for a specific domain and a specific functionality, used to determine the slot requirements and the dependencies between slots for the template. The fourth step is to create an instantiation of newly required knowledge structures, and an instantiation of propagation functions.

![IBROW Structure of classification services](image)

Figure 5.5: IBROW Structure of classification services. The rectangles represent input and output. The ovals represent the parameterised parts of the IBROW skeleton for the family of classification services (taken from [141]).

**Matching Phase** The example used in this section is to construct a template to perform the classification of publications into categories based
on their keywords. The keywords presented by the authors are used as input. A generic task model for this type of classification can be found in [140], and an instantiation of this model in [139] as a parameterised skeleton, visualised in Figure 5.5. In this model a number of subprocesses are identified for classification: with sequential activation of the subprocesses which in this thesis are referred to as Check, MicroMatch, Aggregate, Admit, Select. The task model offers more functionality than needed, the superfluous part are removed in the following phase.

Reduction/Extension Phase In this step the initial selected task model is evaluated, in which superfluous parts are removed by reduction and additional required functionality is added by extension (e.g., based on parts of other task models). In the example, the selected generic task model for classification is over-complete: the first subprocess Check checks for incorrect observations. In the desired functionality the observations used as input are assumed to be well structured and correct.

Figure 5.6: The reduction of the generic task model and an initial activity diagram for the Classification Template.

The subtask Check is deleted, and the input of the Micromatch task is adapted to handle Observations, instead of only Legal Observations, and the MicroMatch is assumed to include the Knowledge directly, not as an
5.2 Structures

additional input, as depicted on the left-hand side in Figure 5.6. Based on this structure an initial high-level version of the template is constructed, containing a high level description of its control, and a generic definition of its slots, as depicted on the right-hand side in Figure 5.6.

Because the task model contains four remaining separate subtasks, four slots are defined: one for each task. As the four slots are defined in a similar way, only the definition of one slot, the third slot sl3 defined for the third task Admit, is shown as an example.

The generic task model contains information on the generic class/functionality of the subprocess, and its input and output. Table 5.1 depicts this information as high-level requirements on the generic class (rq1), the generic input (rq2), and the generic output (rq3) of the component that is to be inserted into slot sl3.

Table 5.1: Slot requirements for the Admit slot, based on a definition of functionality, input and output.

<table>
<thead>
<tr>
<th>rq</th>
<th>class</th>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>rq1</td>
<td>Admit</td>
<td>Aggregated Scores</td>
<td>Candidate Classes</td>
</tr>
</tbody>
</table>

After defining the requirements for the slots, the control structure is defined. A control structure similar to the control in the generic task composition is used: sequential activation of all the subtasks infers sequential activation over the slots sl1, sl2, sl3, sl4.

For the overall template T_{simple}, a simple description is defined, expressed in Table 5.2 based on the overall functionality, and the defined input and output, similar to the way that slots are defined.

Table 5.2: High level attributes for the template.

<table>
<thead>
<tr>
<th>attrib</th>
<th>class</th>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrib1</td>
<td>Classification</td>
<td>Observations</td>
<td>Selected Class</td>
</tr>
</tbody>
</table>

This resulting basic template provides high-level requirements to create a service composition for classification, however, it is still generic. To enable
reconfiguration in a specific application domain and a specialised task, more specific information is needed. This information is added in the next two phases.

**Specialisation Phase** First, the template is specialised for a specific domain, in this case the classification of publications. The mapping to the publication domain is done by a specialisation of the MicroMatch Slot, to evaluate the observation based on publication related features. The slot of the template is mapped to the domain of classification as follows:

- *MicroMatch*: Micromatching is specialised to publications, classified by keywords. The task *MicroMatch* is replaced by *Publications MicroMatch*.

The resulting template is a generic template for the classification of publications. Second, the template needs to be further specialised to adhere to the following:

1. **Strictness of class selection**: Only allow classes in which all the class features are present, no inconsistent features are present, nor additional features that are unexplained.

2. **Number of solutions requested**: Only request the best matching result, and ignore the rest.

Based on these requirements, more specific slot requirements can be created. *Admit* is specialised to *Strict Admit* to meet the first requirement, and the Select is specialised to *Select Best*.

**Instantiation** This phase is used to finalise the template construction, in which knowledge structures and functions are defined to reason about the template. This last step is important for templates that have propagation functions. These functions need to be accessible to the automated reconfiguration process. In the instantiation phase, these functions are created and deployed, so that they can be referenced in the template descriptions. The resulting template is the Strict Publication Classification Template. In this scenario no propagation functions are modelled and needed, and the instantiation phase can be completed without any additional effort.

This concludes the description of the process to construct a template, based on a generic task model. The available web services and templates have been described, together with a description of the construction of templates.
5.3 Reconfiguration Example

5.2.3 Initial Configuration

To illustrate the reconfiguration process, an initial configuration of web services and templates is used, based on the those described in the previous sections. The initial configuration is a configuration for a classification task, and a hierarchical view of the configuration is presented in Figure 5.7.

This initial configuration is able to handle observations that have at most one value, and in which features can be identified as being inconsistent, unexplained, and missing. Per class the features are gathered, and a number of admissibility criteria are combined: consistent, known, and present. The result is a single class with the highest score.

Figure 5.7: The initial configuration $Conf_{init}$.

For the reconfiguration example, one of the services needs to be marked as a failing service. In this example the service $FilterStrictClasses$ is assumed to fail.

5.3 Reconfiguration Example

This section presents a reconfiguration trace as produced by the process described in Chapter 4. The trace in this section is based on the web services described in Section 5.2.1, the templates described in Section 5.2.2, and the initial configuration described in Section 5.2.3. The failing service is known, and explicitly identified as the web service $FilterStrictClasses$. The trace described in this section is based on an implemented prototype, also described in Chapter 4. The detailed trace generated by the prototype is available in Appendix I.

This example is a simple illustration of single-level reconfiguration, in which the focus does not need to be adapted. It illustrates how reconfiguration is based on the slot requirements defined in the templates, instead of based on the description of the failing service. In addition this example shows that the replacement of a component is not just based on matching
Single-Level Adaptation:  
Reconfiguration using Classification Services

![Diagram]

Figure 5.8: Overview of the trace of the example Classification Reconfiguration.

only atomic services, but on constructing a new template-based subconfiguration as well.

For reference, Figure 5.8 displays an overview of the reconfiguration trace. In the remainder of this section the trace, based on the overview as presented above, is discussed in detail. Based on the initial situation in which the web service *FilterStrictClasses* fails, the trace discusses the following steps:

1. Focus Determination determines the slot with the failing service.

2. Requirement Determination creates a list of requirements based on the requirements for the slot in focus.


4. Integration replaces the failing service with the newly created configuration.

After the Integration process, the reconfiguration finishes with success in a single run.
5.3 Reconfiguration Example

5.3.1 Focus Determination

The goal of Focus Determination is to determine the focus and scope of the adaptation by the reconfiguration process.

In the current scenario the implemented policy is to keep the change as small as possible. The focus is incrementally increased so that in each activation of focus determination, the focus marks a larger part of the configuration to be replaced.

In the first activation of the process of Focus Determination, the focus is by default on the slot with the failing service. As FilterStrictClasses is marked as the failing service, slot $Sl3$ is focused on. As illustrated in Figure 5.9, the scope of reconfiguration is limited to the replacement of only the failing service.

Figure 5.9: The first focus, limiting the area affected by reconfiguration. The focus is on slot Sl3 of the template.

The input of the Focus Determination process:

- Initial configuration: $Conf_{init}$
- Failing web service: $FilterStrictClasses$

The output of the first activation of the Focus Determination process:

- Focus on: $Sl3$

The Focus Determination process has successfully determined a focus. By setting the focus, the scope for the reconfiguration process is set, and the reconfiguration process continues.
5.3.2 Requirement Determination

The goal of Requirement Determination is to gather the requirements for a configuration that can act as a replacement for the configuration currently inserted into the slot in focus. A slot description is contained in the template description within the initial configuration. Requirement Determination determines two types of requirements: requirements based on slot descriptions and structural requirements.

Within the template definition of template $T_{simple}$ the requirements for the slot $Sl3$, which is the slot in focus, are given:

- $r1$ (functionality): FilterStrictClasses
- $r2$ (input): Aggregated Scores
- $r3$ (output): Candidate Classes

To ensure that an alternative is found for the failing service, a structural requirement $r4$ is added, specifying that the failing web service is excluded from use within any slot in the upcoming configuration process.

- $rq4$ exclude(component: FilterStrictClasses) ; AllSlots)

Concluding, the input for the Requirement Determination process is:

- Initial configuration: $Conf_{init}$
- Failing web service: $FilterStrictClasses$
- Focus: $Sl3$

The output of the Requirement Determination process is:

- Requirement Set: $RQS1 = \{r1, r2, r3, r4\}$

5.3.3 Template-based Configuration

The template-based configuration process will attempt to construct a new configuration that satisfies the given requirement set, based on the templates and services available in its repository.

The template based configuration process receives two kinds of requirements:

- requirement for the exclusion of specific structures,
The configuration process starts with the following information:

- Requirement Set: $RQS_1 = \{r_1, r_2, r_3, r_4\}$

The structural requirement $r_4$ for the exclusion of a specific service will be handled by the repository, by excluding this service from its matching results.

The main steps of the configuration process are the following:

1. initiate a new configuration and requirement set,
2. find a component satisfying the given requirement set,
3. incrementally fill in open slots of the top component and inserted templates,
4. check the preconditions and postconditions of the newly created configuration.\footnote{Preconditions and postconditions are checked as a last step, to ensure that the satisfaction of requirements on properties that propagate can be checked.}

**Initiate a new configuration and requirement set**  As at the start there is neither a configuration supplied nor a default configuration available, an empty configuration is created, and the initial requirement set is taken to be the set of requirements provided to the configuration process.

**Find a component satisfying the initial requirement set**  The initial configuration is an empty configuration. The generic policy for the configuration process in this scenario is to prefer simple configurations over complex configurations. Therefore matching is first limited to web services, but in our scenario no results are returned. As no web service could be found, the search is extended to templates. In our scenario, a template is found: Strict Admission, as displayed in Figure 5.10.

**Incrementally fill open slots**  The current configuration under construction has a top component, but is not strongly complete: the template has four open slots. Each of these slots has to be filled with a strongly complete and correct sub-configuration to successfully finish this configuration process. Filling each slot can be considered a separate configuration problem, but interdependencies must be taken into account. For each slot the requirements
are determined, based on the slot definition. If a created sub-configuration in a slot yields additional dependencies, then these dependencies are added as requirements to the slot concerned. Within the configuration process a separate pivot is used to determine the focus.

The pivot focusses on slot $S_{II}$, which is still open, of the StrictAdmission template, based on a random selection, as illustrated in Figure 5.11. Based on the slot definition, requirements are added, bound to the context of slot $S_{II}$ within this configuration. The requirements for slot $S_{II}$ are:

- (functionality): FilterConsistentClasses
- (input): Scored Observations
- (output): Consistent Classes

Similar to the procedure of searching the top level component, a component is sought that satisfies the set of requirements for slot $S_{II}$. As a simple sub-configuration is preferred over more complex configurations, the initial search is limited to web services, and yields a result. The web service FilterConsistentClasses is found, and inserted into slot $S_{II}$. This sub-configuration is strongly complete, as no open slots exist within this sub-configuration, and
5.3 Reconfiguration Example

checking preconditions and postconditions does not result in violations. Filling the first slot is deemed to be successful. The pivot is moved from the first slot to slot $sl_2$, the next empty slot. The process for filling in the three remaining slots is similar to the filling of the first slot, and will not be further described in detail. The resulting subconfiguration $Conf_{new}$ is displayed in Figure 5.12.

![Diagram](image)

Figure 5.12: The new configuration resulting from the Template-based Configuration process, based on the given set of requirements.

**Check preconditions and postconditions** A final check of preconditions and postconditions is performed after a strongly complete configuration (in which all slots in that configuration have been filled) has been constructed. This check is needed to ensure that all dependencies between inserted services and templates are in line with the requirements. Within this example there are no additional preconditions and postconditions. As a strongly complete configuration has been constructed, no violations are found, and all requirements have been satisfied, the template-based configuration determines it has successfully completed its task, and returns the completed configuration as a result.

5.3.4 Integration

The goal of the Integration process is to combine the initial configuration and the newly created sub-configuration into a single template-based configuration. The point at which the two configurations need to be combined is specified by the focus. The integration process contains two subprocesses: to combine the newly created configuration with the initial configuration at the point of the focus; and to annotate the resulting configuration with properties. This annotation step prepares
the configuration for later reconfiguration. In this section the annotation process is not described, but the following chapter presents a scenario in which the annotation process is explained.

The input of the Integration process:

– Initial configuration: \( Conf_{init} \)
– Focus: \( Sl3 \)
– New Configuration: \( Conf_{new} \)

Output:

– Resulting Configuration: \( Conf_{resulting} \)

The initial configuration (as displayed in Figure 5.7) is to be combined with the newly created subconfiguration (as displayed in Figure 5.12). The two configurations are combined at the given focus (as displayed in Figure 5.9). The failing service is removed from slot \( Sl3 \) of the initial configuration, and replaced by the newly created subconfiguration. The resulting configuration structure of \( Conf_{resulting} \) is displayed in Figure 5.13.

![Figure 5.13: The configuration resulting from the Integration process.](image)

The successful integration concludes the reconfiguration process of the single-level configuration, based on a failing service.

### 5.4 Discussion

This chapter describes single-level template-based reconfiguration in more detail for classification of publications. In contrast to most other single-level reconfiguration approaches described in the literature (see Chapter 2), the approach presented in this chapter is neither based on a set of predefined fixes nor on a precise match of a semantic description for a single service. The template-based structure in this thesis allows for the creation of
a subconfiguration to replace the failing service, that meets the requirements
of the this configuration based on the slot-requirements.

In template-based reconfiguration knowledge is local, supporting local
reasoning and adaptation, whereas the IBROW project assumes central knowl-
edge (of which not all has been included in this scenario).
Single-Level Adaptation:

Reconfiguration using Classification Services
Chapter 6

Multi-Level Adaptation: Reconfiguration using Math Services

This chapter presents the second of two scenarios to illustrate template-based reconfiguration, namely multi-level reconfiguration using templates. The domain of classification services is used within this scenario. In multi-level reconfiguration adaptation is not limited to replacing merely the failing service: adaptation can be more extensive, affecting more of the internal structure of the configuration. In addition this section includes reasoning on Quality of Service (QoS) properties of a configuration, specifically accuracy. This reasoning is enabled by propagation functions, by which properties of an overall configuration can be determined based on the properties of the individual services and the templates. This scenario assumes that the failed service has been identified, and that instance-based replacement, which is part of component-based replacement, is not possible.

This chapter is structured as follows. Section 6.1 provides some background of the mathematical services used. Section 6.2 describes the structures available for reconfiguration, Section 6.3 describes the process of reconfiguration, followed by discussion in Section 6.4.

6.1 Background

The services presented in this chapter are based on a set of mathematical services in TRIANA\[89\]. TRIANA is a graphical toolkit for workflows using pluggable components. The set of mathematical services in TRIANA

\[89\]TRIANA project page: http://www.trianacode.org
has been developed by the Grid-Enabled Numerical and Symbolic Services (GENSS) project. Within the GENSS project the main research focus was on matchmaking techniques for advertisement and discovery of mathematical services. The GENSS project addresses the combination of Grid computing and mathematical web services for mathematical problem analysis.

Within this scenario the scope of reuse of the mathematical services is as follows: (1) the TRIANA services have been implemented with a small extension to allow small variations in accuracy (in terms of the number of decimals returned), and to allow access outside of the TRIANA toolkit; (2) simple OWL-S and WSDL annotations of the services have been devised; (3) a set of templates needed to be defined to determine the surface area of a triangle, which enables reasoning about attributes of composed services. An initial configuration has been created for reconfiguration.

Quality of service

The Quality of Service attribute included in reasoning in the scenario in this chapter is accuracy, which is related to numerical properties:

Accuracy: The amounts of decimals of a number.

The following concepts relate accuracy to mathematical services:

Precision: The total number of decimals returned by a mathematical service.

Yield: The maximum number of decimals in the result of a mathematical operation.

Preservation of information: If the precision is greater than or equal to the yield, then none of the generated decimals in a calculation will be lost.

The services included in this domain vary in precision, as they return different numbers of decimals. Precision is interesting when using mathematical services, as two similar services differing only in precision can produce different results. Differences in the results are caused by differences in rounding during the mathematical operations. At design time these differences in outcome may be acceptable, but when a complex service is reconfigured, the basic assumption will be that it will generate the same answers as prior to the

---

GENSS project page: [http://genss.cs.bath.ac.uk](http://genss.cs.bath.ac.uk)
reconfiguration. Accuracy and precision thus are interesting QoS attributes for reconfiguration.

It is assumed that the number of decimals of the inputs is limited a priori. This assumption enables defining requirements on precision for services, that can be checked for satisfaction. Limiting the number of decimals can be achieved by pre-formatting the inputs: for instance, operations based on financial records often have two decimals (e.g., €192.31).

**Simple mathematical operations** Basic mathematical operations vary in the number of decimals in their results, based on the number of decimals of their inputs. To illustrate, Table 6.1 gives an overview for several mathematical operations.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Function to determine the yield ((F_{\text{yield}}))</th>
<th>Example ((x=2.421; y=0.24))</th>
<th>Result</th>
<th>Yield</th>
</tr>
</thead>
<tbody>
<tr>
<td>(z = x - y)</td>
<td>(\max{\text{Acc}_x, \text{Acc}_y})</td>
<td>(2.181)</td>
<td>(3)</td>
<td></td>
</tr>
<tr>
<td>(z = x + y)</td>
<td>(\max{\text{Acc}_x, \text{Acc}_y})</td>
<td>(2.661)</td>
<td>(3)</td>
<td></td>
</tr>
<tr>
<td>(z = x \cdot y)</td>
<td>(\text{Acc}_x + \text{Acc}_y)</td>
<td>(0.58104)</td>
<td>(5)</td>
<td></td>
</tr>
<tr>
<td>(z = x^2)</td>
<td>(\text{Acc}_x + \text{Acc}_x)</td>
<td>(5.861241)</td>
<td>(6)</td>
<td></td>
</tr>
<tr>
<td>(z = \sqrt{x})</td>
<td>(\infty)</td>
<td>(1.555956297586...)</td>
<td>(\infty)</td>
<td></td>
</tr>
</tbody>
</table>

**Reasoning on accuracy in templates** When a template is used to define a complex (as in composed) mathematical operation, the properties of this template related to accuracy are dependent on the properties of the services and templates inserted into its slots. In fact, the *precision* and *yield* of the template equal the precision and yield of the service that generates the output of the template.

The property *preserves information* for a template is dependent on the properties of all the subconfigurations inserted into its slots. If the *preserves information* property holds for all slots, then this property also holds for the whole template.

---

8Note that accuracy is not used in this context to ensure the correctness of a calculation, as in significance, but to ensure that a modification of the configuration will lead to the same answer.
Determining *preserves information* is not trivial, as dependencies are introduced by the control structure within the template. The order of the individual operations is relevant for determining whether each of the slots *preserves information*. Consider the following two examples: \( x \cdot (y + y) \) and \( (x \cdot y) + (x \cdot y) \), which both contain addition and multiplication, but in different orders, with the same result, \( 2xy \). When the same services are used, with the same precision, the answers generated by the two examples differ: one of the calculations preserves information, while the other does not, as shown in Table 6.2.

This example illustrates that this QoS property cannot be determined if a configuration is considered as a black box. However, this property can be determined using the local knowledge in templates. In the next section propagation functions are defined for reasoning on the preservation of information.

Table 6.2: A different order of similar services can affect the results of an operation. In the examples, \( x = 0.43 \) and \( y = 0.991 \), the precision of Addition is 3 and the precision for Multiplication is 6.

<table>
<thead>
<tr>
<th>Example</th>
<th>Unconstrained</th>
<th>Constrained</th>
<th>Information preserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \cdot (y + y) )</td>
<td>0.85226</td>
<td>0.85226</td>
<td>Yes</td>
</tr>
<tr>
<td>( (x \cdot y) + (x \cdot y) )</td>
<td>0.85226</td>
<td>0.852</td>
<td>No, two decimals lost</td>
</tr>
</tbody>
</table>

### 6.2 Structures

This section describes the structures available for reconfiguration within the mathematical services scenario. Section 6.2.1 introduces the services and Section 6.2.2 introduces the templates. Section 6.2.3 presents the initial configuration used in this scenario.

#### 6.2.1 Web Services

The web services in this scenario perform these simple mathematical operations: add, subtract, multiply, square root, and square. The services vary in precision, that is, the number of decimals in their results. Note that there are different services that perform the same mathematical operation, however with different values for the precision, such as for the *subtract* operation.
The properties displayed in Table 6.3 are defined as an OWL-S Profile: ServiceName, ServiceCategory, Description, Input, Output, and Accuracy.

Table 6.3: Selection of properties for available atomic mathematical services.

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Service Category</th>
<th>Profile Input</th>
<th>Profile Output</th>
<th>Description</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ws1</td>
<td>Area</td>
<td>Base, Height</td>
<td>Area</td>
<td>$\frac{1}{2} \cdot Base \cdot Height$</td>
<td>20</td>
</tr>
<tr>
<td>ws2</td>
<td>Sqrt</td>
<td>$x$</td>
<td>$z$</td>
<td>$z = \sqrt{x}$</td>
<td>16</td>
</tr>
<tr>
<td>ws3</td>
<td>Square</td>
<td>$x$</td>
<td>$z$</td>
<td>$z = x^2$</td>
<td>8</td>
</tr>
<tr>
<td>ws4</td>
<td>Square</td>
<td>$x$</td>
<td>$z$</td>
<td>$z = x^2$</td>
<td>9</td>
</tr>
<tr>
<td>ws5</td>
<td>Subtract</td>
<td>$x, y$</td>
<td>$z$</td>
<td>$z =</td>
<td>x - y</td>
</tr>
<tr>
<td>ws6</td>
<td>Subtract</td>
<td>$x, y$</td>
<td>$z$</td>
<td>$z =</td>
<td>x - y</td>
</tr>
<tr>
<td>ws7</td>
<td>Add</td>
<td>$x, y$</td>
<td>$z$</td>
<td>$z = x + y$</td>
<td>9</td>
</tr>
<tr>
<td>ws8</td>
<td>Multiply</td>
<td>$x, y$</td>
<td>$z$</td>
<td>$z = x \cdot y$</td>
<td>8</td>
</tr>
</tbody>
</table>

Most of the web services (ws2 to ws8) perform basic mathematical operations. Web service ws1 does more than a basic mathematical operation, as it determines the surface area of a triangle based on the base and the height of the triangle. This web service is specialised for this specific domain.

These mathematical operations can be combined to offer more complex operations. This is structured using templates, as shown in the following section.

6.2.2 Templates

The given services need to be combined to perform more complex mathematical operations. To support calculations of the area of a triangle, for example, the following templates, as described in Table 6.4, have been developed:

- Template t1: determine the area of a right triangle, given the hypotenuse and base of this triangle,
- Template t2: determine the height of a right triangle, given the base and hypotenuse,
- Template t3 and t4: two templates to determine the squared height of a right triangle (in different ways).
Table 6.4: Selection of properties for available mathematical templates.

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Service Category</th>
<th>Profile Input</th>
<th>Profile Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>Determine Area</td>
<td>Base, Hypotenuse</td>
<td>Area</td>
</tr>
<tr>
<td>t2</td>
<td>Determine Height</td>
<td>Base, Hypotenuse</td>
<td>Height</td>
</tr>
<tr>
<td>t3</td>
<td>Determine Squared Height</td>
<td>Base, Hypotenuse</td>
<td>SquaredHeight</td>
</tr>
<tr>
<td>t4</td>
<td>Determine Squared Height</td>
<td>Base, Hypotenuse</td>
<td>SquaredHeight</td>
</tr>
</tbody>
</table>

Each of these four templates is presented in more detail below.

**Determine Area (t1)** This template, displayed in Figure 6.1, is defined for determining the area of a right triangle (i.e., with a square corner). The lengths of the base and the hypotenuse are known, the length of the height of the triangle is unknown. This template contains two slots:

1. determine the height of the triangle, based on the length of the base and the hypotenuse,
2. determine the area of a triangle, based on the height and the length of the base.

![Activity diagram for the Determine Area template.](image)

Figure 6.1: Activity diagram for the Determine Area template.

The slot requirements on precision in this template are based on:

- the precision of the inputs that the user expects to provide,
For this scenario the accuracy of the inputs is as follows:

- **Base**: 4,
- **Hypotenuse**: 3.

**Determine Height (t2)** This template, displayed in Figure 6.2, determines the height of a right triangle, given the base and hypotenuse. The template contains two slots:

1. determine the height of the triangle, based on the lengths of the base and the hypotenuse,
2. determine the area of the triangle, based on the height and the length of the base.

The precision of the result of template t2 is equal to the precision resulting from the component inserted into slot 2 (**Determine Root**), and requires that the component in slot 1 preserves information in its operations, given the accuracy of the inputs.

**Determine Squared Height (t3)** This template, displayed in Figure 6.3, determines the squared height of a right triangle, given the length of the base and of the hypotenuse. This template completes the basic operations of taking the square of each of the two inputs, and subtracting the results. The template contains three slots:

1. determine the square of a given number,
2. determine the square of a given number
3. subtract two numbers.

Figure 6.3: Activity diagram for the first Determine Squared Height template.

Whether the template preserves information for a given accuracy of the inputs, depends on the properties of the services and templates inserted into its slots. A propagation function is used to determine whether this property holds for a template. Requirements on the properties derived by the individual functions are specified as slot requirements, dependent on the specific configuration in which the template is used.

The propagation function for this template specifies that if for all of the components inserted into its slots the property preserves information holds, then the property preserves information holds for the template. Note that, due to the control structure, the accuracy of the inputs varies for each of the slots.

For the slots the following functions are used to determine the yield (the number of decimals generated):

- Slot 1: \(\text{Yield}_{3,s1} = F_{\text{yield}3, s1}(\text{Acc}_{\text{hypothenuse}})\)
- Slot 2: \(\text{Yield}_{3,s2} = F_{\text{yield}3, s2}(\text{Acc}_{\text{base}})\)

This slot is defined separately, as a stateless execution environment is assumed. Otherwise the first slot could also be reused to perform the second operation, with a different control structure defined for the template.
6.2 Structures

- Slot 3: \[ Yield_{t3,s3} = F_{yield_{t3,s3}}(Yield_{t3,s1}, Yield_{t3,s2}) \]

The functions \( F_{yield_{t3,s1}} \), \( F_{yield_{t3,s2}} \), and \( F_{yield_{t3,s3}} \) are the functions that need to be specified by the services or templates filled into the slots. The following slot requirements are defined based on the derived properties and the requirement preserves information:

- Slot 1: Precision \( \geq Yield_{t3,s1} \)
- Slot 2: Precision \( \geq Yield_{t3,s2} \)
- Slot 3: Precision \( \geq Yield_{t3,s3} \)

**Determine Squared Height (t4)** This template determines the squared height of a right triangle, given the length of the base and of the hypotenuse. This templates is based on usage of services to subtract, add, and multiply, based on the equation \( c^2 - b^2 = (c + b) \cdot (c - b) \), where \( c \) is the length of hypotenuse and \( b \) is the length of the base. This template contains three slots:

1. add two numbers,
2. subtract two numbers,
3. multiply two numbers.

The activity diagram for this template is displayed in Figure 6.4.

![Activity diagram for the second Determine Squared Height template](image-url)
The property propagation function requires specific inputs per slot (as it requires the accuracy of the inputs). The second function is generic, specifying that if the precision of all of the components is greater than or equal to the number of decimals generated, then the property preserves information holds.

For the slots the following functions are used to determine the yield (numbers of decimals generated):

- Slot 1: \( \text{Yield}_{t_4,s_1} = F_{\text{Yield}}(\text{Acc}_{\text{hypotenuse}}, \text{Acc}_{\text{base}}) \)
- Slot 2: \( \text{Yield}_{t_4,s_2} = F_{\text{Yield}}(\text{Acc}_{\text{hypotenuse}}, \text{Acc}_{\text{base}}) \)
- Slot 3: \( \text{Yield}_{t_4,s_3} = F_{\text{Yield}}(\text{Yield}_{t_4,s_1}, \text{Yield}_{t_4,s_2}) \)

The following slot requirements are defined based on the derived properties and the requirement preserves information for each of the slots of template \( t_4 \):

- Slot 1: Precision \( \geq \) \( \text{Yield}_{t_4,s_1} \)
- Slot 2: Precision \( \geq \) \( \text{Yield}_{t_4,s_2} \)
- Slot 3: Precision \( \geq \) \( \text{Yield}_{t_4,s_3} \)

Note that both the propagation functions and the requirements on the derived properties of template \( t_3 \) and \( t_4 \) seem alike. This is because the control structure is similar. A difference in the control structure, or in the number of slots, would lead to different propagation functions and different requirements on the derived properties.

6.2.3 Initial Configuration

This section introduces the initial service configuration, based on the templates and web services described in the previous sections. The templates and services are available in a local service repository. The initial configuration contains three templates and five web services. Figure 6.5 depicts a hierarchical view of the configuration.

Template \( t_1 \) is used to determine the area of a right triangle based on the base and the hypotenuse. Template \( t_1 \) divides this problem into two parts: determine the height, and determine the area based on the base and the height.

Template \( t_2 \) is used to determine the height of a right triangle. To determine the height (\( a \)) of a right triangle (see Figure 6.6), the Pythagorean
6.2 Structures

Figure 6.5: Hierarchical view of the initial configuration.

Figure 6.6: The triangle of which the area needs to be determined. Both the Base \((b)\) and the Hypotenuse \((c)\) are known, but the Height \((a)\) is unknown. The angle between \(a\) and \(b\) is 90°.

The theorem \((a^2 + b^2 = c^2)\), thus \(a = \sqrt{c^2 - b^2}\) is used, where \(c\) is the hypotenuse, and \(b\) is the base. Template t2 divides this problem into two parts: determine the squared height and determine the root of the resulting answer. The first slot has been filled with template t3, and the Determine Root slot with web service ws2.

Template t3 is used to generate the squared height. Two services are used to determine the square of the hypotenuse and square of the base, and a third service is used to subtract the results.

The initial configuration has been designed to handle inputs within a range of accuracy. Based on the information on the inputs, requirements for accuracy of the slots of the configuration can be determined.

The next example focusses on the accuracy related requirements for the slots in template t3 of this configuration. These are determined requirements on the properties which are derived from the configuration. To use this function for the QoS property preserves information, the preconditions for the functions are met:

- the maximum numbers of decimals of the input are specified as requirements \(Acc_{hypotenuse} = 3\) and \(Acc_{base} = 4\),
- Slot \(Sl_{t2,s1}\) requires that the inserted component preserves information.
Using the slot requirements for the required precision of each of the slots of template $t_3$, which are:

- $Sl_{t3,s1}$: Precision $\geq Yield_{t3,s1}$
- $Sl_{t3,s2}$: Precision $\geq Yield_{t3,s2}$
- $Sl_{t3,s3}$: Precision $\geq Yield_{t3,s3}$

The function to determine the yield, as specified in the template definition, can be refined to the functions of the services (see Table 6.1) inserted in the slots:

- $Yield_{t3,s1} = Fyield_{t3,s1}(Acc_{hypothenuse}) = Fyield_{square}(Acc_{hypothenuse}) = Acc_{hypothenuse} + Acc_{hypothenuse} = 3 + 3 = 6$
- $Yield_{t3,s2} = Fyield_{t3,s2}(Acc_{base}) = Fyield_{square}(Acc_{base}) = Acc_{base} + Acc_{base} = 4 + 4 = 8$
- $Yield_{t3,s3} = Fyield_{t3,s3}(Yield_{t3,s1}, Yield_{t3,s2}) = Fyield_{subtract}(6, 8) = max\{6, 8\} = 8$

One of the assumptions for reconfiguration in this thesis, is that a configuration has been prepared for reconfiguration, and that the initial configuration should not violate the given requirements. The satisfaction of the requirements can be checked by comparing the requirements with the precision of the services inserted into the slots. The precision should be greater than or equal to the yield of each slot.

- $Sl_{t3,s1}$: Precision$_{ws3} = 8$
- $Sl_{t3,s2}$: Precision$_{ws4} = 9$
- $Sl_{t3,s3}$: Precision$_{ws5} = 10$

This confirms that for the template $t3$ in this configuration, the property preserves information holds. The initial configuration does not violate the requirements and is ready for reconfiguration.
6.3 Reconfiguration Example

This section presents a reconfiguration trace based on the process defined in Chapter 4 using the web services described in Section 6.2.1, the templates described in Section 6.2.2, and the initial configuration described in Section 6.2.3. The failing service is known, and explicitly identified as the component \textit{ws5}, the Subtract service. The initial information is given to the reconfiguration process, and the start of the reconfiguration process is requested. The trace described in this section is based on an implemented prototype, also described in Chapter 4. The detailed trace generated by the prototype is available in Appendix J.

Figure 6.7: Overview of the trace of example Math reconfiguration.

This example is a simple illustration of multi-level reconfiguration, in which the adaptation has consequences beyond only the failing web service. In addition this example shows how reasoning with QoS attributes is done, and how requirements and properties are propagated through a configuration composed of different levels.

For reference, Figure 6.7 displays an overview of the reconfiguration trace. The initial situation is that the web service that performs subtraction has failed in the initial configuration.
6.3.1 Focus Determination (1st)

The goal of Focus Determination is to determine the focus and scope of the adaptation that is being performed by the reconfiguration process as a whole. The default strategy for Focus Determination is single-level reconfiguration: the scope of adaptation is minimised, and the focus is limited to the slot with the failing web service. The failing service is identified as \(ws5\), in slot \(Sl_{l3,s3}\). Therefore, the first focus is on slot \(Sl_{l3,s3}\), as illustrated by Figure 6.8.

![Figure 6.8: The resulting first focus, limiting the area affected by reconfiguration to slot \(Sl_{l3,s3}\).](image)

Input of the Focus Determination process:

- Initial configuration: \(Conf_{init}\)
- Failing web service: \(ws5\)

Output of the Focus Determination process:

- Focus: \(Sl_{l3,s3}\)

The Focus Determination process has successfully found a focus. By having determined the focus, the scope for the rest of the reconfiguration process is set and the reconfiguration process continues.
6.3.2 Requirement Determination (1st)

The goal of Requirement Determination is to gather the requirements for a configuration that can act as a replacement for the configuration currently inserted into the slot in focus. A slot description is contained in the template description within the initial configuration. Requirement Determination determines two types of requirements: requirements based on slot descriptions and structural requirements.

Within the template definition of template $t_{i3}$ the requirements for slot $Sl_{3,s3}$ are found:

- $r_1$ (functionality): Subtract
- $r_2$ (input): Number
- $r_3$ (output): Number
- $r_4$ (qos): Precision $\geq Yield_{i3,s3}$

As the template-based configuration process needs to find an alternative filling for slot $Sl_{3,s3}$, the requirements $r_1, r_2, r_3, r_4$ are selected. To ensure that an alternative configuration is found, a structural requirement $r_5$ is added, specifying that the failing web service $ws5$ is excluded from use within any slot in the upcoming configuration process.

- $r_5$ exclude(component: $ws5$ ; Allslots)

Concluding, the inputs for the Requirement Determination process are:

- Initial configuration: $Conf_{init}$
- Failing web service: $ws5$
- Focus: $Sl_{3,s3}$

The output of the Requirement Determination process is:

- Requirement Set: $RQS1 = \{r_1, r_2, r_3, r_4, r_5\}$
6.3.3 Template-based Configuration (1st)

The template-based configuration process attempts to construct a new configuration that meets the given requirement set, based on the templates and services available in its repository. The configuration process starts with the following input information:

- Requirement Set: \( RQS1 = \{ r1, r2, r3, r4, r5 \} \)

The structural requirement \( r5 \) for the exclusion of a specific web service is handled by the repository.

The main steps in this configuration trace are the following:

1. Initiate a new configuration and requirement set.
2. Find a top component satisfying the given requirement set.
3. When a strongly complete configuration has been constructed, check the non-functional requirements, preconditions and post-conditions.

**Initiate a new configuration and requirement set.** An empty configuration \( Conf_{add1} \) is created as the initial configuration. The initial requirement set is based on the set of requirements provided to the configuration process.

**Find a top component satisfying the given requirement set.** The policy for selecting components is to prefer simple configurations over complex configurations. A web service matching the requirements is preferred over a template. The component repository is requested to return web services that fulfill the requirements. The repository contains two services that offer the operation Subtract (ws5 and ws6), however the failing service (ws5) is excluded from the match results by the component repository due to the structural requirement. A web service ws6 is returned as the only result, this component is set as the top component of \( Conf_{add1} \). Finding a top component has succeeded, and \( Conf_{add1} \) is a strongly complete subconfiguration, as the top component is a web service. Now the non-functional requirement can be checked.

**Check non-functional requirements, preconditions and postconditions.** A final check of non-functional requirements, preconditions and post-conditions is performed after a strongly complete configuration has been constructed. Requirement \( rq4 \), specifying that the precision needs to be greater
than or equal to the $Yield_{t_3,s_3}$, is checked. The $Yield_{t_3,s_3}$ equals 8, which has been annotated with the initial configuration, as described in Section 6.2.3. As $w6$ is the top component of $Conf_{add1}$, the properties of $w6$ determine the properties of $Conf_{add1}$. The precision of 6 is too low to satisfy $rq4$. Therefore $Conf_{add1}$ is rejected. As the template-based configuration process cannot create a configuration satisfying the given requirement set, it fails. The failure of the template-based configuration process leads to a new activation of Focus Determination.

### 6.3.4 Focus Determination (2nd)

As the configuration process did not succeed, a new focus is determined. The default strategy of a single-level reconfiguration approach has failed. Therefore the focus will be adapted to examine a larger part of the internal structure: multi-level reconfiguration. Based on the template-based structure of the initial configuration, the focus moves up to the slot that contains the template with the faulty service ($ws5$). This template $t_3$ is in slot $Sl_{2,s1}$.

As illustrated by Figure 6.9, the possible adaptation is now extended to a larger part of the initial configuration.

Figure 6.9: The second focus, changing the area affected by reconfiguration to the first slot of template $t_2$.

Input of the Focus Determination process:

- Initial configuration: $Conf_{init}$
- Failing web service: $ws5$
Previous focus: $S_{t_3,s_3}$

Output of the Focus Determination process:

- Focus: $S_{t_2,s_1}$

### 6.3.5 Requirement Determination (2nd)

Requirement Determination gathers the requirements for the slot in focus. Within the template description of template $t_2$ the requirements for slot $S_{t_2,s_1}$ are located:

- r6 (functionality): Determine Squared Height
- r7 (input): Base, Hypotenuse
- r8 (output): Squared Height
- r9 (qos): Preserves information

As the template-based configuration process needs to find an alternative filling for slot $S_{t_2,s_1}$, the requirements $r6, r7, r8, r9$ are selected. To ensure that the resulting configuration does not contain the failing service, structural requirement $r5$ is added, specifying that the failing web service is excluded from use within any slot in the upcoming configuration process. Note the structural requirement does not exclude the template $t_3$ to be used in slot $S_{t_2,s_1}$: the template could still yield a satisfactory result, with other services inserted into its slots.

- rq5: exclude(component: ws5 ; Allslots)

Concluding, the inputs for the Requirement Determination process are:

- Initial configuration: $Conf_{init}$
- Failing web service: $ws5$
- Focus: $S_{t_2,s_1}$

The output of the Requirement Determination process is:

- Requirement Set: $RQS_2 = \{r6, r7, r8, r9, r5\}$
6.3 Reconfiguration Example

6.3.6 Template-based Configuration (2nd)

Based on the new requirement set a configuration is to be constructed. The preference is for a simple configuration: a single web service.

The configuration process starts with the following information:

- Requirement Set: $\text{RQS2} = \{r6, r7, r8, r9, r5\}$

The structural requirement $r5$ for the exclusion of a specific web service is again handled by the repository, by excluding the failing service from its matching results.

The main steps of the configuration trace are the following:

1. Initiate a new configuration and requirement set.
2. Find a top component satisfying the given requirement set.
3. Incrementally fill open slots.
4. When a strongly complete configuration is created, check the preconditions and postconditions.

**Initiate a new configuration and requirement set.** At the start of the configuration process a new configuration is initiated. As there is no configuration supplied nor a default configuration available, an empty configuration $C_{add2}$ is created. The initial requirement set is based on the set of requirements provided to the configuration process.

**Find a top component satisfying the given requirement set.** The policy for configuration is to prefer simple configurations over complex configurations. Therefore the matching result is first limited to web services, which does not succeed. As a consequence the search is broadened to also cover templates. This results in finding template t4, as displayed in Figure 6.10. Template t4 is set as the top component of the configuration $C_{add2}$.

**Incrementally fill open slots.** The current created configuration $C_{add2}$ has a top component, but is not strongly complete: the template has three open slots. Each of these slots has to be filled with a strongly complete and correct sub-configuration to successfully finish this configuration process, otherwise another top component must be selected. Each slot can be
Multi-Level Adaptation: 
Reconfiguration using Math Services

considered a small configuration problem, but interdependencies must be
taken into account. For each slot the requirements for that slot are deter-
mined, based on the slot definition. If a sub-configuration in a slot creates
additional dependencies, then these dependencies are added as requirements
of the slot. Within the configuration process a separate pivot is used to de-
termine the focus within the template-based configuration process (enabling
to focus reasoning on filling a specific slot).

A small jump is made in the trace, to the point that $S_{4,s1}$ is filled with
$ws7$, and $S_{4,s3}$ with $ws8$. These two slots have been filled by matching the
requirements of these slots. The pivot focusses on the last open slot $S_{4,s2}$,
as displayed in Figure 6.11.

The requirements for slot $S_{4,s2}$ are:

– (functionality): Subtract
– (input): Number
– (output): Number

Similar to the procedure of searching components for the other slots, a
component is sought that satisfies the set of requirements for slot $S_{4,s2}$. As
6.3 Reconfiguration Example

A simple sub-configuration is preferred over more complex configurations, the initial search is limited to web services, and yields a result. Web service ws6 is found, and inserted into slot $S_{t4,s2}$.

The resulting sub-configuration $C_{add2}$ is strongly complete, as all slots have been filled with strongly complete subconfigurations. This configuration is displayed in Figure 6.12. However, it is still unknown whether this configuration meets the QoS requirement rq9, specifying that the configuration needs to preserve information.

Figure 6.12: Configuration $Conf_{add2}$ resulting from the Template-based Configuration process, based on the given set of requirements.

Check non-functional requirements, preconditions and postconditions. A final check of preconditions and post conditions is performed after a strongly complete configuration has been found. Within this context also the QoS attributes are taken into account (rq9). This requires determining the QoS property preserves information for the configuration $C_{add2}$, in this case based on template $t4$ and the services inserted into its slots. To this end the property propagation function is used, that uses the QoS attributes of the inserted services as input. The property propagation function for template $t4$ has been specified earlier in Section 6.2.2.

The following requirements are part of the requirement set:

- the maximum numbers of decimals of the input are specified as requirements as $Acc_{hypotenuse} = 3$ and $Acc_{base} = 4$,
- slot $S_{t4,s3}$ requires that the inserted component preserves information.

The function to determine the yield, as specified in the template definition, can be refined to the functions of the services (see Table 6.1) inserted into the slots:

- $Yield_{t4,s1} = F_{yield_{t4,s1}}(Acc_{hypotenuse}, Acc_{base}) = F_{yield_{add}}(3, 4) = max\{3, 4\} = 4$
Multi-Level Adaptation: Reconfiguration using Math Services

$\text{Yield}_{t_4,s_2} = F_{\text{yield}}_{t_4,s_2}(Acc_{\text{hypothenuse}}, Acc_{\text{base}}) = F_{\text{yield}}_{\text{subtract}}(3, 4) = \max\{3, 4\} = 4$

$\text{Yield}_{t_4,s_3} = F_{\text{yield}}_{t_4,s_3}(\text{Yield}_{t_4,s_1}, \text{Yield}_{t_4,s_2}) = F_{\text{yield}}_{\text{multiply}}(4, 4) = 4 + 4 = 8$

For each slot its yield can be compared with the precision of the inserted service to determine whether it preserves information:

- $\text{Sl}_{t_4,s_1}$ preserves information if: $\text{Prec}_{ws^7} >= \text{Yield}_{t_4,s_1}; 9 >= 4; \text{True}$;
- $\text{Sl}_{t_4,s_1}$ preserves information if: $\text{Prec}_{ws^6} >= \text{Yield}_{t_4,s_1}; 6 >= 4; \text{True}$;
- $\text{Sl}_{t_4,s_1}$ preserves information if: $\text{Prec}_{ws^8} >= \text{Yield}_{t_4,s_1}; 8 >= 8; \text{True}$;

The propagation function determines that template$_4$ preserves information, as for each of its slot the property preserves information holds. As $t_4$ is the top level component, this property also holds for $\text{Conf}_{\text{add}^2}$. The Template-based Configuration process concludes that the created configuration $\text{Conf}_{\text{add}^2}$ is strongly complete, correct and satisfies all requirements specified in RQS2. The process has finished successfully and delivered as result the configuration $\text{Conf}_{\text{add}^2}$. This configuration is ready to be integrated with the initial configuration.

6.3.7 Integration

The goal of the Integration process is to combine the initial configuration and the newly created sub-configuration into a single new template-based configuration. The point of which the two configurations need to be combined is specified by the current focus. The integration process contains two subprocesses: combining the additional configuration with the initial configuration at the point of the focus; and annotating the resulting configuration with resulting properties. This additional annotation prepares the configuration for later reconfiguration.

Inputs of the Integration process:

- Initial configuration: $\text{Conf}_{\text{init}}$
- Focus: $\text{Sl}_{t_2,s_1}$
- New Configuration: $\text{Conf}_{\text{add}^2}$

Output of the Integration process:

- Resulting Configuration: $\text{Conf}_{\text{result}}$
Combining the configurations

The initial configuration (as displayed in Figure 6.5) is to be combined with the newly created subconfiguration (as displayed in Figure 6.12). The two configurations are combined at the current focus (as displayed in Figure 6.9). The faulty web service ws5 and the whole template containing the faulty service are removed from slot $S_{12,s1}$ of the initial configuration, and replaced by the newly created subconfiguration. The resulting configuration $Conf_{result}$ is displayed in Figure 6.13.

Figure 6.13: Configuration $Conf_{result}$ resulting from the Integration process.

Annotation

The previous subprocess created a strongly complete configuration. To facilitate reconfiguration, the annotation of this configuration is to include the propagated properties, which will have its influence on the QoS requirements, if the configuration needs to be adapted again. This additional annotation is created using an propagation function based on the newly created configuration. While the QoS requirements are dependent on which services and templates are used, this is recalculated in principle for all the slots in the configuration.

In this case, the changes for the slot requirements for the precision are illustrated, by using the propagation functions related to the preservation of information. These propagation functions determine the yield for each slot. By including the propagated properties explicitly in the annotation of each level of the configuration, reconfiguration is simplified, as all relevant properties are available as local knowledge.

The yield of each slot is determined, similar as in the last step described in Section 6.3 and included as derived properties. If the resulting service
configuration would need to be reconfigured, then the following properties hold:

- \( \text{Yield}_{t_4,s_1} = 4 \)
- \( \text{Yield}_{t_4,s_2} = 4 \)
- \( \text{Yield}_{t_4,s_3} = 8 \)

After concluding the annotation, the reconfiguration process successfully finishes. This process has (1) adapted a larger part of the initial configuration, than just the failing part; (2) found a replacement, that still adheres to the global QoS constraints; and (3) prepared the adapted configuration for reconfiguration, by completing annotations of the derived properties, required to assess the slot requirements locally in case of reconfiguration.

### 6.4 Discussion

This chapter presents an example scenario of multi-level reconfiguration, with explicit reasoning about requirements of the overall configuration. For the annotations, further integration with work developed in the MONET project\(^{10}\) could enable more extensive reasoning. In the MONET project service annotations have been developed specifically for mathematical services.

The Quality of Service attribute chosen in this scenario is an alternative to more common QoS parameters, such as response time and reliability. With the QoS attribute chosen in this scenario, less numerable aspects have been illustrated (specifically *preserves information*). The propagation function and requirement on the derived properties could as well be applied to other QoS attributes.

Within this scenario, and in this approach, it is assumed that properties are propagated upward before the start of the reconfiguration. This assumption can be relaxed if Requirement Determination is extended with additional functionality to determine these properties on the fly, similar to the annotation part of the Integration process.

The strong conceptual model of the mathematical domain has potential for automated creation of templates, as mathematical equality can be used to determine whether two alternative templates offer the same functionality. This could serve as a basis for automated extension of the (core) contents of a mathematical repository.

\(^{10}\)Project page MONET: [http://monet.nag.co.uk](http://monet.nag.co.uk)
Chapter 7

Discussion and Future Research

This chapter discusses the contributions of this thesis and future research. Section 7.1 presents the results and contributions of this thesis. Section 7.2 proposes some directions for future research. Section 7.3 concludes with a selection of proposals for the application of automated template-based reconfiguration to other environments and application domains than the ones portrayed in this thesis.

7.1 Results and Contributions

This thesis has as its hypothesis that automated adaptation of complex services is feasible using local knowledge. Local knowledge allows changes to be local, reducing structural change to a minimum. To create a system able to autonomously perform reconfiguration, Chapter 1 poses the following research questions:

Q1 How can local knowledge be represented and included within a configuration?

Q2 What process is needed for automated re-configuration using local knowledge?

Chapter 2 presents related research on automated reconfiguration, and concludes that each of the different approaches has its strengths and weaknesses. Component based replacement is an option that remains closest to the initial configuration, but depends heavily on the availability of an exact match. Single-level reconfiguration supports more advanced adaptations, but this requires each configuration to be specifically annotated. Multi-level configuration allows more variation in adapting the part afflicted by
the reconfiguration, but requires more compositional annotated structures. Configuration as reconfiguration is often hindered by the lack of explicit requirements for the configuration that needs to be reconfigured. Ideally a hybrid approach is constructed, which can vary the scope of the adaptation, and allows the most appropriate approach for the selected scope. This is the approach described in this thesis.

The representation and inclusion of local knowledge within a configuration is addressed in Chapter 3. In this chapter the following structures are defined for this purpose: templates in Section 3.2, slots in Section 3.2, and template-based configurations in Section 3.3. A template defines and annotates the requirements associated with a complex web service in more detail than an individual service, yet in less detail than a complete complex service. A template represents a single level of a configuration. The slots in a template specify the requirements for each of the components (i.e., a web service or a template) in a template. Propagation, see Section 3.4, resolves dependencies between different levels in a template-based configuration to enable reasoning on requirements on, for example, Quality of Service using local compositional.

The process needed for automated re-configuration using local knowledge is addressed in Chapter 4. A generic reconfiguration process is defined, with subprocesses Focus Determination in Section 4.2.1, Requirement Determination in Section 4.2.2, Template-based Configuration in Section 4.2.3, and Integration in Section 4.2.4. A prototype is implemented as described in Section 4.3, with further details in Appendices D to H.

The need for reconfiguration is based on failure of a service used in a complex configuration.

The feasibility of template-based reconfiguration is shown by its application in two domains: classification services in Chapter 5, and mathematical services in Chapter 6. Single-level reconfiguration is illustrated in Chapter 5, which illustrates that the approach presented is at least as strong as the current reconfiguration approaches. Multi-level reconfiguration, illustrated in Chapter 6, shows the added value of a gradual increase of the scope of adaptation, and of local reasoning on Quality of Service attributes. A single prototype supports two use cases (i.e., scenarios). These two use cases show that (1) the structures defined to represent and include local knowledge are generic enough be used with existing services from two different domains, (2) the process defined for reconfiguration is generic enough to resolve reconfiguration problems in two different domains, and that it can handle both single-level reconfiguration challenges, as described in Section 5.3, as multi-level reconfiguration challenges, as described in Section 6.3.
7.2 Strengthening the Approach

This section discusses future research to extend the applicability of the approach of this thesis. Directions for structure are given in Section 7.2.1 and directions for process in Section 7.2.2.

7.2.1 Structure

Regarding the structure, strengthening the approach concerns the following aspects:

- scalability,
- propagation,
- requirements,
- service level agreements,
- semantic descriptions.

**Scalability.** A template-based configuration is a composition of templates and web services. Compositional modelling has its challenges [75], for example as modelling remains knowledge intensive, knowledge acquisition is a potential bottleneck. Reduction of the knowledge needed should lessen the knowledge acquisition effort. This could be enabled by reusing existing knowledge sources, and defining generic knowledge, for example default propagation rules. Alternatively, automated annotation is to be explored [119].

To model small variations of a compositional task may require the development of a new compositional model, which would mean for the template-based approach the need to create a new template for each variation. This could be partially resolved by the development of a parameterised super-template, which could be used to generate different templates for different parameters.

In the design of a set of compositional components, the designers can forget to make assumptions explicit, which can hinder the reuse of the components in combination with other sets. For architectural mismatches, [53] proposes directions that could resolve some of these problems. Similar solutions could be used to support designers to make conceptual assumptions
Discussion and Future Research

explicit, such as providing techniques to bridge mismatches, and the development of guidelines for architectural design. The matured state of development of semantic annotation tools, such as Protégé [103], enhances the support for making these assumptions explicit.

The level of granularity on which services are decomposed is an open issue, as more fine-grained components allow more variations. However, it requires additional knowledge on which combinations are feasible. To extend the set of available components, more research in the (automated) generation of templates would be interesting, similar research on the creation of workflows such as [55]. Resources which could be used to construct templates are, for example, patterns as in [112, 56, 52, 144, 118], Generic tasks [36], or cases from Case-Based Reasoning [1].

Propagation. Propagation can be further researched and extended, for example, in to include information on real time QoS properties [121]. Properties related to security in services [16, 155], for example, have not discussed. To which extent security can be included in a compositional approach such as the approach presented in this thesis requires further research [41, 29, 11].

Different ways of supporting mathematical expressions in OWL-S need to be further explored. For example, OWL-S has handlers to support expressions in SWRL or KIF. These handlers may possibly provide a means to define mathematic expressions.

Furthermore, support for defining local knowledge requirements to use in propagation is needed. For complex web services several approaches, such as single level knowledge included in approaches such as [32, 164, 65] to reason about overall Quality of Service, may be used to describe global requirements within single templates and their translation to individual components in a template.

Service Level Agreements. The approach presented has no formalised use, such as Service Level Agreements, of services, nor a way to incorporate sanctioning mechanisms for services that fail to meet their specifications, such as in [114, 113, 71]. A possible usage could be to negotiate Service Level Agreements, e.g., using WS-Agreements [6, 7], based on the requirements specified for slots. The service or template used to fill in the slot would have to adhere to the SLA.

Semantic descriptions. Reconfiguration of complex configurations using components other than web services or templates are not discussed within this thesis. The reconfiguration process could, in theory, treat other com-
ponents similar to web services, as long as the requirements can be related to their description. Another possibility is to extend the template-based approach to support descriptions other than OWL-S, for example BPEL [68], or WSMO [117].

7.2.2 Process

Regarding the process, strengthening our approach concern the following aspects:

- general applicability,
- policies and service level agreements,
- distributed reconfiguration,
- integration.

**General applicability.** The presented model assumes that the reconfiguration process is not constrained in time. A possible extension is to enforce time constraints on the configuration process, based on heuristics to decide when to change focus. As the finiteness of the template-based reconfiguration process is dependent on the finiteness of the configuration process, adding time constraints specifically on the configuration process is an effective solution. The Generic Design Model, on which our configuration process is based, would allow this by means of design process objectives for a configuration process, used to differentiate strategies to explore the configuration space [146].

A second point for improvement of the process is the handling of multiple failing services, as in the presented model multiple failures are handled sequentially which might be an inefficient solution. A possible solution would be to opt for a less local initial focus that contains all failing components.

A third possible extension of the template-based reconfiguration process is the inclusion of a history of previous solutions, which can improve the performance if similar requests for reconfiguration have to be processed.

A fourth extension would be to use the knowledge in a multi-level template-based configuration for multi-level monitoring of QoS properties.

**Policies and Service Level Agreements.** For reconfiguration and configuration the current applied policy is based on preferring local adaptations and components of low complexity. This is a generic policy, but an interesting extension would be to allow domain specific policies, for example, preferring
default slot fillings for templates. Policies might also take other aspects into account, of the amount of time, or the resources available for reasoning into account. In distributed reconfiguration, policies might vary per location, to meet local requirements on the process.

If Service Level Agreements are incorporated, policies could be specified to include or exclude specific organisations offering web services or templates. To incorporate Service Level Agreements in the reconfiguration process integration is needed with SLA negotiation, such as extending WS-Agreement [6].

Another interesting aspect for policies is to take into account a standard over capacity for Quality of Service attributes to increase the robustness of the Quality of Service of the overall configuration.

**Distributed reconfiguration.** The reconfiguration process in this thesis has been viewed from a centralised perspective, but the reconfiguration process can easily be modelled as being distributed. There is direct constraint on selecting multiple focusses in parallel, and having multiple template-based configuration processes running at the same time for the different focusses. This would require the addition of a policy to the Integration process for handling multiple outcomes of the configuration process. Such policies could have preferences on, for example, the duration of the configuration process (first available result is chosen), or select the solution offering the best overall Quality of Service using the functions.

Focus Determination could be done distributed such that the decision to replace a larger part of the configuration with a new configuration is made locally by a reconfiguration manager for each template, as is proposed for our approach in [148]. Distributed determination of the focus is, for example, useful in enabling a distributed version of a self-healing system.

**Integration.** This paragraph discusses integration with other configuration and matching processes, as well as integration with processes for self-healing and self-optimisation.

**Other configuration approaches.** The template-based reconfiguration model is not restricted to using a configuration process based on the Generic Design Model [145], as presented in this thesis. Other configuration processes can be used, provided that these are able to handle requirements resulting from the Requirement Determination process, and that the result is described as a web service. Other configuration processes could enable the
application of template-based reconfiguration to a broader range of components, offering alternative solutions, or improvements in performance.

**Other matching services.** If a configuration process could use multiple matchers, it would enable the use of matchers, for example [40, 77, 67, 86, 87, 132, 108, 163], with specific domain knowledge, or with access to specialised services. Using an open matching architecture, such as Knoogle[37], would enable multiple matching mechanisms to be used, on a set of user-defined repositories—this is particularly useful to enable application-specific matchers to co-exist alongside generic matchers based on term syntax. An architecture combining the approach presented in this thesis and Knoogle is described in [137].

**Enabling Self-healing.** Self-healing [74], further discussed in [73, 125], enables a system to automatically detect, diagnose and repair localised problems, thus removing the need of direct human intervention. To achieve dynamic self-healing of complex web services based on the approach presented in this thesis, the reconfiguration process needs to be extended with both monitoring and deployment. For monitoring propagation offer interesting information. At each level, slot requirements locally defined using propagation to determine the constraints that need to be monitored. The monitoring process is responsible to identify failing services, to deactivate the current complex service, and to initiate a reconfiguration process. Deployment is responsible for deploying the reconfigured complex service in the required environment, and to ensure that the new complex service is registered as the replacement for the failing service.

For simple adaptations state could be preserved, by making the state of complex service external, as in [60], or using components and a framework that allows state to be saved and re-injected, as in [64].

**Enabling Self-optimisation.** [74] enables systems to continuously explore options to improve their performance. Template-based reconfiguration could also be used within self-optimisation of existing services. As the reconfiguration process as described is not directly coupled to instantiation of the resulting configuration, a virtual ghost-reconfiguration process could be started for any component in the configuration in which that particular component is marked as a (virtual) failing service. The configuration process can be optimised to prefer components with better QoS properties, by extending

---

11KNOOGLE project page: [http://knoogle.cs.bath.ac.uk/](http://knoogle.cs.bath.ac.uk/)
the generic configuration process policies. If the ghost-reconfiguration process finds a new configuration, then the QoS properties of this new configuration. If the new configuration would be more efficient (that is, with a better Quality of Service), then the current running complex service can be deactivated, and the more efficient complex service deployed. The mechanism to replace a running existing configuration is a point of research. Assumptions for template-based self-optimisation to work, is when configurations have stable QoS properties, and when continuously new components are available in repositories.

7.3 Extensions to Other Environments and Application Domains

This section describes a number of interesting environments for the application of template-based reconfiguration: agent-based environments, workflow environments and virtual organisations. Furthermore, two interesting application domains are presented: overlay networks for adaptation, and autonomic computing.

Agent-based environments. Agents operate in environments that are typically dynamic, and they need to adapt if they are not able to cope with the changes of their environments [25]. The pro-activeness and autonomy of agents [160] allow them to reason about how they adapt. Such adaptations can range from knowledge, to interface, or even to the structure of the agents. For knowledge-based and interface-based adaptations, solutions for structure-based adaptations are not commonly available [149].

A service for an agent to request structure-based adaptation, as proposed in [25], would allow agents not only to reason on adaptations, but also to actually be adapted. This does require agents to be compositionally structured. By structuring agents using templates and components, the template-based reconfiguration can be applied to software agents.

For agent reconfiguration it would be interesting to further explore the preservation of state after reconfiguration, for example, if reasoning components are also reconfigurable.

Another interesting application is the generative migration of agents [21, 106, 107], in which the migration of an agent is supported by a reconfiguration process that adapts the agent to the requirements of the targeted location. A possible issue in the agent domain is the lack of a unified model for software agents, and the lack of availability of annotated components to build software
Another possibility to integrate automated template-based reconfiguration would be by making both web services and a web service reconfiguration service accessible to software agents, allowing them to adapt existing complex web services on demand. This could be enabled by using an agent platform which provides agent with access to web services, such as AgentScape which offers a Web Service Gateway \[105\].

**Workflow environments.** Integration of our reconfiguration approach in a GRID environment is an area with interest, due to the broad support for deployment in large-scale environments \[50, 44\]. The presented template-based reconfiguration offers a high-level adaptivity to GRID workflows, while the GRID environment would offer support for distributed execution and resource allocation and adaptivity at the level of automated load balancing, for example \[83, 131, 161\]. Initial steps for integration of template-based reconfiguration with the workflow enabled toolkit TRIANA\[12\] \[89, 137\] are described in \[147\].

**Virtual Organisations.** If the concept of automated multi-level reconfiguration is extended to multiple autonomous multi-layered entities, it enters the realm of virtual organisations \[49\]. Considering adaptation of an organisation is interesting, as the management of the different layers will require reasoning with different concepts. The upper management layer reasons on high-level organisation goals, for example growth or increase of market share, whereas the lower layers handle work-based problems, on a regular workflow level. Propagation could be used to enable reasoning over lower layers. In addition, if multiple organisations are involved, integration of more formal contracts such as Service Level Agreements is required, using, for example, WSLA \[72\]. However, aside of the technical integration of service level agreements, also the legal integration remains a challenge, as the legal conditions and requirements still need to be identified for the use of web services \[18\].

A demanding application domain in the context of virtual organisations is crisis management. Reconfiguration of complex systems is best controllable if the environment is predictable, the possible actions are limited and repeatable, the system is homogeneous in organisation, or the system is centralised. The domain of crisis management \[138\], however, violates all of these conditions. Crises are highly dynamic and chaotic, therefore solutions differ not only from one crisis to another, but a solution has to be continuously adapted to changes in the situation. Since in general multiple autonomous

\[12\]TRIANA project page: [http://www.trianacode.org/](http://www.trianacode.org/)
 organisations are involved in dealing with a crisis, coordination of crisis re-
response units is inherently distributed. As different organisations are involved,
different policies for different services can be expected. Virtual multi-level
organisations in crisis management are not solely limited to artificial systems:
Actor-Agent Communities in which both humans and artificial systems co-
operate with shared goals as described in [158] can profit from the approach
presented in this thesis.

An initial mapping to the domain of crisis management is given in [152],
in which template-based reconfiguration is combined with SMDS [151] as an
alternative approach to composition and deployment.

**Overlay Networks for Adaptation.** Execution of complex workflows is
one of many applications of Peer-to-Peer (P2P) networks. Sustaining com-
plex workflows in a dynamic P2P network requires adaptation when hosts
involved in the workflow lose network connectivity. Adaptation of complex
workflows without centralised monitoring and control is a challenge, and is
often limited to instance replacement. If requirements are given on the per-
formance of the complex workflow as a whole, then possibilities for automated
adaptation are limited even more.

Research in [148] describes an initial mapping with which template-based
reconfiguration could be applied in a Peer to Peer environment, as adaptive
overlay networks, and extending work on P2P aggregation, such as in [111].
As each node within an overlay network can have individual requirements
and matching functions, template based configurations can perform dynamic
decentralised clustering [104], to create clusters of services and templates
which can be used to fill specific slots, or use coalition formation [2] to identify
possible useful services.

Note that this is different from Semantic Overlay Networks [42], in which
overlay networks are constructed to optimise answering semantic queries over
the different nodes. The overlay network is related to the content, and not
specifically to the Quality of Service of the subparts. An overlay structure
used to reason on Quality of Service is presented in OverQoS [82], however
the aim of OverQoS is to create an overview of the overall network, and
not to allow parts of the system to adapt. This point is addressed in [58],
where workflows can even be partially reconfigured to increase the QoS. How-
ever the adaptive overlay networks that could be enabled by using templates
could perform adaptation completely distributed, whereas the approach pre-
vented in [58] requires an overview of the complete workflow, thus acting as
a centralised reconfigurator.
7.3 Extensions to Other Environments and Application Domain

**Autonomic Computing.** Automated template-based reconfiguration is of interest from an autonomic computing [74] perspective, as our approach is related to some of the prime aspects of autonomic computing: self-configuration, self-healing, and optionally self-optimisation. Template-based reconfiguration is an enabler for self-adapting complex services [45].

Both the structure and the process allow interesting applications in autonomic computing. The multi-layered structure allows the specification and reasoning on dependencies on different levels, using propagation. The template-based reconfiguration process allows the specification of different policies. In a distributed setting of the reconfiguration process, the policies can be based on the requirements of the local hosts and organisations. For problem localisation in autonomic computing [73] the slots of a template-based structure can be used, and the scope can be gradually incremented using the policies as described in Focus Determination in Section 4.2.1. In combination with Service Level Agreements, template-based reconfiguration would offer a good basis to create a system based on autonomic computing.

To allow self-healing and self-optimisation, template-based reconfiguration needs to be extended with monitoring. A multi-layered approach to self-management is, for example, [61] offering a case-based reasoning approach for diagnosis and monitoring [62]. Combining this with template-based reconfiguration, would require templates to be annotated with case-based reasoning knowledge, describing the behaviour for monitoring.
Discussion and Future Research
List of Figures

1.1 Schematic overview of thesis. .............................................. 3

3.1 Web service architecture (adapted from [133]). ...................... 16
3.2 The OWL-S ontology ...................................................... 18
3.3 Hidden design rationale, lost at deployment time of a service. .... 19
3.4 Web service architecture (adapted from [133]), extended with a 'partial service configuration description' layer (3b). ......... 20
3.5 Example of a template description expressed in OWL-S. .......... 23
3.6 Schematic overview of strong completeness in a template-based web service configuration. The strongly complete components are grey, and the the components that are not strongly complete are white. The configuration as a whole is not strongly complete. ...................................................... 26
3.7 Propagation of average execution time in two steps. ............... 28

4.1 Overview of processes for the adaptation of web service configurations. ...................................................... 31
4.2 Activity diagram of the Reconfiguration Process .................... 32
4.3 Overview of the Reconfiguration Process ............................... 39
4.4 Example of an initial configuration with a failing web service. ... 40

5.1 Activity diagram for the Strict Admission template .................. 47
5.2 Activity diagram for the Strict Publication Classification Template ...................................................... 48
5.3 On the left the activity diagram of a single FilterStrictClasses service, and on the right the activity diagram of a configuration, using 4 services, offering an equivalent functionality. ... 49
5.4 Venn-diagram of the sets of classes filtered by the three separate services. The intersection of the sets results in the desired class set. ...................................................... 50
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>IBROW Structure of classification services. The rectangles represent input and output. The ovals represent the parameterised parts of the IBROW skeleton for the family of classification services (taken from [141]).</td>
</tr>
<tr>
<td>5.6</td>
<td>The reduction of the generic task model and an initial activity diagram for the Classification Template.</td>
</tr>
<tr>
<td>5.7</td>
<td>The initial configuration ( Conf_{int} ).</td>
</tr>
<tr>
<td>5.8</td>
<td>Overview of the trace of the example Classification Reconfiguration.</td>
</tr>
<tr>
<td>5.9</td>
<td>The first focus, limiting the area affected by reconfiguration.</td>
</tr>
<tr>
<td>5.10</td>
<td>The template found as a top component.</td>
</tr>
<tr>
<td>5.11</td>
<td>The pivot focuses on the first open slot.</td>
</tr>
<tr>
<td>5.12</td>
<td>The new configuration resulting from the Template-based Configuration process, based on the given set of requirements.</td>
</tr>
<tr>
<td>5.13</td>
<td>The configuration resulting from the Integration process.</td>
</tr>
<tr>
<td>6.1</td>
<td>Activity diagram for the Determine Area template.</td>
</tr>
<tr>
<td>6.2</td>
<td>Activity diagram for the Determine Height template.</td>
</tr>
<tr>
<td>6.3</td>
<td>Activity diagram for the first Determine Squared Height template.</td>
</tr>
<tr>
<td>6.4</td>
<td>Activity diagram for the second Determine Squared Height template.</td>
</tr>
<tr>
<td>6.5</td>
<td>Hierarchical view of the initial configuration.</td>
</tr>
<tr>
<td>6.6</td>
<td>The triangle of which the area needs to be determined. Both the ( Base (b) ) and the ( Hypotenuse (c) ) are known, but the ( Height (a) ) is unknown. The angle between ( a ) and ( b ) is 90°.</td>
</tr>
<tr>
<td>6.7</td>
<td>Overview of the trace of example Math reconfiguration.</td>
</tr>
<tr>
<td>6.8</td>
<td>The resulting first focus, limiting the area affected by reconfiguration to slot ( S_{t1}^{43.s3} ).</td>
</tr>
<tr>
<td>6.9</td>
<td>The second focus, changing the area affected by reconfiguration to the first slot of template ( t_2 ).</td>
</tr>
<tr>
<td>6.10</td>
<td>The template that is found as a top level component for the configuration to be inserted into slot ( S_{t2,s1}^{43} ).</td>
</tr>
<tr>
<td>6.11</td>
<td>The pivot for filling slots focuses on the second open slot.</td>
</tr>
<tr>
<td>6.12</td>
<td>Configuration ( Conf_{add2} ) resulting from the Template-based Configuration process, based on the given set of requirements.</td>
</tr>
<tr>
<td>6.13</td>
<td>Configuration ( Conf_{result} ) resulting from the Integration process.</td>
</tr>
<tr>
<td>C.1</td>
<td>Template-based configuration is a specific design process.</td>
</tr>
<tr>
<td>C.2</td>
<td>The role of the processes in the design process.</td>
</tr>
</tbody>
</table>
C.3 Overall Design Strategies for template-based configuration that DPC sends to RQSM and DODM. . . . . . . . . . . . . . . . 154
Bibliography


Part I

OWL-S Based Template Definitions
Appendix A

Math Template 1

This appendix contains an example specification based on the OWL-S of a template. The template described is the first template of the Math domain, as defined in Section 6.2.2.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uridef [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.2/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.2/Profile.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.2/Process.owl">
  <!ENTITY math "http://reliant.teknowledge.com/DAML/SUMO.owl">
  <!ENTITY template "http://www.cs.vu.nl/~sander/COWS/Templates.owl">
  <!ENTITY triangle "http://www.cs.vu.nl/~sander/COWS/mathDomain/Triangle.owl">
]>}

<rdf:RDF
  xmlns:rdf="&rdf;#"
  xmlns:rdfs="&rdfs;#"
  xmlns:owl="&owl;#"
  xmlns:xsd="&xsd;#"
  xmlns:service="&service;#"
  xmlns:profile="&profile;#"
  xmlns:process="&process;#"
  xmlns:math="&math;#"
  xmlns:template="&template;#"
  xmlns:triangle="&triangle;#"
The Service Declaration

Cette template determine the area of a right angled triangle.

The height of the triangle is unknown, the other two sides are known.

The template contains two slots
1. to determine the height of the triangle given the length of the other two sides,
2. to determine the area of the triangle based on the height and the base.

A textual description of the service

This template determines the area of a right angled triangle.

The height of the triangle is unknown, the other two sides are known.

The template contains two slots
1. to determine the height of the triangle given the length of the other two sides,
2. to determine the area of the triangle based on the height and the base.

A definition of the slots in this template
<template:hasSlot rdf:resource="#DetermineHeight"/>
<template:hasSlot rdf:resource="#DetermineArea"/>
</triangle:determineArea2>

<process:ProcessModel rdf:ID="template1Process">
  <process:hasProcess rdf:resource="#P1"/>
  <service:describes rdf:resource="#template1Profile"/>
</process:ProcessModel>

<template:Slot rdf:ID="DetermineHeightSlot">
  <template:textDescription>
    This slot should be filled with a component that determines the height of a triangle with one right angle, given the base and the hypotenuse of the triangle. The resulting precision in the output needs to be greater than or equal to 8.
  </template:textDescription>

  <template:requirementsList>
    <template:requirement rdf:ID="rq1">
      <!-- Require a template or service of a serviceCategory reflecting the required functionality -->
      <template:property>
        profile:serviceCategory
      </template:property>
      <template:property>
        triangle:DetermineHeight
      </template:property>
    </template:requirement>
    <template:requirement rdf:ID="rq2">
      <!-- Require that output has a precision of at least 8 -->
      <template:property>
        resultingPrecision
      </template:property>
      <math:greaterThanOrEqualTo>
      </math:greaterThanOrEqualTo>
    </template:requirement>
  </template:requirementsList>
</template:Slot>
8
</math:greaterThanOrEqualTo>
</template:requirement>
</template:requirementsList>

<!-- IO specification -->
<profile:input>
  <profile:ParameterDescription rdf:ID="BaseSlot1">
    <profile:parameterName>Base</profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#Base"/>
  </profile:ParameterDescription>
</profile:input>

<profile:input>
  <profile:ParameterDescription rdf:ID="HypotenuseSlot1">
    <profile:parameterName>Hypotenuse</profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#Hypotenuse"/>
  </profile:ParameterDescription>
</profile:input>

<profile:output>
  <profile:ParameterDescription rdf:ID="HeightSlot1">
    <profile:parameterName>Height</profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#Height"/>
  </profile:ParameterDescription>
</profile:output>
</template:Slot>

<!-- Slot definitions (second slot) -->
<!-- -->

<template:Slot rdf:ID="DetermineArea">
  <template:textDescription>
    This slot should be filled with a component that determines the height of a right-angled triangle, given the base and height of the triangle. The resulting precision in the output needs to be greater than or equal to 8.
  </template:textDescription>
</template:Slot>

<template:requirementsList>
  <template:requirement rdf:ID="rq3">
    <!-- Require a template or service of a serviceCategory reflecting the required functionality -->
  </template:requirement>
</template:requirementsList>
<template:property>
  profile:serviceCategory
</template:property>
<template:value>
  triangle:DetermineArea
</template:value>
</template:requirement>
<template:requirement rdf:ID="rq4">
  <!-- Require that output has a precision of at least 8 -->
  <template:property>
    resultingPrecision
  </template:property>
  <math:greaterThanOrEqualTo>8</math:greaterThanOrEqualTo>
</template:requirement>
</template:requirementsList>

<!-- IO specification -->
<profile:input>
  <profile:ParameterDescription rdf:ID="BaseSlot2">
    <profile:paramName>Base</profile:paramName>
    <profile:restrictedTo rdf:resource="&triangle;#Base"/>
  </profile:ParameterDescription>
</profile:input>

<profile:input>
  <profile:ParameterDescription rdf:ID="HeightSlot2">
    <profile:paramName>Height</profile:paramName>
    <profile:restrictedTo rdf:resource="&triangle;#Height"/>
  </profile:ParameterDescription>
</profile:input>

<profile:output>
  <profile:ParameterDescription rdf:ID="AreaSlot2">
    <profile:paramName>Area</profile:paramName>
    <profile:restrictedTo rdf:resource="&triangle;#Area"/>
  </profile:ParameterDescription>
</profile:output>
</template:Slot>

<!-- Definition of top level Process as a composed process -->
<owl:Class rdf:ID="P1">
  <rdfs:subClassOf rdf:resource="&process;#CompositeProcess"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&process;#composedOf"/>
      <owl:toClass>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection"/>
<process:hasDataFrom>
  <process:InputBinding>
    <process:toParam rdf:resource="#BaseSlot2"/>
    <process:valueSource>
      <process:ValueOf>
        <process:theVar rdf:resource="#Base"/>
        <process:fromProcess
          rdf:resource="#TheParentPerform"/>
      </process:ValueOf>
      <process:valueSource>
        <process:InputBinding>
          <process:InputBinding>
            <process:toParam
              rdf:resource="#HeightSlot2"/>
          </process:InputBinding>
          <process:valueSource>
            <process:ValueOf>
              <process:theVar rdf:resource="HeightSlot1"/>
              <process:fromProcess
                rdf:resource="#PerformDetermineHeightSlot"/>
            </process:ValueOf>
            <process:valueSource>
              <process:InputBinding>
                <process:hasDataFrom>
                  <process:Perform>
                    <process:listOfInstancesOf>
                      <owl:Class>
                        <owl:toClass>
                          <owl:Restriction>
                            <owl:intersectionOf>
                              <owl:Restriction>
                                <owl:toClass>
                              </owl:Restriction>
                            </owl:intersectionOf>
                          </owl:toClass>
                        </owl:Class>
                      </owl:listOfInstancesOf>
                    </process:Perform>
                  </process:hasResult>
                </process:hasResult>
              </process:InputBinding>
            </process:valueSource>
          </process:InputBinding>
        </process:valueSource>
      </process:InputBinding>
      <process:ValueOf>
        <process:theVar rdf:resource="#AreaSlot2"/>
        <process:fromProcess rdf:resource="#Area"/>
      </process:ValueOf>
    </process:valueSource>
  </process:InputBinding>
</process:hasDataFrom>
Perform\texttt{DetermineAreaSlot} />
</process:valueOf>
</process:valueSource>
</process:OutputBinding>
</process:withOutput>
</process:Result>
</process:hasResult>
</owl:Class>

<!— IO specification -->
<profile:input>
<profile:ParameterDescription rdf:ID="Base">
  <profile:parameterName>Base</profile:parameterName>
  <profile:restrictedTo rdf:resource="&triangle;#Base"/>
</profile:ParameterDescription>
</profile:input>
<profile:input>
<profile:ParameterDescription rdf:ID="Hypotenuse">
  <profile:parameterName>Hypotenuse</profile:parameterName>
  <profile:restrictedTo rdf:resource="&triangle;#Hypotenuse"/>
</profile:ParameterDescription>
</profile:input>
<profile:output>
<profile:ParameterDescription rdf:ID="Area">
  <profile:parameterName>Area</profile:parameterName>
  <profile:restrictedTo rdf:resource="&triangle;#Area"/>
</profile:ParameterDescription>
</profile:output>
</template:Slot>
Appendix B
Math Template 4

This appendix contains an example specification based on the OWL-S of a template. The template described is the fourth template of the Math domain, as defined in Section 6.2.2. In contrast to the previous example, this definition includes references to a propagation function.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uridef [ 
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns"> 
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema"> 
  <!ENTITY owl "http://www.w3.org/2002/07/owl"> 
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema"> 
  <!ENTITY service "http://www.daml.org/services/owl-s/1.2/Service.owl"> 
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.2/Profile.owl"> 
  <!ENTITY process "http://www.daml.org/services/owl-s/1.2/Process.owl"> 
  <!ENTITY math "http://reliant.teknowledge.com/DAML/SUMO.owl"> 
  <!ENTITY template "http://www.cs.vu.nl/~sander/COWS/Template.owl"> 
  <!ENTITY triangle "http://www.cs.vu.nl/~sander/COWS/mathDomain/Triangle.owl"> 
]> 

<rdf:RDF 
  xmlns:rdf="&rdf;#" 
  xmlns:rdfs="&rdfs;#" 
  xmlns:owl="&owl;#" 
  xmlns:xsd="&xsd;#" 
  xmlns:service="&service;#" 
  xmlns:profile="&profile;#" 
  xmlns:process="&process;#" 
  xmlns:math="&math;#" 
>
This example template demonstrates several possibilities to support propagated properties:
1) pose requirements in slots on properties that need to be propagated
2) refer to a propagation function to determine a property for the overall template (the property PreservesInformation)

---

---

The Service Declaration

---

<service:Service rdf:ID="MathTemplate4Service">  
<!-- Reference to the Profile -->  
<service:presents rdf:resource="#MathTemplate4Profile"/>  
<!-- Reference to the Process Model -->  
<service:describedBy rdf:resource="#MathTemplate4Process"/>  
</service:Service>

---

The Service Profile Declaration

---

<triangle:DetermineSquaredHeight rdf:ID="MathTemplate4Profile">  
<rdfs:subClassOf rdf:resource="&template;#Template"/>  
<!-- reference to the service specification -->  
<service:describedBy rdf:resource="#MathTemplate4Service"/>  
<!-- reference to the process model specification, P1 -->  
<profile:has_process rdf:resource="#P1"/>  
<!-- define the name of the service -->  
<profile:serviceName>  
  t4_Determine_Squared_Height  
</profile:serviceName>  
<!-- A textual description of the service -->  
<profile:textDescription>  
  This template determines the squared height of a right
angled triangle.
This template has 3 slots for components (ie services or templates)
capable of addition, subtraction and multiplication, to equate
equation \( c_2 - b_2 = (c+b) \ast (c-b) \).
The template contains three slots:
1 Add two numbers
2 Subtract two numbers
3 multiply two numbers.
</profile:textDescription>

<!— a definition of the slots in this template -->
<template:hasSlot rdf:resource="#Add"/>
<template:hasSlot rdf:resource="#Subtract"/>
<template:hasSlot rdf:resource="#Multiply"/>
</triangle:DetermineSquaredHeight>

<!—

*************************************************************************
* The Service Process Declaration *
*************************************************************************

-->

<process:ProcessModel rdf:ID="MathTemplate4Process">
  <process:hasProcess rdf:resource="#P1"/>
  <service:describes rdf:resource="#MathTemplate4Profile"/>
</process:ProcessModel>

<!— Slot definitions -->
<!—---------------------------------------------------------------------- -->
<template:Slot rdf:ID="Add">
  <template:textDescription>
    This slot should be filled with a component (ie services
    or template)
capable of adding two numbers.
  </template:textDescription>

  <template:requirementsList>
    <template:requirement rdf:ID="rq1">
      <!— Require a template or service of a serviceCategory reflecting
the required functionality -->
      <template:property>
        profile:serviceCategory
      </template:property>
    </template:requirement>
  </template:requirementsList>
</template:Slot>
<template:value>
  triangle: Add
</template:value>
</template:requirement>
<template:requirement rdf:ID="rq2">
  <!-- Require the precision of the output is at least equal or greater than the yield of this slot -->
  <template:property>
    triangle: Precision
  </template:property>
  <math:greaterThanOrEqualTo>
    <rdf:ID="#YieldSlot1"/>
  </math:greaterThanOrEqualTo>
</template:requirement>
</template:requirementsList>

<!-- IO specification -->
<profile:input>
  <profile:ParameterDescription rdf:ID="HypotenuseSlot1">
    <profile:parameterName> Hypotenuse </profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#Hypotenuse"/>
  </profile:ParameterDescription>
</profile:input>
<profile:input>
  <profile:ParameterDescription rdf:ID="BaseSlot1">
    <profile:parameterName> Base </profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#Base"/>
  </profile:ParameterDescription>
</profile:input>
<profile:output>
  <profile:ParameterDescription rdf:ID="AddedHypothenuseAndBaseSlot1">
    <profile:parameterName> AddedHypothenuseAndBase </profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#AddedHypothenuseAndBase"/>
  </profile:ParameterDescription>
</profile:output>

<!-- Yield function Slot 1 -->
<template:propagatedProperty rdf:ID="YieldSlot1">
  <!-- Refers to the yield generated by the component inserted in this slot. The resulting property that needs to be generated by the property -->
</template:propagatedProperty>
propagation function: the yield –>
<propagation:propertyType rdf:resource="&triangle;#Yield"/>

<!— pointers to the inputs for the propagation function —->
<propagation:PropagationFunction>
<propagation:Input rdf:ID="propInput1">
<!— first input: the accuracy of the hypotenuse —->
<propagation:hasType rdf:about="&triangle;#Accuracy"/>
<ofInput rdf:about="#HypotenuseSlot1"/>
</propagation:Input>
<propagation:Input rdf:ID="propInput2">
<!— second input: the accuracy of the base —->
<propagation:hasType rdf:about="&triangle;#Accuracy"/>
<ofInput rdf:about="#BaseSlot1"/>
</propagation:Input>
<propagation:hasFunctionHandler>
<rdf:resource="&triangle;#DetermineYield"/>
</propagation:hasFunctionHandler>
</propagation:PropagationFunction>
</template:propagatedProperty>

<!— Slot definitions (second slot) —->
<!— ________________________________________ —->

<template:Slot rdf:ID="Subtract">
<template:textDescription>
This slot should be filled with a component (ie service or template)
capable of subtracting two numbers.
</template:textDescription>

<template:requirementsList>
<template:requirement rdf:ID="rq3">
<!— Require a template or service of a serviceCategory reflecting
the required functionality —->
<template:property>
profile:serviceCategory
</template:property>
<template:value>
triangle:Subtract
</template:value>
</template:requirement>
<template:requirement rdf:ID="rq4">

<!— Require the precision of the output is at least equal or greater than the yield of this slot —>
<template:property>
  triangle:Precision
</template:property>
<math:greaterThanOrEqualTo>
  <rdf:ID="#YieldSlot2"/>
</math:greaterThanOrEqualTo>
</template:requirement>
</template:requirementsList>

<!— IO specification —>
<profile:input>
  <profile:ParameterDescription rdf:ID="HypotenuseSlot2">
    <profile:parameterName> Hypotenuse </profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#Hypotenuse"/>
  </profile:ParameterDescription>
</profile:input>
<profile:input>
  <profile:ParameterDescription rdf:ID="BaseSlot2">
    <profile:parameterName> Base </profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#Base"/>
  </profile:ParameterDescription>
</profile:input>
<profile:output>
  <profile:ParameterDescription rdf:ID="SubtractedHypothenuseBaseSlot2">
    <profile:parameterName> SubtractedHypothenuseBase </profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#SubtractedHypothenuseBase"/>
  </profile:ParameterDescription>
</profile:output>

<!— Yield function Slot 2 —>
<template:propagatedProperty rdf:ID="YieldSlot2">
  <!— Refers to the yield generated by the component inserted in this slot. The resulting property that needs to be generated by the property propagation function: the yield —>
  <propagation:propertyType rdf:resource="&triangle;#Yield"/>

  <!— pointers to the inputs for the propagation function —>
  <propagation:PropagationFunction>
<propagation:Input rdf:ID="propInput1Slot2">
  <!-- first input: the accuracy of the hypotenuse -->
  <propagation:hasType rdf:about="&triangle;#Accuracy"/>
  <ofInput rdf:about="#HypotenuseSlot2"/>
</propagation:Input>

<propagation:Input rdf:ID="propInput2Slot2">
  <!-- second input: the accuracy of the base -->
  <propagation:hasType rdf:about="&triangle;#Accuracy"/>
  <ofInput rdf:about="#BaseSlot2"/>
</propagation:Input>

<propagation:hasFunctionHandler>
  <rdf:resource="&triangle;#DetermineYield"/>
</propagation:hasFunctionHandler>
</propagation:PropagationFunction>
</template:propagatedProperty>

<!-- Slot definitions (third slot) -->

<template:Slot rdf:ID="Multiply">
  <template:textDescription>
    This slot should be filled with a component (ie service or template) capable of multiplying two numbers.
  </template:textDescription>

<template:requirementsList>
  <template:requirement rdf:ID="rq5">
    <!-- Require a template or service of a serviceCategory reflecting the required functionality -->
    <template:property>
      profile:serviceCategory
    </template:property>
    <template:value>
      triangle:Multiply
    </template:value>
  </template:requirement>
  <template:requirement rdf:ID="rq6">
    <!-- Require the precision of the output is at least equal or greater than the yield of this slot -->
    <template:property>
      triangle:Precision
    </template:property>
  </template:requirement>
</template:requirementsList>
<math:greaterThanOrEqualTo>
  <rdf:ID="#YieldSlot3"/>
</math:greaterThanOrEqualTo>
</template:requirement>
</template:requirementsList>

<!-- IO specification -->
<profile:input>
  <profile:ParameterDescription rdf:ID="AddedHypothenuseAndBaseSlot3">
    <profile:parameterName>AddedHypothenuseAndBase</profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#AddedHypothenuseAndBase"/>
  </profile:ParameterDescription>
</profile:input>

<profile:input>
  <profile:ParameterDescription rdf:ID="SubtractedHypothenuseBaseSlot3">
    <profile:parameterName>SubtractedHypothenuseBase</profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#SubtractedHypothenuseBase"/>
  </profile:ParameterDescription>
</profile:input>

<profile:output>
  <profile:ParameterDescription rdf:ID="SquaredHeightSlot3">
    <profile:parameterName>SquaredHeight</profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#SquaredHeight"/>
  </profile:ParameterDescription>
</profile:output>

<!-- Yield function Slot 3 -->
<template:propagatedProperty rdf:ID="YieldSlot3">
  <!-- Refers to the yield generated by the component inserted in this slot. The resulting property that needs to be generated by the property propagation function: the yield -->
  <propagation:propertyType rdf:resource="&triangle;#Yield"/>

  <!-- pointers to the inputs for the propagation function -->
  <propagation:PropagationFunction>
    <propagation:Input rdf:ID="propInput1Slot3">
      <!-- first input: the accuracy of the hypothenuse -->
      <propagation:hasType rdf:about="&triangle;#Yield"/>
<ofInput rdf:about="#YieldSlot1"/>
</propagation:Input>
<propagation:Input rdf:ID="propInput2Slot3"
    <!— second input: the accuracy of the base —->
    <propagation:hasType rdf:about="&triangle;#Yield"/>
<ofInput rdf:about="#YieldSlot2"/>
</propagation:Input>
<propagation:hasFunctionHandler>
    <rdf:resource="&triangle;#DetermineYield"/>
</propagation:hasFunctionHandler>
</template:PropagationFunction>
</template:propagatedProperty>
</template:Slot>

<!— Definition of top level Process as a composed process —->
<!— Defined as a sequenceOf(Split−Join(slot1, slot2), slot3) —->
<owl:Class rdf:ID="P1">
    <rdfs:subClassOf rdf:resource="&process;#CompositeProcess"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&process;#composedOf"/>
            <owl:toClass>
                <owl:Class>
                    <owl:intersectionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="process:Sequence"/>
                        <owl:Restriction>
                            <owl:onProperty rdf:resource="&process;#components"/>
                        </owl:Restriction>
                        <owl:toClass>
                            <owl:Class>
                                <owl:intersectionOf rdf:parseType="Collection">
                                    <owl:Class rdf:about="process:Split−Join"/>
                                    <owl:Restriction>
                                        <owl:onProperty rdf:resource="&process;#components"/>
                                    </owl:Restriction>
                                    <owl:toClass>
                                        <owl:Class>
                                            <process:listOfInstancesOf rdf:parseType="Collection">
                                                <!— Activate the first slot —->
                                                <process:Perform rdf:ID="PerformAddSlot">
                                                    <process:process rdf:resource="#Add"/>
                                                </process:Perform>
                                                <process:hasDataFrom>
                                                    <process:InputBinding>
                                                        <process:toParam
                                                            rdf:resource="&hypotenuse;Slot1"/>
                                                    </process:InputBinding>
                                                </process:hasDataFrom>
                                            </process:listOfInstancesOf>
                                        </owl:Class>
                                    </owl:toClass>
                                </owl:Class>
                            </owl:toClass>
                        </owl:Class>
                    </owl:intersectionOf>
                </owl:Class>
            </owl:toClass>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<process:valueSource>
  <process:ValueOf>
    <process:theVar rdf:resource="Hypotenuse"/>
  </process:ValueOf>
  <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
</process:valueSource>
</process:InputBinding>
<process:InputBinding>
  <process:toParam rdf:resource="#BaseSlot1"/>
  <process:valueSource>
    <process:ValueOf>
      <process:theVar rdf:resource="Base"/>
      <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
    </process:ValueOf>
  </process:valueSource>
</process:InputBinding>
</process:valueSource>
<!-- Activate the second slot -->
<process:Perform rdf:ID="PerformSubtract">
  <process:process rdf:resource="#Subtract"/>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:toParam rdf:resource="#HypotenuseSlot2"/>
    </process:InputBinding>
    <process:valueSource>
      <process:ValueOf>
        <process:theVar rdf:resource="Hypotenuse"/>
        <process:fromProcess rdf:resource="&process;#TheParentPerform"/>
      </process:ValueOf>
    </process:valueSource>
  </process:hasDataFrom>
</process:Perform>
<process:InputBinding>
  <process:toParam rdf:resource="#BaseSlot2"/>
<owl:Class rdf:ID="Square"/>
<owl:Restriction rdf:ID="Square Restriction"/>
</owl:Class>

<owl:Class rdf:ID="Hypotenuse"/>
<owl:Restriction rdf:ID="Hypotenuse Restriction"/>
</owl:Class>

<owl:Class rdf:ID="Base"/>
<owl:Restriction rdf:ID="Base Restriction"/>
</owl:Class>

<owl:Class rdf:ID="AlwaysTrue"/>
<owl:Restriction rdf:ID="AlwaysTrue Restriction"/>
</owl:Class>

<owl:Class rdf:ID="SquaredHeight"/>
<owl:Restriction rdf:ID="SquaredHeight Restriction"/>
</owl:Class>

<owl:Class rdf:ID="SquaredHeightSlot3"/>
<owl:Restriction rdf:ID="SquaredHeightSlot3 Restriction"/>
</owl:Class>

<owl:Class rdf:ID="PerformMultiply"/>
<owl:Restriction rdf:ID="PerformMultiply Restriction"/>
</owl:Class>

<owl:Class rdf:ID="SquaredHeightSlot3 output"/>
<owl:Restriction rdf:ID="SquaredHeightSlot3 output Restriction"/>
</owl:Class>

<!—— IO specification ——>

<profile:input>
  <profile:ParameterDescription rdf:ID="Hypotenuse">
    <profile:parameterName> Hypotenuse </profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#Hypotenuse"/>
  </profile:ParameterDescription>
</profile:input>

<profile:input>
  <profile:ParameterDescription rdf:ID="Base">
    <profile:parameterName> Base </profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#Base"/>
  </profile:ParameterDescription>
</profile:input>
<profile:input>
<profile:output>
  <profile:ParameterDescription rdf:ID="SquaredHeight">
    <profile:parameterName> SquaredHeight </profile:parameterName>
    <profile:restrictedTo rdf:resource="&triangle;#SquaredHeight"/>
  </profile:ParameterDescription>
</profile:output>

<!— Yield function PreserveInformation for overall template —>
<template:propagatedProperty rdf:ID="PreservesInformationFunction">
  <!— Refer to the property PreservesInformation. —>
  <propagation:propertyType rdf:resource="&triangle;#PreservesInformation"/>
  <!— pointers to the inputs for the propagation function —>
  <propagation:Input rdf:ID="PreserveInfoInput1">
    <!— first input: the precision of slot 1 —>
    <propagation:hasSlot="#Add"/>
    <propagation:hasType rdf:about="&triangle;#Precision"/>
  </propagation:Input>
  <propagation:Input rdf:ID="PreserveInfoInput2">
    <!— first input: the yield of slot 1 —>
    <propagation:hasSlot="#Add"/>
    <propagation:hasType rdf:about="&triangle;#Yield"/>
  </propagation:Input>
  <propagation:Input rdf:ID="PreserveInfoInput3">
    <!— first input: the precision of slot 2 —>
    <propagation:hasSlot="#Subtract"/>
    <propagation:hasType rdf:about="&triangle;#Precision"/>
  </propagation:Input>
  <propagation:Input rdf:ID="PreserveInfoInput4">
    <!— first input: the yield of slot 2 —>
    <propagation:hasSlot="#Subtract"/>
    <propagation:hasType rdf:about="&triangle;#Yield"/>
  </propagation:Input>
  <propagation:Input rdf:ID="PreserveInfoInput5">
    <!— first input: the precision of slot 3 —>
    <propagation:hasSlot="#Multiply"/>
    <propagation:hasType rdf:about="&triangle;#Precision"/>
  </propagation:Input>
  <propagation:Input rdf:ID="PreserveInfoInput6"/>
<!— first input: the yield of slot 2 —>
<propagation:hasSlot="#Multiply"/>
<propagation:hasType rdf:about="#Yield"/>
</propagation:Input>

<!— refer to an external function for t4 —>
<!— this function returns true if for all slots the precision is greater than or equal to the yield —>
<propagation:hasFunctionHandler>
  <rdf:resource="#PreservationInformationT4"/>
</propagation:hasFunctionHandler>
</propagation:PropagationFunction>
</template:propagatedProperty>
Part II

Model of Configuration
Appendix C

A Template-Based Configuration Process

This appendix describes the template-based configuration process, used in the prototype described in Chapter 4. As template-based configuration is a specific design process, depicted in Figure C.1, the prototype implementation presented in this thesis is based on a generic model for design. The model used is the Generic Design Model [22, 145, 146]. Section C.1 describes the Generic Design Model, and Section C.2 describes how this model is extended for configuration, and more specifically for template-based configuration.

Figure C.1: Template-based configuration is a specific design process.

C.1 The Generic Design Model

The Generic Design Model is a task- and knowledge model in which the generic elements of a design process are abstracted from the more domain-
specific elements. The Generic Design Model described in this section is a model created using the DESIRE [19] framework. A detailed description of the Generic Design Model is given in [20, 24, 145, 146]. A refinement of the Generic Design Model for compositional design is presented in [157]. A refinement of the model for compositional design that integrates the use of building blocks is given in [98].

The Generic Design Model distinguishes three direct sub processes of the design process: design process co-ordination (DPC), design object description manipulation (DODM) and requirement qualification set manipulation (RQSM). The contribution of each of the three sub processes to the design process is illustrated in Figure C.2. The figure distinguishes three parts of the design process when viewed as a design space: one process responsible for exploring the RQS solution space, another process responsible for exploring the DOD search space, and a separate process responsible for deciding in what space the next exploration step should occur. Relating this to web service configuration, the DOD search space is the search space of all possible web service configurations, and the RQS search space is the search space of all possible requirements for these service configurations.

In the generic model for design the process Requirement Qualification Set Manipulation (RQSM), discussed in Section C.1.2, is responsible for exploring the Requirement Qualification Set (RQS) space, the process Design Object Description Manipulation (DODM), discussed in Section C.1.3, for the Design Object Description (DOD) space, and Design Process Control.
(DPC), discussed in Section C.1.1, is responsible for determining the following step in the overall design process. The following paragraphs discuss the inputs and outputs of the processes, as these are present in the traces in the Appendices I and J in the parts on template-based configuration of the reconfiguration examples discussed in Sections 5.3 and 6.3 respectively.

C.1.1 Design Process Co-ordination

The task Design Process Co-ordination (DPC) is responsible for determining the overall design strategy. DPC uses this information to determine the next step in the design process. DPC monitors the information coming from RQSM and DODM to determine if the overall design strategy is still successful. It also monitors if the design process is still satisfying the overall design objectives. When DPC has determined what the next course of action should be, it communicates this information to RQSM and DODM by providing them the current overall design strategy it has determined.

DPC receives three different types of information on its input:

- design process objectives contains information regarding the design as a whole, such as time constraints. This information is provided by the process or human using the design process.

- RQSM process evaluation contains evaluations of the RQSM process, such as whether the current overall design strategy has been fulfilled or not.

- DODM process evaluation contains information evaluations of the DODM process, similar to RQSM process evaluation.

The following types of information are in the output of the DPC process:

- design process evaluation Contains evaluations of the design process as a whole, such as whether the design process has failed or if it is still running, and evaluations of the initial design process objectives.

- overall design strategy to DODM Contains the information on the overall design strategy, such as a strategy indicating that the next modification step should be performed by RQSM.

- overall design strategy to RQSM Contains information similar to overall design strategy to DODM.
C.1.2 Requirement Qualification Set Manipulation

The task Requirement Qualification Set Manipulation (RQSM) is responsible for exploring the space of requirement qualification sets to find the most suitable, acceptable set. RQSM bases its decisions on the overall design strategy received from DPC and the assessments it receives from DODM.

RQSM has the following types of information on its input interface:

- Overall design strategy to RQSM contains overall design process strategy information, e.g. keep close to the given initial RQS.

- Initial RQS contains the initial requirement qualification set.

- Intermediate DOD assessment contains evaluations of the current design object description with regard to the current qualified requirements set. In this assessment the DODM can state how well it was possible to satisfy the currently given requirements given by RQSM.

The initial RQS is the qualified requirement set that the design process as a whole must try to satisfy. Within the limits indicated by the current overall design strategy, the initial RQS may be modified to find a set for which a design object description can be found.

The following types of information are in the output of RQSM:

- RQSM process evaluation contains evaluations on the progress of the RQSM process, such as information indicating if RQSM was able to find a new RQS modification.

- Intermediate RQS contains the RQS currently being examined in the design process.

- RQS assessment contains evaluations of the requirement qualification sets, e.g., why changes have been made, or whether requirements are conflicting.

- RQS contains the requirement qualification set itself. Together with RQS assessment these form the final qualified requirements, which are one of the outputs of the design process.

C.1.3 Design Object Description Manipulation

The task Design Object Description Manipulation (DODM) is responsible for exploring the space of design object descriptions. The main goal of DODM
C.2 Reusing the Generic Design Model for Template-based Configuration

is to find a DOD that is valid with respect to the requirement qualification set currently under examination.

The following information types are distinguished on the input interface of the process:

- overall design strategy to DODM contains overall design process strategy information.

- intermediate RQS contains the requirement qualification set for which a valid DOD has to be constructed.

- initial DOD contains an initial design object description, for example when re-designing an initial configuration.

DODM produces four types of information in its output:

- DODM process evaluation contains evaluations on the progress of the DODM process, such as information about the success of the modification process.

- DOD assessment contains information on the satisfaction of qualified requirements by a DOD.

- Intermediate DOD assessment contains information similar to DOD assessment, only this information is used as feedback for RQSM.

- DOD contains a design object description itself. Together with DOD assessment these form the final artefact description, which is one of the outputs of the design process.

C.2 Reusing the Generic Design Model for Template-based Configuration

This section describes how the Generic Design Model has been refined and specialised for template-based configuration. A number of strategies, policies and mechanisms have been added: strategies, policies and mechanisms that are generic for template-based configuration. The following subsections discuss the extension of the main components: Section [C.2.1] describes the changes for DPC, Section [C.2.2] describes the changes for RQSM, and finally Section [C.2.3] describes the changes for DODM.
C.2.1 Design Process Coordination Extended

Design Process Coordination (DPC) uses strategic knowledge to guide the design process in general. This includes knowledge about which search space is to be traversed next and when to stop the design process. The strategic knowledge inside DPC is used to determine a general design strategy on which Design Object Description Manipulation (DODM) and Requirement Qualification Set Manipulation (RQSM) base their internal strategy.

DPC can determine its internal strategy on the basis of knowledge on the previous states of DPC, process objectives, information DPC receives from RQSM and from DODM. RQSM and DODM will choose their own local strategy, which is linked to policies, to determine how to fulfil the overall strategy given by DPC. The overall design strategy, and the local design strategies of RQSM and DODM are therefore heavily dependent on each other.

Figure C.3: Overall Design Strategies for template-based configuration that DPC sends to RQSM and DODM.

The specific overall strategies that are used for template-based configuration are depicted in Figure C.3. Initially a main component needs to be found, after which the component needs to be refined until strong completeness, and then it must check whether all requirements have been satisfied,
including the requirements on propagated properties. If a strategy cannot be successfully completed by either RQSM or DODM, the previous overall strategy is selected again. If a main component cannot be determined, then the configuration process ends with a failure to configure, as no configuration could be found to satisfy the given requirements.

The finish conditions, as described in Section 4.2.3, are met if the design is successful, as a successful completion of the strategy refine and complete results in a strongly complete configuration, and the successful completion of the strategy check pre- and post-conditions ensures that the strongly complete configuration is also correct and meets all initial requirements.

**C.2.2 Requirement Qualification Set Manipulation Extended**

The reasoning on requirements includes handling the overall strategy, simple addition and removal of requirements, a preference for selecting requirements for enactment, and including requirements based on the feedback on the designed configuration (to include requirements resulting from slots).

First the composition of RQSM [145] is given, after which the specific extensions are presented:

- RQS modification
  - RQS modification analysis
    * RQS assessment
      - design requirement assessment preparation
      - design requirement assessment
      - design requirement assessment completion
    - overall RQS assessment
      * RQS modification evaluation
      * RQSM process evaluation
  - RQS modification determination
    * RQS modification focus determination
    * RQS modification method determination
    * RQS modification method execution
- RQSM history maintenance
- deductive RQS refinement
A Template-Based Configuration Process

- current RQS maintenance

*RQS modification analysis* is extended to reason in relation to the local and global strategies for template-based configuration. Within this component *overall RQS assessment* specifically extended. Overall RQS assessment contains a mechanism to include feedback from DODM into its new requirements assessment. For this overall DOD assessments are used. This component is extended to allow the generation of requirements, in RQS modification determination, based on slot descriptions of newly added open slots in a configuration created by DODM. As an extension to the original concept, the overall RQS assessment maintains a reference to:

- the configuration in which contains the open slot
- the slot within the configuration
- the slot description containing the slot specification

RQS modification determination is specialised for template-based configuration. *RQS modification focus determination* needs domain specific information for strategies to select the focus of the requirements. Foci are specified for functional, non-functional requirements, and requirements resulting from DOD assessments. These foci are related to local strategies of RQSM, which are linked to the overall design strategy. *RQS modification method determination* and *RQS modification method execution* are limited to adding and removing requirements which are in focus, and relating requirements to specific slots in the configuration. For relating requirements to slots to requirements the results from overall RQS assessment are used. The RQS resulting of the modification method execution states which requirements are to be enacted, thus which need to be satisfied by DODM.

C.2.3 Design Object Description Manipulation Extended

The reasoning on the design object description includes handling the overall strategy, adding and removing components (either web service or template) to or from a configuration, a preference for selecting these components, and assessing the satisfaction of requirements in the given RQS, deducing properties using propagation function, handling newly added slots.

First the composition of DODM [145] is given, after which the specific extensions are presented:

- DODM modification
DOD modification analysis

* DOD assessment
  - basic DOD assessment preparation
  - **basic DOD assessment**
  - basic DOD assessment completion
  - overall DOD assessment
* DOD modification evaluation
* DODM process evaluation

DOD modification determination

* DOD modification focus determination
* DOD modification method determination
* DOD modification method execution

DOD assessment is extended to support template-based configuration. More specific, basic DOD assessment requires knowledge on the checking requirements on a DOD. For template based web service configuration, relating requirements on functionality, input and output, and on other properties to properties of the service configuration is added. As the requirements are related to specific slots, each requirement is assessed against the properties of the partial configuration inserted the slot for which the requirement holds. In addition assessing requirements from the RQS, also feedback is generated for RQSM in specifying the available slots within the assessed DOD. This feedback is used by RQSM to generate requirements for newly added slots.

DOD modification determination is extended to support template-based configuration. All of its subcomponents are extended. DOD modification focus determination has different structural foci for template-based configuration: the main component or a specific open slot. DOD modification method determination and DOD modification method execution allows for the filling, and emptying of slots and the position of the main component. These three components together implement the local policies, related to the global strategies.

For the global overall strategy ‘select main component’, the focus is on the position of the main component, and a local generic policy is activated to prefer web services to fulfill the main component. If no matching web service
is are found, the local generic policy is changed to selecting a template as a main component.

For the global overall strategy ‘refine and complete’ an open slot is selected. The implemented generic policy is to select the first encountered open slot. To fill this open slot a component is selected, for which the policy is to prefer a web service over a template that does not violate the requirements related to this slot. Every activation of DODM only one slot is filled, to allow inclusion of new slot requirements. The overall strategy succeeds and finishes if the resulting configuration is strongly complete and partially correct.

For the global overall strategy ‘check pre-postcondition’ the DOD is not modified using modification determination, but deductive refinement is used to activate propagate all properties, after which the resulting DOD is assessed on all requirements. The DOD needs to be correct to successfully complete this overall strategy.

In addition to this, a backtracking mechanism is included, that, in case of a failure of successful completion of an overall strategy, removes the component that was lastly added.

**Deductive DOD refinement** extended to perform propagation included in the templates of the current configuration. The implementation calls a recursive function for propagation:

```java
// property propagation must be performed bottom-up
// first going down all slots of the template
for (Iterator i = slots.iterator(); i.hasNext();)
    Slot slot = (Slot)i.next();
    BuildingBlock filler = slot.getFiller();
    // check if slot is filled (normally it has been filled)
    if (filler != null) {
        filler.propagateProperties();
    }

for (Iterator j = propPropagationFunctions.iterator(); j.hasNext();)
    PropertyPropagationFunction function = (PropertyPropagationFunction)j.next();
    MyProperty property = function.determineProperty(this);
    propagatedProperties.add(property);
```

Furthermore deductive DOD refinement is used to determine whether partial configurations within the DOD are strongly complete, incorrect, partially correct, or correct. These aspects have been discussed in Section 3.3.
Part III

Code
Appendix D

Overall Reconfiguration Process

```java
package versie_2.component;

import inProgress.DODInstance;
import java.util.Collection;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.logging.Logger;
import test.ExperimentHandler;
import versie_2.informationTypes.AdditionalConfigurationException;
import versie_2.informationTypes.FailingWebService;
import versie_2.informationTypes.FailingWebServiceInterface;
import versie_2.informationTypes.FocusInterface;
import versie_2.informationTypes.InitialConfiguration;
import versie_2.informationTypes.InitialConfigurationInterface;
import versie_2.informationTypes.NewConfigurationInterface;
import versie_2.informationTypes.Repositories;
import versie_2.informationTypes.RepositoriesInterface;
import versie_2.informationTypes.RequirementSetInterface;

public class ReconfigurationProcess implements ReconfigurationInterface {

    /** the log used to log all information */
    protected Logger log = Logger.getLogger(this.getClass().
        getName());

    /**
    *
    */
```
counter to keep track of the configuration tickets that have already been issued. Note that this counter is only for a single session, there is no persistent storage of the counter.

```java
private static int usedConfigurationTickets = 0;
```

// create the components of the reconfiguration process
```java
FocusDeterminationInterface focusDetermination = new FocusDetermination();
RequirementDeterminationInterface reqDetermination = new RequirementDetermination();
TemplateBasedConfigurationInterface templateConfiguration = new TemplateBasedConfiguration();
IntegrationInterface integration = new Integration();
```

// end of component creation

/**
 * private storage of information on the requested reconfiguration jobs
 */
private Collection<StartingInfo> receivedStartingInformation;

public ReconfigurationProcess() {
    receivedStartingInformation = new LinkedList<StartingInfo>();

    // create test situation
    int reconfigurationTicket = setInitialData();
    requestStartReconfiguration(reconfigurationTicket);
}

/**
 * Assumed that this method will be called in sequence, multithreading is not supported
 */
public int generateReconfigurationTicket() {
    usedConfigurationTickets += 1;
    int newTicket = usedConfigurationTickets;
    log.info("Reconfiguration ticket has been generated. Nr: "+
             newTicket+". ");
    return newTicket;
}
public boolean isFinished(int reconfigurationTicket) {
    StartingInfo info = retrieveStartingInformation(reconfigurationTicket);
    if (info == null) {
        return false;
    } else {
        return info.reconfigurationCompleted;
    }
}

public boolean requestStartReconfiguration(int reconfigurationTicket) {
    log.info("Following reconfiguration process is started:" +
             reconfigurationTicket);
    return startReconfiguration(reconfigurationTicket);
}

public void setReconfigurationInformation(int reconfigurationTicket,
                                           FailingWebServiceInterface webService,
                                           InitialConfigurationInterface initialConfig,
                                           RepositoriesInterface repositories) {
    // create a new object holding all this information;
    StartingInfo info = new StartingInfo(reconfigurationTicket,
                                          webService,
                                          initialConfig,
                                          repositories);

    // if information already existed with this starting ticket,
    // must be removed
    // check whether this reconfiguration ticket is already used
    StartingInfo existingInformation =
        retrieveStartingInformation(reconfigurationTicket);
    if (existingInformation != null) {
        receivedStartingInformation.remove(existingInformation);
    }

    // add the new information
    receivedStartingInformation.add(info);
    log.info("Information has been added");
}

/**
 * Method returns the starting information of the given ticket
 * if it is
 * available if not, it returns null;
private StartingInfo retrieveStartingInformation(int reconfigurationTicket) {
    boolean extendedLog = true;

    if (extendedLog) {
        log.info("retrieving Starting information, starting collection size:" + receivedStartingInformation.size());
    }

    for (Iterator i = receivedStartingInformation.iterator(); i.hasNext();) {
        StartingInfo informationObject = (StartingInfo) i.next();
        if (extendedLog) {
            log.info("Ticket in history:" + informationObject.reconfigurationTicket);
        }
        // check whether the ticket equals the given ticket
        if (informationObject.reconfigurationTicket == reconfigurationTicket) {
            // equal reconfiguration ticket found
            // return this starting information
            return informationObject;
        }
    }

    // no matching starting information found
    // return null
    return null;
}

/**
   * Small class to store the starting information of a single reconfiguration process
   * @author sandervansplunter
   */
private class StartingInfo {
    public int reconfigurationTicket;

    public FailingWebServiceInterface webService;
public InitialConfigurationInterface initialConfig;

public RepositoriesInterface repositories;

public FocusInterface currentFocus = null;

/**
 * Storing the result after reconfiguration;
 */
public NewConfigurationInterface resultingConfiguration = null;

/**
 * Boolean to track whether this problem has been solved
 */
boolean reconfigurationCompleted = false;

/**
 * Requires all the information needed at the start of a
 * re-configuration process.
 * @param reconfigurationTicket
 * the identifier for the reconfiguration process
 * @param webService
 * the pointer to the failing web service
 * @param initialConfig
 * the initial web service configuration that is
 * to be
 * reconfigured
 * @param repositories
 * the available web service and template
 * repositories
 */
public StartingInfo(int reconfigurationTicket, 
    FailingWebServiceInterface webService, 
    InitialConfigurationInterface initialConfig, 
    RepositoriesInterface repositories) {
    this.reconfigurationTicket = reconfigurationTicket;
    this.webService = webService;
    this.initialConfig = initialConfig;
    this.repos = repositories;
}

/**
 * Compares based on the reconfigurationTicket
 * @Override
 */
public boolean equals(Object obj) {
    if (obj instanceof StartingInfo) {
        StartingInfo comparedInfo = (StartingInfo) obj;
        if (comparedInfo.reconfigurationTicket ==
            reconfigurationTicket) {
            return true;
        }
    }
    return false;
}

public void setReconfigurationResult(
    NewConfigurationInterface newConfiguration) {
    reconfigurationCompleted = true;
    resultingConfiguration = newConfiguration;
}

/**
 * Start a reconfiguration process for a specific reconfiguration problem
 * @param reconfigurationTicket specifying the reconfiguration problem that is to be reconfigured
 * @return true if the process is completed, false if the information for the reconfiguration ticket is not present
 */
public boolean reconfigure(int reconfigurationTicket) {
    return startReconfiguration(reconfigurationTicket);
}

/**
 * Starts the first loop of reconfiguration. If successful, only a single focus is created
 * @param reconfigurationTicket the re-configuration process involved
 * @return true if successful
 */
private boolean startReconfiguration(int reconfigurationTicket) {
    // retrieve the starting information
log.info("starting reconfiguration of "+ reconfigurationTicket);
StartingInfo startInfo = retrieveStartingInformation(
    reconfigurationTicket);
if (startInfo == null) {
    log.info("Reconfiguration starting information was not set ");
    return false;
}

// retrieve the initial information
FailingWebServiceInterface failingService = startInfo.
    webService;
InitialConfigurationInterface initialConfiguration =
    startInfo.initialConfig;

// start the control loop.

// retrieve the focus
FocusInterface currentFocus = startInfo.currentFocus;
log.info("Determining the focus");
// if process has not been started then the focus still
// needs to be
// initialised
if (currentFocus == null) {
    log.info("Focus has not been determined earlier");
    currentFocus = focusDetermination
        .createInitialFocus(reconfigurationTicket, failingService,
            initialConfiguration);
    // store the focus as reconfiguration information (needed for
    // recursive reconfiguration);
    startInfo.currentFocus = currentFocus;
}
else {
    log.info("Focus has been determined before");
    currentFocus = focusDetermination.adaptFocus(
        reconfigurationTicket);
    // store the focus as reconfiguration information (needed for
    // recursive reconfiguration);
    startInfo.currentFocus = currentFocus;
}

// check whether the new focus is null
if (currentFocus.getFocus() == null) {
    log.info("Unable to determine a focus, reconfiguration will fail");
}
// a new focus could not be found. the reconfiguration process fails
return false;
}
log.info("new\_focus\=" +currentFocus.getFocus().
  getConfigurationID());

// the set of requirements for the configuration process
RequirementSetInterface currentRequirements;
// get the requirements
currentRequirements = reqDetermination.determineRequirements
  (reconfigurationTicket, initialConfiguration,
   currentFocus,
   failingService);
log.info("The following requirements are relevant" +
currentRequirements);
// the configuration resulting from the configuration process
AdditionalConfigurationInterface additionalConfiguration;
// template configuration
additionalConfiguration = templateConfiguration.
createConfiguration(
  reconfigurationTicket, currentRequirements);

// check if the resulting configuration is null
if (((DODInstance)additionalConfiguration.
  getAdditionalConfiguration()).getMainBuildingBlock() ==
  null) {
  // the template-based configuration could not find an configuration
  // for this focus.
  // refocus and do loop thing?
  log.info("Reconfiguration\_process: the configuration\_ process returned\_ an\_ empty\_ result ,\_ configuration\_ has\_ failed");
  return startReconfiguration(reconfigurationTicket);
} else {
  // successfully found a configuration.
  log.info("Successfully\_ found\_ a\_ configuration");
  log.info("ResultingConfiguration\=" + ((DODInstance)
    additionalConfiguration.getAdditionalConfiguration()).
    toStringExtended());
  NewConfigurationInterface newConfiguration;
  // integrate the configurations
  newConfiguration = integration
.integrate(reconfigurationTicket, additionalConfiguration, initialConfiguration, currentFocus);

// store the result
startInfo.setReconfigurationResult(newConfiguration);

// return true, reconfiguration has succeeded
return true;

}

public static void main(String[] args) {
    // starting a configuration problem
    // set initial data
    new ReconfigurationProcess();

}

private int setInitialData() {
    ExperimentHandler test = new ExperimentHandler(false);
    DODInstance dod = (DODInstance) test.getReconstructedDod();
    System.out.println("The reconfiguration starts with: ");
    System.out.println(dod.toStringExtended());
    InitializationInterface initialConf = new InitialConfiguration();
    initialConf.setDod(dod);

    // initialise the failing web service
    FailingWebServiceInterface failingService = new FailingWebService(dod.getFailedComponentName());
    int ticket = generateReconfigurationTicket();
    setReconfigurationInformation(ticket, failingService, initialConf, (new Repositories()));

    return ticket;
}
}
Overall Reconfiguration Process
Appendix E

Focus Determination

package versie_2.component;

import inProgress.DODInstance;

import java.util.Collection;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Vector;

import versie_2.informationTypes.FailingWebServiceInterface;
import versie_2.informationTypes.Focus;
import versie_2.informationTypes.FocusInterface;
import versie_2.informationTypes.InitialConfigurationInterface;
import webService.informationTypes.BuildingBlock;
import webService.informationTypes.Slot;

/**
 * @author sandervansplunter
 *
 */
public class FocusDetermination implements FocusDeterminationInterface {

    Collection<ReconfigurationFocusHistory> focusItem =
            new LinkedList<ReconfigurationFocusHistory>();

    // retrieve the information form the focus History;
    public FocusInterface adaptFocus(int reconfigurationTicket) {
        ReconfigurationFocusHistory historyItem = null;
        for (Iterator i = focusItem.iterator(); i.hasNext();) {
            ReconfigurationFocusHistory element =
                    (ReconfigurationFocusHistory) i.next();
if (element.reconfigurationTicket == reconfigurationTicket) {
    // this is the element that we need.
    historyItem = element;
}

// adapt the focus to the slot above the previous focus
FocusInterface previousFocus = historyItem.foci.getLast();

// retrieve the slot in which the parent template is inserted;
Slot newSlot = previousFocus.getFocus().getParentTemplate().getParentSlot();
FocusInterface newFocus = new Focus(newSlot);
historyItem.foci.add(newFocus);
return newFocus;

public FocusInterface createInitialFocus(int reconfigurationTicket,
                                          FailingWebServiceInterface webService,
                                          InitialConfigurationInterface initConf) {

    // create a reference, specific for this reconfiguration ticket.
    ReconfigurationFocusHistory focusHistory =
        new ReconfigurationFocusHistory(
            reconfigurationTicket, webService, initConf);
    historyItem.add(focusHistory);
    // simple policy: initial focus is the slot in which the failing Web
    // service has been inserted.
    DODInstance initialDOD = initConf.getDOD();
    BuildingBlock main = initialDOD.getMainBuildingBlock();

    // get all building blocks excluding the main bb
    Vector<BuildingBlock> theBlocks = main
        .getInsertedBuildingBlocksRecusively();
    // add the main bb
    theBlocks.add(main);

    // now find the correct building block
    String nameFailingService = webService.getName();

    for (Iterator i = theBlocks.iterator(); i.hasNext();)
    {
        BuildingBlock element = (BuildingBlock) i.next();
        //...
if (element.getName().equalsIgnoreCase(nameFailingService)) {
    // failing service found (or at least one instance found)
    // get the slot in which the failing service is inserted
    Slot parentSlot = element.getParentSlot();
    FocusInterface focus = new Focus(parentSlot);
    focusHistory.foci.add(focus);
    return focus;
}

// the failing service is not part of the given initial configuration,
// a focus can not be determined
FocusInterface focus = new Focus(null);
focusHistory.foci.add(focus);
return new Focus(null);

/**
 * @author sandervansplunter This class is to maintain the information about
 * a single re-configuration process. \ It is supposed to maintain
 * all starting information and all the foci that have been selected
 * within this process.
 */
private class ReconfigurationFocusHistory {

    // all the starting information
    public int reconfigurationTicket;
    public FailingWebServiceInterface webService;
    public InitialConfigurationInterface initConf;

    /**
     * This contains all the foci that have been selected
     */
    public LinkedList<FocusInterface> foci = new LinkedList<
            FocusInterface>();

    public ReconfigurationFocusHistory(int reconfigurationTicket
            , FailingWebServiceInterface webService,
InitialConfigurationInterface initConf) {
    this.reconfigurationTicket = reconfigurationTicket;
    this.webService = webService;
    this.initConf = initConf;
}
}
Appendix F

Requirement Determination

```java
package versie_2.component;

import java.util.Collection;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Vector;
import designProcess.informationTypes.firstMetaLevel.RequirementInstance;
import versie_2.informationTypes.FailingWebServiceInterface;
import versie_2.informationTypes.FocusInterface;
import versie_2.informationTypes.InitialConfigurationInterface;
import versie_2.informationTypes.RequirementSet;
import versie_2.informationTypes.RequirementSetInterface;
import webService.informationTypes.Slot;
import webService.informationTypes.Specification;
import webService.informationTypes.structures.MyProperty;

public class RequirementDetermination implements RequirementDeterminationInterface {

    public RequirementSetInterface determineRequirements(
        int reconfigurationTicket,
        InitialConfigurationInterface configuration,
        FocusInterface focus,
        FailingWebServiceInterface failingService) {

        Collection<RequirementInstance> requirements =
            new Vector<RequirementInstance>();

        Slot focusedSlot = focus.getFocus();
        Specification s = focusedSlot.getSemanticDescription();
        RequirementInstance requirement = new RequirementInstance("func:"
+ s.getFunctionalSpec().FunctionalityToString() );
requirements.add(requirement);

// add other requirements if they are available
LinkedList<MyProperty> additionalProperties = s
    .getAdditionalProperties();
if (!additionalProperties.isEmpty()) {
    for (Iterator i = additionalProperties.iterator(); i.
        hasNext(); ) { 
        MyProperty property = (MyProperty) i.next();
        RequirementInstance rq2 = new RequirementInstance("pos:
        + property.toString() );
        requirements.add(rq2); 
    }
}

RequirementSetInterface requirementSet = new RequirementSet(
    failingService, requirements); 
return requirementSet;
}
Appendix G

Template-based Configuration

```java
package versie_2.component;
import java.util.Collection;
import java.util.Iterator;
import java.util.Vector;
import buildingBlockRetrieval.ComponentLibrary;
import buildingBlockRetrieval.ComponentQueryHandler;
import gdmRebuild.informationTypes.firstMetaLevel.dod.DOD;
import gdmRebuild.informationTypes.secondMetaLevel.unintegrated.RQS;
import inProgress.DODInstance;
import inProgress.RQSInstance;
import designProcess.DesignProcess;
import designProcess.WASABE.RQSM.WASABERequirementKnowledge;
import designProcess.informationTypes.firstMetaLevel.RequirementInstance;
import util.Settings;
import versie_2.informationTypes.AdditionalConfiguration;
import versie_2.informationTypes.AdditionalConfigurationInterface;
import versie_2.informationTypes.RequirementSetInterface;
import webService.informationTypes.BuildingBlock;

public class TemplateBasedConfiguration implements
        TemplateBasedConfigurationInterface {

    public AdditionalConfigurationInterface createConfiguration(
            int reconfigurationTicket,
            RequirementSetInterface currentRequirements) {

        // make sure that the exclusion requirement is performed
        handleExclusionRequirement(currentRequirements);
    }
```
// add the requirements
RQSInstance rqs = new RQSInstance();
Collection<RequirementInstance> requirements =
currentRequirements
  .getRequirements();

for (Iterator iter = requirements.iterator(); iter.hasNext();)
{
  RequirementInstance singleRequirement = (
    RequirementInstance) iter
  .next();
rqs.addRequirement(singleRequirement);

  // evaluate other requirements
  if (singleRequirement.getDefinition().toString().contains(
      "func")) {
    WASABERequirementKnowledge
      .addRequirementAsFunctional(singleRequirement);
  } else {
    WASABERequirementKnowledge
      .addRequirementAsOther(singleRequirement);
  }
}

DesignProcess designProcess = new DesignProcess(new
  DODInstance(),
  (RQS) rqs), (new Settings()));

// configuration is finished
DOD resultingDOD = designProcess.getCurrentDOD();

// reset excluded components
resetExcludedComponents();

  return (new AdditionalConfiguration(resultingDOD));
}

/**
* This method reset the excluded components within the
* configuration process. This is required to be used between two different
* reconfiguration processes, otherwise excluded components will also not be
* available for the following reconfiguration process
* */
```java
private void resetExcludedComponents() {
    ComponentLibrary.excludedBuildingBlocks = new Vector<
        BuildingBlock>();
}

/**
 * This method handles the requirement that some structural components are
 * not allowed to be used within reconfiguration.
 * @param currentRequirements the set of requirements containing the
 * structural requirement
 */
private void handleExclusionRequirement(
    RequirementSetInterface currentRequirements) {
    // modify the static collection of excluded components contained within
    // the library
    ComponentQueryHandler queryHandler = new
        ComponentQueryHandler();

    String failedWsName = currentRequirements.
        getFailingWebService().
            getName();
    BuildingBlock failedBB = queryHandler
        .getBuildingBlockByName(failedWsName);
    if (failedBB != null) {
        //
        ComponentLibrary.excludedBuildingBlocks.add(failedBB);
        System.out.println("Excluding BuildingBlock: " +
            failedBB.toString());
    } else {
        System.out
            .println("Failed web-service could not be found when handling exclusion requirements");
        System.out.print("Failed BB name: " + failedWsName);
        System.exit(1);
    }
}
```
package versie_2.component;

import inProgress.DODInstance;
import versie_2.informationTypes.
    AdditionalConfigurationInterface;
import versie_2.informationTypes.FocusInterface;
import versie_2.informationTypes.InitialConfigurationInterface;
import versie_2.informationTypes.NewConfiguration;
import versie_2.informationTypes.NewConfigurationInterface;
import webService.informationTypes.BuildingBlock;

public class Integration implements IntegrationInterface {

    public NewConfigurationInterface integrate(int reconfigurationTicket,
                                                AdditionalConfigurationInterface additionalConfiguration,
                                                InitialConfigurationInterface initialConfiguration,
                                                FocusInterface currentFocus) {

        BuildingBlock constructedAlternative =
            ((DODInstance) additionalConfiguration
                .getAdditionalConfiguration()).getMainBuildingBlock();
        System.out.println("InitialConfiguration:"
            + initialConfiguration.toString());

        BuildingBlock previouslyInsertedBuildingBlock =
            currentFocus.getFocus().getFiller();

        currentFocus.getFocus().fillWith(constructedAlternative);
// Copy the adapted initial configuration
DODInstance newDOD = new DODInstance(initialConfiguration.
getDOD());
// restore the initial configuration for future reference
currentFocus.getFocus().fillWith(
previouslyInsertedBuildingBlock);

NewConfigurationInterface newConfiguration = new
NewConfiguration(
newDOD);

// perform the additional configuration on the newly
created
// configuration
doAnnotation(newConfiguration);

return newConfiguration;
}

/**
 * This method performs the additional annotations on the
 * configuration to
 * included all propagated properties
 * @param newConfiguration
 * the configuration that is to be annotated
 */
private void doAnnotation(NewConfigurationInterface
newConfiguration) {
BuildingBlock mainComponent;
// propagate all properties
mainComponent = newConfiguration.getMainBuildingBlock();
// check whether configuration is not an empty
configuration;
if (mainComponent != null) {
// this method propagates and annotates the resulting
component
mainComponent.propagateProperties();
}
}
Part IV

Traces
## Appendix I

### Trace of Single-Level Reconfiguration Example

#### Initialisation

| Settings: loading conf/ibrow/designFiles.conf |
| Settings: loading conf/ibrow/scenario/qsInit_extended.conf |
| Settings: loading conf/ibrow/scenario/dodInit.conf |

Let's set the initialisation process:

- **t1_slot5, t1_slot4, t1_slot3, t1_slot2, t1_slot1, failedComponent, w1_name, t1_name, empty, t2_name, mainBuildingBlock, w2_name, t2_slot4, t2_slot3, t2_slot2, w3_name, t2_slot1, w4_name, w5_name**

**searching for t1_name**

**CurrentBBname: classification**

- retrieve slot1 of template t1
- searching for w1_name

**CurrentBBname: OC1−single−value−constraint**

- retrieve slot2 of template t1
- searching for w2_name

**CurrentBBname: FS1−IEUM**

- retrieve slot3 of template t1
- searching for w3_name

**CurrentBBname: MS1a−IEUM**

- retrieve slot4 of template t1
- searching for w4_name

**CurrentBBname: AC( Sconsistent−Sknown−Spresent ) variants**

- retrieve slot5 of template t1
- searching for w5_name

**CurrentBBname: BM4−highest**

- retrieve slot6 of template t1

**classification observationConstraints(configurationID_34) filled with OC1−single−value−constraint**

**featureScoringMechanism(configurationID_35) filled with FS1−IEUM**

**macroScoringMechanism(configurationID_36) filled with MS1a−IEUM**

**solutionAdmissibilityCriterion(configurationID_37) filled with AC( Sconsistent−Sknown−Spresent ) variants**

**betterMatchScores(configurationID_38) filled with BM4−highest**

---

The reconfiguration starts with:

**DOO2**

**classification observationConstraints(configurationID_34) filled with OC1−single−value−constraint**

**featureScoringMechanism(configurationID_35) filled with FS1−IEUM**

**macroScoringMechanism(configurationID_36) filled with MS1a−IEUM**

**solutionAdmissibilityCriterion(configurationID_37) filled with AC( Sconsistent−Sknown−Spresent ) variants**

---

The reconfiguration starts with:

**DOO2**

**classification observationConstraints(configurationID_34) filled with OC1−single−value−constraint**

**featureScoringMechanism(configurationID_35) filled with FS1−IEUM**

**macroScoringMechanism(configurationID_36) filled with MS1a−IEUM**

**solutionAdmissibilityCriterion(configurationID_37) filled with AC( Sconsistent−Sknown−Spresent ) variants**
I.1 Focus Determination

INFO: Determining the focus
INFO: Focus has not been determined earlier
INFO: new focus = configurationID_37

I.2 Requirement Determination

INFO: The following requirements are relevant
requirements = SetFailing_service:version_2.informationTypes.FailingWebService@e6ff0d
r1 func:AC(Sconsistent–Sknown–Spresent) variants

I.3 Template-based Configuration

Initiate a new configuration and requirement set

Excluding BuildingBlock: AC(Sconsistent–Sknown–Spresent) variants
Settings: loading conf\ibrow\designFiles.conf
Settings: loading conf\ibrow\scenario/rq_all\possible, rqs.conf
rqDefinitions have been set
WARNING: Status=report of DPC

No. of times activated: 1
---INPUT:
Received DDM Process Evaluation:
ProcessEvaluationCollection POC_3
Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_2
Find a component satisfying the initial requirement set
Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_9
processEvaluation(Strat_1, GenerateMainTemplateOnFunctionality, Succeeded, RQSM)
Succeeded

---OUTPUT---
Strategy for component to be activated: Strat_1, GenerateMainTemplateOnFunctionality

05−Oct−2009 23:48:12 designProcess.components.DOCM logStatus
WARNING: Status−report of DOCM

No. of times activated: 2

---INPUT---
Received overall Strategy:
Strat_1, GenerateMainTemplateOnFunctionality
Received RQSinstance:
rqs_5

---OUTPUT---
Resulting Overall Dod Assessments:

Resulting RQSinstance Assessments:

Resulting DOD:
DOD_8
composedACvariants
ACvariant1(configurationID_602) not filled
ACvariant2(configurationID_603) not filled
ACvariant3(configurationID_604) not filled
CombineResults(configurationID_605) not filled

Resulting Process Evaluation:
ProcessEvaluationCollection POC_14
processEvaluation(Strat_1, GenerateMainTemplateOnFunctionality, Succeeded, DOCM)
Succeeded

---INPUT---
Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_9
processEvaluation(Strat_1, GenerateMainTemplateOnFunctionality, Succeeded, RQSM)
Succeeded

---OUTPUT---
Strategy for component to be activated: Strat_18, RefineSelectedTemplate

WARNING: Status−report of RQSM

No. of times activated: 2

---INPUT---
Received overall Strategy:

Incrementally fill open slots
I.3 Template-based Configuration

RefineSelectedTemplate

Received DODAssessment:

\[
\begin{align*}
&\text{includesBasicEvaluationInformation}\left(\text{satisfies}\left(DOD_A,\{qr_6,\text{hard},\{r_1 \text{func:AC=Sconsistentskown-Spresent}\}\},\text{true}\right), \ldots\text{true}\right), \\
&\text{includesBasicEvaluationInformation}\left(\text{satisfies}\left(DOD_A,\{qr_2,\text{hard},\{r_5 \text{configurationID=602,func:AC=Sconsistent}\}\},\text{false}\right), \ldots\text{true}\right), \\
&\text{includesBasicEvaluationInformation}\left(\text{satisfies}\left(DOD_A,\{qr_3,\text{hard},\{r_6 \text{configurationID=602,func:AC=Sconsistent}\}\},\text{false}\right), \ldots\text{true}\right), \\
&\text{includesBasicEvaluationInformation}\left(\text{satisfies}\left(DOD_A,\{qr_4,\text{hard},\{r_7 \text{configurationID=602,func:AC=Sconsistent}\}\},\text{false}\right), \ldots\text{true}\right), \\
&\text{includesBasicEvaluationInformation}\left(\text{satisfies}\left(DOD_A,\{qr_5,\text{hard},\{r_8 \text{configurationID=605,func:CombineACs}\}\},\text{false}\right), \ldots\text{true}\right), \\
\end{align*}
\]

Enacted Requirements:

\[
\begin{align*}
&\text{IsToBeSatisfied}\left(r_1 \text{func:AC=Sconsistent-Sknown-Spresent}\right), \\
&\text{IsToBeSatisfied}\left(r_3 \text{configurationID=602,func:AC=Sconsistent}\right), \\
&\text{IsToBeSatisfied}\left(r_5 \text{configurationID=602,func:AC=Sknown}\right), \\
&\text{IsToBeSatisfied}\left(r_7 \text{configurationID=604,func:AC=Sknown}\right), \\
&\text{IsToBeSatisfied}\left(r_9 \text{configurationID=604,func:AC=S inconsistent}\right), \\
&\text{IsToBeSatisfied}\left(r_11 \text{configurationID=604,func:AC=S consistent}\right), \\
&\text{IsToBeSatisfied}\left(r_13 \text{configurationID=605,func:CombineACs}\right), \\
&\text{IsToBeSatisfied}\left(r_15 \text{configurationID=605,func:CombineACs}\right), \\
&\text{IsToBeSatisfied}\left(r_17 \text{configurationID=605,func:CombineACs}\right), \\
&\text{IsToBeSatisfied}\left(r_19 \text{configurationID=605,func:CombineACs}\right), \\
\end{align*}
\]

Resulting RQInstance

processEvaluationCollection POC_10

processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, RQSM)

Resulting Process Evaluation

Strat_18, RefineSelectedTemplate

Output—

No. of times activated: 5

---INPUT---

Received DODM Process Evaluation:

processEvaluation(Strat_18, GenerateMainTemplateOnFunctionality, Succeeded, DODM)

Succeeded

Resulting RQSM Process Evaluation:

processEvaluationCollection POC_20

processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, RQSM)

---OUTPUT---

Strategy for component to be activated: Strat_18


No. of times activated: 3

---INPUT---

Received overall Strategy:

Strat_18, RefineSelectedTemplate

Received RQInstance:

resulting RQInstance:

[]

---OUTPUT---

Resulting Overall DOD Assessments:

[]

Resulting RQInstance Assessments:

\[
\begin{align*}
&\text{includesBasicEvaluationInformation}\left(\text{satisfies}\left(DOD_A,\{qr_6,\text{hard},\{r_1 \text{func:AC=Sconsistent-Sknown-Spresent}\}\},\text{true}\right), \ldots\text{true}\right), \\
&\text{includesBasicEvaluationInformation}\left(\text{satisfies}\left(DOD_A,\{qr_2,\text{hard},\{r_5 \text{configurationID=602,func:AC=Sconsistent}\}\},\text{false}\right), \ldots\text{true}\right), \\
&\text{includesBasicEvaluationInformation}\left(\text{satisfies}\left(DOD_A,\{qr_3,\text{hard},\{r_6 \text{configurationID=602,func:AC=Sconsistent}\}\},\text{false}\right), \ldots\text{true}\right), \\
&\text{includesBasicEvaluationInformation}\left(\text{satisfies}\left(DOD_A,\{qr_4,\text{hard},\{r_7 \text{configurationID=602,func:AC=Sconsistent}\}\},\text{false}\right), \ldots\text{true}\right), \\
&\text{includesBasicEvaluationInformation}\left(\text{satisfies}\left(DOD_A,\{qr_5,\text{hard},\{r_8 \text{configurationID=605,func:AC=Sknown}\}\},\text{false}\right), \ldots\text{true}\right),
\end{align*}
\]
Trace of Single-Level Reconfiguration Example

IncludesBasicEvaluationInformation([rqs_7, ...] satisfies (DOD_0, (qr14, hard, [r15 configurationID_600, func:AC(Spresent)variants(refines r11)])}, true),
...true), true).
IncludesBasicEvaluationInformation([rqs_7, ...] satisfies (DOD_0, (qr15, hard, [r20 configurationID_600, func:AC(Sknown)variants()]), false),
...true), true).
IncludesBasicEvaluationInformation([rqs_7, ...] satisfies (DOD_0, (qr14, hard, [r19 configurationID_600, func:AC(Spresent)variants()]), false),
...true), true).
IncludesBasicEvaluationInformation([rqs_7, ...] satisfies (DOD_0, (qr13, hard, [r18 configurationID_600, func:AC(Sknown)variants()]), false),
...true), true).

Resulting DOD: DOD_0

composedACvariants ACvariant1(configurationID_600) not filled with AC(Sconsistent)variants
ACvariant2(configurationID_600) not filled
ACvariant3(configurationID_604) not filled
CombineResults(configurationID_605) not filled

Resulting Process Evaluation:
ProcessEvaluationCollection POC_24
processEvaluation(Strat_18, RefineSelectedTemplate, Incomplete, DODM) Incomplete, more activations are needed, or input from RQSM

05-Oct-2009 23:48:12 design components DPCold logStatus
WARNING: Status-report of DPC

No. of times activated: 6

INPUT:
Received DODM Process Evaluation:
ProcessEvaluationCollection POC_24
processEvaluation(Strat_18, RefineSelectedTemplate, Incomplete, DODM) Incomplete, more activations are needed, or input from RQSM

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_24
processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

OUTPUT:
Strategy for component to be activated: Strat_18, RefineSelectedTemplate

05-Oct-2009 23:48:12 design components RQSM logStatus
WARNING: Status-report of RQSM

No. of times activated: 3

INPUT:
Received overall Strategy:
Strat_18, RefineSelectedTemplate
Received DODAssessment:
[...]
IncludesBasicEvaluationInformation([rqs_7, ...] satisfies (DOD_0, (qr16, hard, [r17 configurationID_600, func:AC(Sconsistent-Sknown-Spresent)variants()]), true),
...true), true).
IncludesBasicEvaluationInformation([rqs_7, ...] satisfies (DOD_0, (qr17, hard, [r13 configurationID_600, func:AC(Sconsistent)variants(refines r9)])}, true),
...true), true).
IncludesBasicEvaluationInformation([rqs_7, ...] satisfies (DOD_0, (qr18, hard, [r14 configurationID_600, func:AC(Sknown)variants(refines r10)]), false),
...true), true).
IncludesBasicEvaluationInformation([rqs_7, ...] satisfies (DOD_0, (qr19, hard, [r15 configurationID_600, func:AC(Spresent)variants(refines r11)])}, false),
...true), true).
IncludesBasicEvaluationInformation([rqs_7, ...] satisfies (DOD_0, (qr20, hard, [r16 configurationID_600, func:CombineACs(refines r12)]), false),
...true), true).
IncludesBasicEvaluationInformation([rqs_7, ...] satisfies (DOD_0, (qr12, hard, [r17 configurationID_600, func:AC(Sconsistent)variants()]), false),
...true), true).
IncludesBasicEvaluationInformation([rqs_7, ...] satisfies (DOD_0, (qr13, hard, [r18 configurationID_600, func:AC(Sknown)variants()]), false),
...true), true).
IncludesBasicEvaluationInformation([rqs_7, ...] satisfies (DOD_0, (qr14, hard, [r19 configurationID_600, func:AC(Spresent)variants()]), false),
...true), true).
I.3 Template-based Configuration

```plaintext
IncludesBasicEvaluationInformation((rq9,...,satisfies((DOD_20,qr71,hard,[r71 configurationID_605,func:CombineACs()]),false),...,[true],true)),

---INPUT---
Resulting RQInstance
rq9
internal present requirements:
(r1 func:AC(Sconsistent-Sknown-Spresent)variants,true)
(r9 configurationID_602,func:AC(Sconsistent)variants,true)
(r13 configurationID_602,func:AC(Sconsistent)variants(refines r9),true)
(r10 configurationID_603,func:AC(Sknown)variants,true)
(r14 configurationID_603,func:AC(Sknown)variants(refines r10),true)
(r11 configurationID_604,func:AC(Spresent)variants(true)
(r12 configurationID_605,func:CombineACs(true)

Enacted Requirements:
IsToBeSatisfied(r1 func:AC(Sconsistent-Sknown-Spresent)variants)
IsToBeSatisfied(r13 configurationID_602,func:AC(Sconsistent)variants(refines r9))
IsToBeSatisfied(r14 configurationID_603,func:AC(Sknown)variants(refines r10))
IsToBeSatisfied(r15 configurationID_604,func:AC(Spresent)variants(refines r11))
IsToBeSatisfied(r16 configurationID_605,func:CombineACs(refines r12))

Resulting Process Evaluation:
ProcessEvaluationCollection POC_19
processEvaluation(Strat_5,RefineSelectedTemplate,Succeeded,RQSM,Succeeded

WARNING Status-report of POC

No. of times activated: 7

---INPUT---
Received DODM Process Evaluation:
ProcessEvaluationCollection POC_24
processEvaluation(Strat_18,RefineSelectedTemplate,Incomplete,DXM) Incomplete, more activations are needed, or input from RQSM

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_20
processEvaluation(Strat_18,RefineSelectedTemplate,Succeeded,RQSM,Succeeded

---OUTPUT---
Strategy for component to be activated: Strat_18, RefineSelectedTemplate

WARNING Status-report of DXM

No. of times activated: 4

---INPUT---
Received overall Strategy:
Strat_18, RefineSelectedTemplate

Received RQ Instance:
rq9

---OUTPUT---
Resulting Overall Dod Assessments:

Resulting RQ Instance Assessments:

[IncludesBasicEvaluationInformation((rq9,...,satisfies((DOD_12,(qr30,hard,[r30 configurationID_605,func:CombineACs()]),false),...,[true],true)),

IncludesBasicEvaluationInformation((rq9,...,satisfies((DOD_12,(qr31,hard,[r31 configurationID_602,func:AC(Sconsistent)variants(refines r9)]),false),...,[true],true),

IncludesBasicEvaluationInformation((rq9,...,satisfies((DOD_12,(qr32,hard,[r32 configurationID_603,func:AC(Sknown)variants(refines r10)]),false),...,[true],true),

IncludesBasicEvaluationInformation((rq9,...,satisfies((DOD_12,(qr33,hard,[r33 configurationID_604,func:AC(Spresent)variants(refines r11)]),false),...,[true],true),

IncludesBasicEvaluationInformation((rq9,...,satisfies((DOD_12,(qr34,hard,[r34 configurationID_605,func:CombineACs(refines r12)]),false),...,[true],true),

IncludesBasicEvaluationInformation((rq9,...,satisfies((DOD_12,(qr35,hard,[r35 configurationID_606,func:AC(Sconsistent)variants]),false),...,[true],true),

IncludesBasicEvaluationInformation((rq9,...,satisfies((DOD_12,(qr36,hard,[r36 configurationID_607,func:AC(Sknown)variants]),false),...,[true],true),

IncludesBasicEvaluationInformation((rq9,...,satisfies((DOD_12,(qr37,hard,[r37 configurationID_608,func:AC(Spresent)variants]),false),...,[true],true),

IncludesBasicEvaluationInformation((rq9,...,satisfies((DOD_12,(qr38,hard,[r38 configurationID_609,func:AC(Sconsistent)variants]),false),...,[true],true),

IncludesBasicEvaluationInformation((rq9,...,satisfies((DOD_12,(qr39,hard,[r39 configurationID_610,func:AC(Sknown)variants]),false),...,[true],true),

IncludesBasicEvaluationInformation((rq9,...,satisfies((DOD_12,(qr40,hard,[r40 configurationID_611,func:AC(Spresent)variants]),false),...,[true],true)
```

192 Trace of Single-Level Reconfiguration Example

```plaintext
| ... , true , true ) ,
| includesBasicEvaluationInformation ( [ rqs , ... ] satisfies ( ( DOD , qr29 , hard , [ r24 configurationID_605 , func:CombineACs ] ) , false ) ,
| ... , true , true ) ,
| Resulting DOD:
| DOD_12
| composedACvariants ACVariant1 ( configurationID_602 ) filled with AC( Sconsistent ) variants
| ACVariant2 ( configurationID_603 ) filled with AC( Sknown ) variants
| ACVariant3 ( configurationID_604 ) not filled
| CombineResults ( configurationID_605 ) not filled
| Resulting Process Evaluation:
| ProcessEvaluationCollection POC_34
| processEvaluation ( Strat_18 , RefineSelectedTemplate , Incomplete , DOD ) Incomplete , more activations are needed , or input from RQSM
| WARNING: Status - report of DPC
| No. of times activated: 8
| ---INPUT---
| Received DOD Process Evaluation:
| ProcessEvaluationCollection POC_34
| processEvaluation ( Strat_18 , RefineSelectedTemplate , Incomplete , DOD ) Incomplete , more activations are needed , or input from RQSM
| Received RQSM Process Evaluation:
| ProcessEvaluationCollection POC_34
| processEvaluation ( Strat_18 , RefineSelectedTemplate , Succeeded , RQSM ) Succeeded
| ---OUTPUT---
| Strategy for component to be activated: Strat_18 , RefineSelectedTemplate
| WARNING: Status - report of RQSM
| No. of times activated: 4
| ---INPUT---
| Received overall Strategy:
| Strat_18 , RefineSelectedTemplate
| Received DOD Assessment:
| [ ... ] IncludesBasicEvaluationInformation ( [ rqs , ... ] satisfies ( ( DOD , qr30 , hard , [ r1 configurationID_605 , func:CombineACs ] ) , true ) ,
| ... , true , true ) ,
| IncludesBasicEvaluationInformation ( [ rqs , ... ] satisfies ( ( DOD , qr31 , hard , [ r13 configurationID_602 , func:AC( Sconsistent ) variants ( refines r9 ) ] ) , true ) ,
| ... , true , true ) ,
| IncludesBasicEvaluationInformation ( [ rqs , ... ] satisfies ( ( DOD , qr32 , hard , [ r14 configurationID_603 , func:AC( Sknown ) variants ( refines r10 ) ] ) , true ) ,
| ... , true , true ) ,
| IncludesBasicEvaluationInformation ( [ rqs , ... ] satisfies ( ( DOD , qr33 , hard , [ r15 configurationID_604 , func:AC( Spresent ) variants ( refines r11 ) ] ) , false ) ,
| ... , true , true ) ,
| IncludesBasicEvaluationInformation ( [ rqs , ... ] satisfies ( ( DOD , qr34 , hard , [ r16 configurationID_605 , func:CombineACs ( refines r12 ) ] ) , false ) ,
| ... , true , true ) ,
| IncludesBasicEvaluationInformation ( [ rqs , ... ] satisfies ( ( DOD , qr26 , hard , [ r21 configurationID_602 , func:AC( Sconsistent ) variants ] ) , false ) ,
| ... , true , true ) ,
| IncludesBasicEvaluationInformation ( [ rqs , ... ] satisfies ( ( DOD , qr27 , hard , [ r22 configurationID_603 , func:AC( Sknown ) variants ] ) , false ) ,
| ... , true , true ) ,
| IncludesBasicEvaluationInformation ( [ rqs , ... ] satisfies ( ( DOD , qr28 , hard , [ r23 configurationID_604 , func:AC( Spresent ) variants ] ) , false ) ,
| ... , true , true ) ,
| IncludesBasicEvaluationInformation ( [ rqs , ... ] satisfies ( ( DOD , qr29 , hard , [ r24 configurationID_605 , func:CombineACs ] ) , false ) ,
| ... , true , true ) ,
| ...
| ---OUTPUT---
| Resulting RQSM instance
| rqs_10
| internal present requirements:
| ( r1 func:AC( Sconsistent = Sknown = Spresent ) variants , true )
| ( r9 configurationID_602 , func:AC( Sconsistent ) variants , true )
| ( r13 configurationID_602 , func:AC( Sconsistent ) variants ( refines r9 ) , true )
| ( r10 configurationID_603 , func:AC( Sknown ) variants , true )
| ( r14 configurationID_603 , func:AC( Sknown ) variants ( refines r10 ) , true )
| ( r1 configurationID_604 , func:AC( Spresent ) variants , true )
| ( r15 configurationID_604 , func:AC( Spresent ) variants ( refines r11 ) , true )
```
I.3 Template-based Configuration

Enacted Requirements:
IsToBeSatisfied(r13 configurationID_602 , func:AC(Sconsist-Unknown-Spresent)variants)  
IsToBeSatisfied(r14 configurationID_603 , func:AC(Sconsist)variants(refines r9))  
IsToBeSatisfied(r15 configurationID_604 , func:AC(Spresent)variants(refines r11))  
IsToBeSatisfied(r16 configurationID_605 , func:CombineACs(refines r12))

Resulting Process Evaluation:
ProcessEvaluationCollection POC_20
processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

WARNING: Status—report of DPC

No. of times activated: 5

RefineSelectedTemplate, Incomplete, DOD_M

WARNING: Status—report of DPC

No. of times activated: 9

RefineSelectedTemplate, Incomplete, DOD_M

Input—

Received DODM Process Evaluation:
ProcessEvaluationCollection POC_34
processEvaluation(Strat_18, RefineSelectedTemplate, Incomplete, DODM) Incomplete, more activations are needed, or input from RQSM

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_30
processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

Output—

Strategy for component to be activated: Strat_18, RefineSelectedTemplate

Output—

Received RQSinstance:

Input—

Received overall Strategy:
Strat_18, RefineSelectedTemplate

Received RQSinstance:

Output—

Resulting Overall Dd Assessments:

Resulting RQSinstance Assessments:

| IncludesBasicEvaluationInformation(
<table>
<thead>
<tr>
<th>elements</th>
<th>satisfies (DOD_ID, qr43, hard, [r3 func:AC(Sconsistent-Spresent)variants]), true)</th>
</tr>
</thead>
<tbody>
<tr>
<td>... true, true)</td>
<td></td>
</tr>
<tr>
<td>IncludesBasicEvaluationInformation((rqs_10, ..., satisfies (DOD_ID, qr44, hard, [r1 func:AC(Sconsistent-Unknown-Spresent)variants]), true)</td>
<td></td>
</tr>
<tr>
<td>... true, true)</td>
<td></td>
</tr>
<tr>
<td>IncludesBasicEvaluationInformation((rqs_10, ..., satisfies (DOD_ID, qr45, hard, [r13 configurationID_602, func:AC(Sconsistent)variants(refines r9)]), true)</td>
<td></td>
</tr>
<tr>
<td>... true, true)</td>
<td></td>
</tr>
<tr>
<td>IncludesBasicEvaluationInformation((rqs_10, ..., satisfies (DOD_ID, qr46, hard, [r14 configurationID_603, func:AC(Sknown)variants(refines r10)]), true)</td>
<td></td>
</tr>
<tr>
<td>... true, true)</td>
<td></td>
</tr>
<tr>
<td>IncludesBasicEvaluationInformation((rqs_10, ..., satisfies (DOD_ID, qr47, hard, [r15 configurationID_604, func:AC(Spresent)variants(refines r11)]), true)</td>
<td></td>
</tr>
<tr>
<td>... true, true)</td>
<td></td>
</tr>
<tr>
<td>IncludesBasicEvaluationInformation((rqs_10, ..., satisfies (DOD_ID, qr48, hard, [r16 configurationID_605, func:CombineACs(refines r12)]), false)</td>
<td></td>
</tr>
<tr>
<td>... true, true)</td>
<td></td>
</tr>
<tr>
<td>IncludesBasicEvaluationInformation((rqs_10, ..., satisfies (DOD_ID, qr40, hard, [r25 configurationID_602, func:AC(Sconsistent)variants]), false)</td>
<td></td>
</tr>
<tr>
<td>... true, true)</td>
<td></td>
</tr>
<tr>
<td>IncludesBasicEvaluationInformation((rqs_10, ..., satisfies (DOD_ID, qr41, hard, [r26 configurationID_603, func:AC(Sknown)variants]), false)</td>
<td></td>
</tr>
<tr>
<td>... true, true)</td>
<td></td>
</tr>
<tr>
<td>IncludesBasicEvaluationInformation((rqs_10, ..., satisfies (DOD_ID, qr42, hard, [r27 configurationID_604, func:AC(Spresent)variants]), false)</td>
<td></td>
</tr>
<tr>
<td>... true, true)</td>
<td></td>
</tr>
<tr>
<td>IncludesBasicEvaluationInformation((rqs_10, ..., satisfies (DOD_ID, qr43, hard, [r28 configurationID_605, func:CombineACs]), false)</td>
<td></td>
</tr>
<tr>
<td>... true, true)</td>
<td></td>
</tr>
</tbody>
</table>

Resulting DOD:

DOD_14

composedACvariants ACvariant1(configurationID_602) filled with AC(Sconsistent)variants
ACvariant2(configurationID_603) filled with AC(Sknown)variants
ACvariant3(configurationID_604) filled with AC(Spresent)variants
CombineResults(configurationID_605) not filled

Resulting Process Evaluation:
ProcessEvaluationCollection POC_44
Trace of Single-Level Reconfiguration Example

process Evaluation (Strat_18, RefineSelectedTemplate, Incomplete, DODM) Incomplete, more activations are needed, or input from RQSM

05-Oct-2009 23:48:12 design Process Components DPC old log Status
WARNING: Status report of DPC

No. of times activated: 10

INPUT

Received DODM Process Evaluation:
Process Evaluation Collection POC_44
process Evaluation (Strat_18, RefineSelectedTemplate, Incomplete, DODM) Incomplete, more activations are needed, or input from RQSM

Received RQSM Process Evaluation:
Process Evaluation Collection POC_39
process Evaluation (Strat_18, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

OUTPUT

Strategy for component to be activated: Strat_18, RefineSelectedTemplate

WARNING: Status report of RQSM

No. of times activated: 5

INPUT

Received overall Strategy:
Strat_18, RefineSelectedTemplate
Received DOD Assessment:
{[ ] ... [IncludesBasicEvaluationInformation ({rqs} ... | satisfies ((DOD,44,hard,[r1 func:AC(Sconsistent-Sknown-Spresent)variants(refines r9)]),$true), ... ,true),true) ... [IncludesBasicEvaluationInformation ({rqs} ... | satisfies ((DOD,44,hard,[r13 configurationID_602,func:AC(Sconsistent)variants(refines r9)]),$true), ... ,true),true),true) ... [IncludesBasicEvaluationInformation ({rqs} ... | satisfies ((DOD,44,hard,[r14 configurationID_603,func:AC(Sknown)variants(refines r10)]),$true), ... ,true),true),true) ... [IncludesBasicEvaluationInformation ({rqs} ... | satisfies ((DOD,44,hard,[r15 configurationID_604,func:AC(Spresent)variants(refines r11)]),$true), ... ,true),true),true)} ...

OUTPUT

Resulting RQSt instance rqs_11
internal present requirements:
(r1 func:AC(Sconsistent-Sknown-Spresent)variants,true)
(r9 configurationID_602,func:AC(Sconsistent)variants,true)
(r13 configurationID_602,func:AC(Sconsistent)variants(refines r9),true)
(r10 configurationID_603,func:AC(Sknown)variants,true)
(r14 configurationID_603,func:AC(Sknown)variants(refines r10),true)
(r11 configurationID_604,func:AC(Spresent)variants(true)
(r12 configurationID_604,func:CombineACs(true)
(r16 configurationID_605,func:CombineACs(refines r12),true)

Enacted Requirements:
IsToBeSatisfied(r1 func:AC(Sconsistent-Sknown-Spresent)variants)
IsToBeSatisfied(r13 configurationID_602,func:AC(Sconsistent)variants(refines r9))
IsToBeSatisfied(r14 configurationID_603,func:AC(Sknown)variants(refines r10))
IsToBeSatisfied(r15 configurationID_604,func:AC(Spresent)variants(refines r11))
IsToBeSatisfied(r16 configurationID_605,func:CombineACs(refines r12))

Resulting Process Evaluation:
Process Evaluation Collection POC_49
process Evaluation (Strat_18, RefineSelectedTemplate, Succeeded, RQSM) Succeeded
I.3 Template-based Configuration

No. of times activated: 11

---INPUT---
Received DOXM Process Evaluation:
processEvaluation(Strat_18, RefineSelectedTemplate, Incomplete, DOXM) Incomplete, more activations are needed, or input from RQSM

---OUTPUT---
Strategy for component to be activated: Strat_18, RefineSelectedTemplate

No. of times activated: 6

---INPUT---
Received overall Strategy:
Strat_18, RefineSelectedTemplate
Received RQSinstance:
rqs_11

---OUTPUT---
Resulting Overall DOD Assessments:

Resulting RQSinstance Assessments:

Resulting DOX:
DOD_16 composedACvariants ACvariant1(configurationID_602) filled with AC(Sconsistent)variants
ACvariant2(configurationID_603) filled with AC(Sknown)variants
ACvariant3(configurationID_604) filled with AC(Spresent)variants
CombineResults(configurationID_605) filled with CombineACs

Resulting Process Evaluation:
processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, DOXM) Succeeded

Check pre- and post-conditions

No. of times activated: 12
Trace of Single-Level Reconfiguration Example

--- INPUT ---
Received DODM Process Evaluation:
ProcessEvaluationCollection POC_54
processEvaluation(STRAT_18, RefineSelectedTemplate, Succeeded, DODM) Succeeded

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_49
processEvaluation(STRAT_18, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

--- OUTPUT ---
Strategy for component to be activated: STRAT_51, TestPre_PostConditions

WARNING: Status--report of RQSM

No. of times activated: 6

--- INPUT ---
Received overall Strategy:
STRAT_51, TestPre_PostConditions

Received DODAssessment:

postConditions (message) true

--- OUTPUT ---
Resulting RQSinstance
rqs_13
internal present requirements:
(r1 func AC (Sconsistent $Sknown $Spresent) variants, true)
(r9 configurationID_602 func AC (Sconsistent) variants, true)
(r13 configurationID_606 func AC (Sconsistent) variants (refines r9), true)
(r14 configurationID_603 func AC (Sknown) variants (refines r10), true)
(r11 configurationID_604 func AC (Spresent) variants (refines r11), true)
(r12 configurationID_605 func CombineACs, true)
(r16 configurationID_605 func CombineACs (refines r12), true)

Enacted Requirements:
IsToBeSatisfied (r1 func AC (Sconsistent $Sknown $Spresent) variants)
IsToBeSatisfied (r14 configurationID_603 func AC (Sknown) variants (refines r11))
IsToBeSatisfied (r15 configurationID_604 func AC (Spresent) variants (refines r11))
IsToBeSatisfied (r16 configurationID_605 func CombineACs (refines r12))

Resulting Process Evaluation:
ProcessEvaluationCollection POC_59
processEvaluation(STRAT_51, TestPre_PostConditions, Succeeded, RQSM) Succeeded

05-Oct-2009 23:48:12 designProcess components DPC old logStatus
WARNING: Status--report of DPC

No. of times activated: 13

--- INPUT ---
Received DODM Process Evaluation:
ProcessEvaluationCollection POC_54
processEvaluation(STRAT_18, RefineSelectedTemplate, Succeeded, DODM) Succeeded
I.3 Template-based Configuration

Received RQSM Process Evaluation:

<table>
<thead>
<tr>
<th>ProcessEvaluationCollection</th>
<th>POC_59</th>
</tr>
</thead>
</table>

**Input**

Strategy for component to be activated: Strat_51, TestPre_PostConditions

**OUTPUT**

Received overall Strategy:

Strat_51, TestPre_PostConditions

Received RQSIM instance:

rqs_13

**OUTPUT**

Resulting Overal Dod Assessments:

[]

Resulting RQSIM instance Assessments:

[]

Received DOD:

DOD_18

composedACvariants ACvariant1(configurationID_602) filled with AC(Sconsistent)variants

ACvariant2(configurationID_603) filled with AC(Sknown)variants

ACvariant3(configurationID_604) filled with AC(Spresent)variants

Resulting Process Evaluation:

<table>
<thead>
<tr>
<th>ProcessEvaluationCollection</th>
<th>POC_64</th>
</tr>
</thead>
</table>

**Input**

Received DODM Process Evaluation:

ProcessEvaluationCollection POC_64

**OUTPUT**

Strategy for component to be activated: null

Design Finished
I.4 Integration

INFO: Starting integration
InitialConfiguration:DOD_1
classification observationConstraints (configurationID_34) filled with OCI–single–value–constraint
featureScoringMechanism (configurationID_35) filled with FS1–IEUM
macroScoringMechanism (configurationID_36) filled with MS1a–IEUM
solutionAdmissibilityCriterion (configurationID_37) filled with AC(Sconsistent–Sknown–Spresent)variants
betterMatchScores (configurationID_38) filled with BM4-highest
Failed Component = AC(Sconsistent–Sknown–Spresent)variants
INFO: Final configuration, after integration:
DOD_20
classification observationConstraints (configurationID_34) filled with OCI–single–value–constraint
featureScoringMechanism (configurationID_35) filled with FS1–IEUM
macroScoringMechanism (configurationID_36) filled with MS1a–IEUM
solutionAdmissibilityCriterion (configurationID_37) filled with composedACvariants
ACVariant1 (configurationID_602) filled with AC(Sconsistent)variants
ACVariant2 (configurationID_603) filled with AC(Sknown)variants
ACVariant3 (configurationID_604) filled with AC(Spresent)variants
CombineResults (configurationID_605) filled with CombineACs
betterMatchScores (configurationID_38) filled with BM4-highest
Appendix J

Trace of Multi-Level Reconfiguration Example

Initialisation

```ini
Settings: loading conf/math/designFiles.conf
Settings: loading conf/math/scenario/rqsInit_extended.conf
Settings: loading conf/math/scenario/dodInit.conf
Settings: loading conf/math/scenario/processObjInit.conf
[t1_slot2, t1_slot1, failedComponent, ws4_name, t1_name, empty, t2_name,
 mainBuildingBlock, ws2_name, t3_slot3, t3_slot2, t3_slot1, t2_slot2, ws3_name, t3_name
 t2_slot1, ws4_name, ws5_name]
searching for :t1_name
CurrentBBname:t1DetermineArea
retrieve slot1 of template t1
searching for :t2_name
CurrentBBname:t2DetermineHeight
retrieve slot1 of template t2
searching for :t3_name
CurrentBBname:t3DetermineSquaredHeight
retrieve slot1 of template t3
searching for :ws3_name
CurrentBBname:wsMathSquare1
retrieve slot2 of template t3
searching for :ws4_name
CurrentBBname:wsMathSquare2
retrieve slot3 of template t3
searching for :ws5_name
CurrentBBname:wsMathSubtract1
retrieve slot4 of template t3
retrieve slot2 of template t2
searching for :ws2_name
CurrentBBname:wsMathSquareroot
retrieve slot3 of template t2
retrieve slot2 of template t1
searching for :ws1_name
CurrentBBname:wsMathArea
retrieve slot3 of template t1
 t1DetermineArea sl1DetermineHeight (configurationID_53) filled with t2DetermineHeight
sl1DetermineSquaredHeight (configurationID_127) filled with t3DetermineSquaredHeight
sl1Square (configurationID_201) filled with wsMathSquare1
sl2Square (configurationID_202) filled with wsMathSquare2
sl3Subtract (configurationID_203) filled with wsMathSubtract1
sl2DetermineRoot (configurationID_128) filled with wsMathSquareroot
sl2DetermineArea (configurationID_54) filled with wsMathArea

sl1DetermineHeight (configurationID_53) filled with t2DetermineHeight
sl1DetermineSquaredHeight (configurationID_127) filled with t3DetermineSquaredHeight
sl1Square (configurationID_201) filled with wsMathSquare1
sl2Square (configurationID_202) filled with wsMathSquare2
sl3Subtract (configurationID_203) filled with wsMathSubtract1
```
The configuration starts with:

1. Determine Area (configurationID 54) filled with wsMathArea

The configurations start with:

- s2DetermineRoot (configurationID 128) filled with wsMathSquareroot
- s2DetermineArea (configurationID 54) filled with wsMathArea

Failed Component = wsMathSubtract1

INFO: Reconfiguration ticket has been generated. Nr: 1.


INFO: Retrieving Starting Information, starting collection size: 1


INFO: Information has been added


INFO: Ticket in history: 1


J.1 Focus Determination (1st)

INFO: Determining the focus

INFO: Focus has not been determined earlier


J.2 Requirement Determination (1st)

INFO: The following requirements are relevant

Requirement Set

Failing service: version_2.informationTypes.FailingWebService@e6ff0d

r1 func: Subtract
r2 post: PrecisionGreaterThanYield, true

J.3 Template-based Configuration (1st)

Initiate a new configuration and requirement set
J.3 Template-based Configuration (1st)

Excluding Building Block: wsMathSubtract1

Settings: loading conf/ibrow/designFiles.conf
Settings: loading conf/ibrow/designFiles.conf
Settings: loading conf/ibrow/designFiles.conf
rqDefinitions have been set

05-Oct-2009 23:45:27 designProcess.components.DPCold logStatus
WARNING: Status—report of DPC

No. of times activated: 1

---INPUT---
Received DDEM Process Evaluation:
ProcessEvaluationCollection POC,3

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC,2

---OUTPUT---
Strategy for component to be activated: Strat_1, GenerateMainTemplateOnFunctionality

WARNING: Status—report of DDEM

No. of times activated: 1

---INPUT---
Received overall Strategy:
Strat_1, GenerateMainTemplateOnFunctionality
Received RQSM instance:
rqs_1

---OUTPUT---
Resulting Overall Dod Assessments:

Resulting RQSInstance Assessments:

Resulting DOD:
empty DOD

Resulting Process Evaluation:
ProcessEvaluationCollection POC,4
processEvaluation(Strat_1, GenerateMainTemplateOnFunctionality, Incomplete, DDEM)
Incomplete, more activations are needed, or input from RQSM

WARNING: Status—report of DPC

No. of times activated: 2

---INPUT---
Received DDEM Process Evaluation:
ProcessEvaluationCollection POC,4
processEvaluation(Strat_1, GenerateMainTemplateOnFunctionality, Incomplete, DDEM)
Incomplete, more activations are needed, or input from RQSM

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC,2

---OUTPUT---
Strategy for component to be activated: Strat_1, GenerateMainTemplateOnFunctionality

WARNING: Status—report of RQSM

No. of times activated: 1

---INPUT---
Received overall Strategy:
Strat_1, GenerateMainTemplateOnFunctionality
Received DOD Assessment:

---OUTPUT---
Resulting RQSM instance
rqs_5
internal present requirements:
(r1 func:Subtract, true)
(r2 post:PrecisionGreaterThanYield, true, true)

Enacted Requirements:
IsToBeSatisfied(r1 func:Subtract)

Resulting Process Evaluation:
ProcessEvaluationCollection POC,9
processEvaluation(Strat_1, GenerateMainTemplateOnFunctionality, Succeeded, RQSM)
Succeeded
### Find a component satisfying the initial requirement set

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Process Components</th>
<th>Log Status</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No. of times activated: 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Received DODM Process Evaluation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProcessEvaluationCollection POC₄</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProcessEvaluation(Strat₁, GenerateMainTemplateOnFunctionality, Incomplete, DODM)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Incomplete, more activations are needed, or input from RQSM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Received RQSM Process Evaluation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProcessEvaluationCollection POC₉</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>processEvaluation(Strat₁, GenerateMainTemplateOnFunctionality, Succeeded, RQSM)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Succeeded</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**OUTPUT**

Strategy for component to be activated: Strat₁, GenerateMainTemplateOnFunctionality

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Process Components</th>
<th>Log Status</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No. of times activated: 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Received overall Strategy:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Strat₁, GenerateMainTemplateOnFunctionality</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Received RQSitnance:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>rqs₅</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUTPUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resulting Overall Dod Assessments:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resulting RQSitnance Assessments:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IncludesBasicEvaluationInformation((rqs₅,...,[satisfies((DOD₈,(qr2,hard,[r1 func:Subtract])]),true],</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[,...,true],true),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resulting DOD:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DOD₈</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>wsMathSubtract2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resulting Process Evaluation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProcessEvaluationCollection POC₉</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>processEvaluation(Strat₁, GenerateMainTemplateOnFunctionality, Succeeded, DODM)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Succeeded</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Incrementally fill open slots

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Process Components</th>
<th>Log Status</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No. of times activated: 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Received DODM Process Evaluation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProcessEvaluationCollection POC₄</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>processEvaluation(Strat₁, GenerateMainTemplateOnFunctionality, Succeeded, DODM)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Succeeded</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Received RQSM Process Evaluation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProcessEvaluationCollection POC₉</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>processEvaluation(Strat₁, GenerateMainTemplateOnFunctionality, Succeeded, RQSM)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Succeeded</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUTPUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Strategy for component to be activated: Strat₁₈, RefineSelectedTemplate</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Process Components</th>
<th>Log Status</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No. of times activated: 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
J.3 Template-based Configuration (1st)  

**INPUT**  
Received overall Strategy: Strat_18, RefineSelectedTemplate
Received DOD Assessment:
- 

**OUTPUT**  
Resulting RQSample
- internal present requirements:
  - (r1 func: Subtract, true)
  - (r2 post: PrecisionGreaterThanYield, true, true)

Enacted Requirements:
- IsToBeSatisfied(r1 func: Subtract)

Resulting Process Evaluation:
- ProcessEvaluationCollection POC_10
  - processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

---

**INPUT**  
Received overall Strategy: Strat_18, RefineSelectedTemplate
Received RQSample:
- internal present requirements:
  - (r1 func: Subtract, true)
  - (r2 post: PrecisionGreaterThanYield, true, true)

Enacted Requirements:
- IsToBeSatisfied(r1 func: Subtract)

Resulting Process Evaluation:
- ProcessEvaluationCollection POC_10
  - processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

---

**OUTPUT**  
Strategy for component to be activated: Strat_18, RefineSelectedTemplate

**INPUT**  
Received overall Strategy: Strat_18, RefineSelectedTemplate
Received RQSample:
- internal present requirements:
  - (r1 func: Subtract, true)
  - (r2 post: PrecisionGreaterThanYield, true, true)

Enacted Requirements:
- IsToBeSatisfied(r1 func: Subtract)

Resulting Process Evaluation:
- ProcessEvaluationCollection POC_10
  - processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

---

**OUTPUT**  
Resulting Overall DOD Assessments:

**INPUT**  
Received overall Strategy: Strat_18, RefineSelectedTemplate
Received RQSample:
- internal present requirements:
  - (r1 func: Subtract, true)
  - (r2 post: PrecisionGreaterThanYield, true, true)

Enacted Requirements:
- IsToBeSatisfied(r1 func: Subtract)

Resulting Process Evaluation:
- ProcessEvaluationCollection POC_10
  - processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, DODM) Succeeded

---

Check pre- and post-conditions

**INPUT**  
Received overall Strategy: Strat_18, RefineSelectedTemplate
Received RQSample:
- internal present requirements:
  - (r1 func: Subtract, true)
  - (r2 post: PrecisionGreaterThanYield, true, true)

Enacted Requirements:
- IsToBeSatisfied(r1 func: Subtract)

Resulting Process Evaluation:
- ProcessEvaluationCollection POC_10
  - processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, DODM) Succeeded

---

**INPUT**  
Received overall Strategy: Strat_18, RefineSelectedTemplate
Received RQSample:
- internal present requirements:
  - (r1 func: Subtract, true)
  - (r2 post: PrecisionGreaterThanYield, true, true)

Enacted Requirements:
- IsToBeSatisfied(r1 func: Subtract)

Resulting Process Evaluation:
- ProcessEvaluationCollection POC_10
  - processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, DODM) Succeeded
Trace of Multi-Level Reconfiguration Example

ProcessEvaluationCollection POC_19
processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

---OUTPUT---
Strat for component to be activated: Strat_27, TestPre_PostConditions

WARNING: Status—report of RQSM

No. of times activated: 3

---INPUT---
Received overall Strategy: Strat_27, TestPre_PostConditions
Received DODAssessment:

[]
[]

---OUTPUT---
Resulting RQSinstance

rq_10

Resulting Overall DOD Assessments:

[]
[]

Resulting DOD:

DOD_12

wsMathSubtract2

Resulting Process Evaluation:

ProcessEvaluationCollection POC_24
processEvaluation(Strat_18, RefineSelectedTemplate, Succeeded, DODM) Succeeded

---OUTPUT---
Strat for component to be activated: Strat_27, TestPre_PostConditions

WARNING: Status—report of DODM

No. of times activated: 4

---INPUT---
Received overall Strategy: Strat_27, TestPre_PostConditions
Received RQSinstance:

rq_10

---OUTPUT---
Resulting Overall DOD Assessments:

[]
[]

Resulting DOD:

DOD_12

wsMathSubtract2

Resulting Process Evaluation:

ProcessEvaluationCollection POC_24
processEvaluation(Strat_27, TestPre_PostConditions, Failed, DODM) Failed because no successful strategies could be found

05-Oct-2009 23:45:27 designProcess.components.DPCold logStatus
WARNING: Status—report of DPC
**J.3 Template-based Configuration (1st)**

---

**INPUT**

Received DODM Process Evaluation:

ProcessEvaluationCollection POC\_34

processEvaluation(Strat\_27, TestPre\_PostConditions, Failed, DODM) Failed because no successfull strategies could be found

Received RQSM Process Evaluation:

ProcessEvaluationCollection POC\_29

processEvaluation(Strat\_27, TestPre\_PostConditions, Succeeded, RQSM) Succeeded

---

**OUTPUT**

Strategy for component to be activated: Strat\_36, RefineSelectedTemplate

---

**INPUT**

Received Overall Strategy:

Strat\_36, RefineSelectedTemplate

Received DOD Assessment:

[ [ ] , [ IncludesBasicEvaluationInformation((rqs\_10, [... satisfies((DOD\_12, (qr8, hard, [r1 func: Subtract]), true), [... true], true),

|... true], true),

|... true], true),

Resulting RQSinstance

rqs\_13

internal present requirements:

(r1 func: Subtract, true)

(r2 post: PrecisionGreaterThanYield, true, true)

Enacted Requirements:

IsToBeSatisfied(r1 func: Subtract)

IsToBeSatisfied(r2 post: PrecisionGreaterThanYield, true)

---

**OUTPUT**

Resulting Overall Dod Assessments:

[ ]

Resulting RQSInstance Assessments:

[ IncludesBasicEvaluationInformation((rqs\_13, [... satisfies((DOD\_15, (qr10, hard, [r1 func: Subtract]), false), [... true], true),

|... true], true),

|... true], true),

Resulting DOD:
DOD
empty DOD

Resulting Process Evaluation:
ProcessEvaluationCollection POC_44
processEvaluation(Strat_46, RefineSelectedTemplate, Failed, DODM) Failed because no successful strategies could be found

05-Oct-2009 23:45:27 designProcess.components.DPC_old logStatus
WARNING: Status--report of DPC
No. of times activated: 10

---INPUT---
Received DODM Process Evaluation:
ProcessEvaluationCollection POC_44
processEvaluation(Strat_46, RefineSelectedTemplate, Failed, DODM) Failed because no successful strategies could be found

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_49
processEvaluation(Strat_46, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

---OUTPUT---
Strategy for component to be activated: Strat_45, GenerateMainTemplateOnFunctionality

WARNING: Status--report of RQSM
No. of times activated: 5

---INPUT---
Received overall Strategy:
Strat_45, GenerateMainTemplateOnFunctionality
Received DODAssessment:
[[], [IncludesBasicEvaluationInformation([rqs_13 ...] satisfies (DOD_15,(qr10,hard,[r1 func:Subtract]),false), [...,true],true),
IncludesBasicEvaluationInformation([rqs_13 ...] satisfies (DOD_15,(qr11,hard,[r2 post:PrecisionGreaterThanYield, true]),false), [...,true],true),]]

---OUTPUT---
Resulting RQSinstance
rqs_16
internal present requirements:
(r1 func:Subtract,true)
(r2 post:PrecisionGreaterThanYield,true, true)

Enacted Requirements:
IsToBeSatisfied(r1 func:Subtract)

Resulting Process Evaluation:
ProcessEvaluationCollection POC_49
processEvaluation(Strat_45, GenerateMainTemplateOnFunctionality, Succeeded, RQSM) Succeeded

05-Oct-2009 23:45:27 designProcess.components.DPC_old logStatus
WARNING: Status--report of DPC
No. of times activated: 11

---INPUT---
Received DODM Process Evaluation:
ProcessEvaluationCollection POC_44
processEvaluation(Strat_46, RefineSelectedTemplate, Failed, DODM) Failed because no successful strategies could be found

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_49
processEvaluation(Strat_45, GenerateMainTemplateOnFunctionality, Succeeded, RQSM) Succeeded

---OUTPUT---
Strategy for component to be activated: Strat_45, GenerateMainTemplateOnFunctionality

WARNING: Status--report of DODM
No. of times activated: 6

---INPUT---
Received overall Strategy:
Strat_45, GenerateMainTemplateOnFunctionality
Received RQSinstance:
J.4 Focus Determination (2nd)

INFO: Determining the focus
INFO: Focusing has been determined before
INFO: new focus =configurationID_127

J.5 Requirement Determination (2nd)

INFO: The following requirements are relevant
Requirement Set
Failing service: version_2, informationTypes, FailingWebService@e6ff0d
r7 func: DetermineSquaredHeight
J.6 Template-based Configuration (2nd)

Initiate a new configuration and requirement set

Excluding Building Block: wsMathSubtract1
05–Oct–2009 23:45:27 designProcess.components.DPC_old logStatus

WARNING: Status—report of DPC

No. of times activated: 1

—INPUT—

Received DODM Process Evaluation:
ProcessEvaluationCollection POC_62
Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_61

—OUTPUT—

Strategy for component to be activated: Strat,55, GenerateMainTemplateOnFunctionality

Settings: loading conf//ibrow/designFiles.conf
Settings: loading conf//ibrow/designFiles.conf
Settings: loading conf//ibrow/scenario//rq_all_possible.rqs.conf
rqDefinitions have been set


WARNING: Status—report of DODM

No. of times activated: 1

—INPUT—

Received overall Strategy:
Strat,55, GenerateMainTemplateOnFunctionality
Received RQSinstance:
rq,19

—OUTPUT—

Resulting Overall Dod Assessments:
[]
Resulting RQSinstance Assessments:
[]
Resulting DOD: DOD,23
empty DOD

Resulting Process Evaluation:
ProcessEvaluationCollection POC,63
processEvaluation(Strat,55, GenerateMainTemplateOnFunctionality, Incomplete, DODM)
Incomplete, more activations are needed, or input from RQSM

05–Oct–2009 23:45:27 designProcess.components.DPC_old logStatus

WARNING: Status—report of DPC

No. of times activated: 2

—INPUT—

Received DODM Process Evaluation:
ProcessEvaluationCollection POC,63
processEvaluation(Strat,55, GenerateMainTemplateOnFunctionality, Incomplete, DODM)
Incomplete, more activations are needed, or input from RQSM

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC,61

—OUTPUT—

Strategy for component to be activated: Strat,55, GenerateMainTemplateOnFunctionality


WARNING: Status—report of RQSM

No. of times activated: 1

—INPUT—

Received overall Strategy:
Strat,55, GenerateMainTemplateOnFunctionality
Received DODAssessment:
[[], []]

—OUTPUT—

Resulting RQSinstance:
rq,23
f  internal present requirements:
(r7 func:DetermineSquaredHeight, true)

Enacted Requirements:
Find a component satisfying the initial requirement set

Find a component satisfying the initial requirement set

Warning: Status report of DPC

No. of times activated: 3

--- INPUT ---
Received DODM Process Evaluation:
Process Evaluation Collection POC, 98
processEvaluation(Strat, 55, GenerateMainTemplateOnFunctionality, Incomplete, DODM)
Incomplete, more activations are needed, or input from RQSM

Received RQSM Process Evaluation:
Process Evaluation Collection POC, 98
processEvaluation(Strat, 55, GenerateMainTemplateOnFunctionality, Succeeded, RQSM)

--- OUTPUT ---
Strategy for component to be activated: Strat, 55, GenerateMainTemplateOnFunctionality

--- INPUT ---
Received overall Strategy:
Strat, 55, GenerateMainTemplateOnFunctionality

--- OUTPUT ---
Resulting Overall Dod Assessments:

Resulting RQSi instance:

Resulting Process Evaluation:
Process Evaluation Collection POC, 73
processEvaluation(Strat, 55, GenerateMainTemplateOnFunctionality, Succeeded, DODM)
Succeeded
Incrementally fill open slots

05-Oct-2009 23:45:27 designProcess components DPC old log Status

WARNING: Status report of DPC

No. of times activated: 4

INPUT

Received DOD Process Evaluation:
ProcessEvaluationCollection POC_73
processEvaluation (Strat_55, GenerateMainTemplateOnFunctionality, Succeeded, DOD)
Succeeded

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_88
processEvaluation (Strat_55, GenerateMainTemplateOnFunctionality, Succeeded, RQSM)
Succeeded

OUTPUT

Strategy for component to be activated: Strat_64, RefineSelectedTemplate

05-Oct-2009 23:45:27 designProcess components RQSM log Status

WARNING: Status report of RQSM

No. of times activated: 2

INPUT

Received overall Strategy:
Strat_64, RefineSelectedTemplate

Received DOD Assessment:
[] [. IncludesBasicEvaluationInformation (\{rqs_23, ...\} satisfies (DOD_25, (qr20, hard, [r7
func: DetermineSquaredHeight])), true),

... true), true).

IncludesBasicEvaluationInformation (\{rqs_23, ...\} satisfies (DOD_25, (qr14, hard, [r11
configurationID_1035, post: PrecisionGreaterThanYield, true])), false)

... true), true).

IncludesBasicEvaluationInformation (\{rqs_23, ...\} satisfies (DOD_25, (qr15, hard, [r12
configurationID_1035, func: Square1]), false)

... true), true).

IncludesBasicEvaluationInformation (\{rqs_23, ...\} satisfies (DOD_25, (qr16, hard, [r13
configurationID_1036, post: PrecisionGreaterThanYield, true]), false)

... true), true).

IncludesBasicEvaluationInformation (\{rqs_23, ...\} satisfies (DOD_25, (qr17, hard, [r14
configurationID_1036, func: Square1])), false)

... true), true).

IncludesBasicEvaluationInformation (\{rqs_23, ...\} satisfies (DOD_25, (qr18, hard, [r15
configurationID_1036, post: PrecisionGreaterThanYield, true]), false)

... true), true).

IncludesBasicEvaluationInformation (\{rqs_23, ...\} satisfies (DOD_25, (qr19, hard, [r16
configurationID_1037, func: Subtract])), false)

... true), true).

--- OUTPUT ---

Resulting RQS instance

rqs_25

internal present requirements:

(r7 func: DetermineSquaredHeight, true)

(r23 configurationID_1035, post: PrecisionGreaterThanYield, true (refines r17), true)

(r24 configurationID_1035, func: Square1 (refines r18), true)

(r19 configurationID_1036, post: PrecisionGreaterThanYield, true, true)

(r25 configurationID_1036, post: PrecisionGreaterThanYield, true (refines r19), true)

(r26 configurationID_1036, func: Square1 (refines r20), true)

(r21 configurationID_1037, post: PrecisionGreaterThanYield, true, true)

(r27 configurationID_1037, post: PrecisionGreaterThanYield, true (refines r21), true)

(r22 configurationID_1037, func: Subtract, true)

(r28 configurationID_1037, func: Subtract (refines r22), true)

Enacted Requirements:

1. IsToBeSatisfied (r7 func: DetermineSquaredHeight)
2. IsToBeSatisfied (r23 configurationID_1035, post: PrecisionGreaterThanYield, true (refines r17))
3. IsToBeSatisfied (r24 configurationID_1035, func: Square1 (refines r18))
4. IsToBeSatisfied (r25 configurationID_1036, post: PrecisionGreaterThanYield, true (refines r19))
5. IsToBeSatisfied (r26 configurationID_1036, func: Square1 (refines r20))
6. IsToBeSatisfied (r21 configurationID_1037, post: PrecisionGreaterThanYield, true (refines r21))
7. IsToBeSatisfied (r22 configurationID_1037, func: Subtract (refines r22))
J.6 Template-based Configuration (2nd)

Resulting Process Evaluation
ProcessEvaluationCollection POC_276

05-Oct-2009 23:45:27 designProcess.components.DPC old logStatus
WARNING: Status - report of DPC

No. of times activated: 5

INPUT:
Received DODM Process Evaluation:
ProcessEvaluationCollection POC_273

processEvaluation(Strat_64, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_278

processEvaluation(Strat_64, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

OUTPUT:
Strategy for component to be activated: Strat_64, RefineSelectedTemplate

WARNING: Status - report of DODM

No. of times activated: 3

INPUT:
Received overall Strategy:
Strat_64, RefineSelectedTemplate

Received RQSimstance:

OUTPUT:
Resulting Overall DOD Assessments:

Resulting RQSimstance Assessments:

Resulting Process Evaluation:
ProcessEvaluationCollection POC_276

05-Oct-2009 23:45:27 designProcess.components.DPC old logStatus
WARNING: Status - report of DPC

No. of times activated: 5

INPUT:
Received DODM Process Evaluation:
ProcessEvaluationCollection POC_273

processEvaluation(Strat_64, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_278

processEvaluation(Strat_64, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

OUTPUT:
Strategy for component to be activated: Strat_64, RefineSelectedTemplate

WARNING: Status - report of DODM

No. of times activated: 3

INPUT:
Received overall Strategy:
Strat_64, RefineSelectedTemplate

Received RQSimstance:

OUTPUT:
Resulting Overall DOD Assessments:

Resulting RQSimstance Assessments:

Resulting Process Evaluation:
Trace of Multi-Level Reconfiguration Example

Received DOD Process Evaluation:
ProcessEvaluationCollection POC_64, RefineSelectedTemplate, Incomplete, DODM Incomplete, more activations are needed, or input from RQSM

No. of times activated: 6

INPUT:
Received DODM Process Evaluation:
ProcessEvaluationCollection POC_64, RefineSelectedTemplate, Incomplete, DODM Incomplete, more activations are needed, or input from RQSM

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_28, processEvaluation(Strategy, r21 configurationID, true, refines r22 configurationID, false), r17 configurationID, post:PrecisionGreaterThanYield, true, refines r17 configurationID, false), r18 configurationID, func:DetermineSquaredHeight, true, refines r18 configurationID, false), r19 configurationID, post:PrecisionGreaterThanYield, true, refines r19 configurationID, false), r20 configurationID, func:DetermineSquaredHeight, true, refines r20 configurationID, false), r21 configurationID, post:PrecisionGreaterThanYield, true, refines r21 configurationID, false), r22 configurationID, func:Subtract, true), r23 configurationID, post:PrecisionGreaterThanYield, true, refines r23 configurationID, false), r24 configurationID, func:DetermineSquaredHeight, true, refines r24 configurationID, false), r25 configurationID, post:PrecisionGreaterThanYield, true, refines r25 configurationID, false), r26 configurationID, func:DetermineSquaredHeight, true, refines r26 configurationID, false)

No. of times activated: 7

OUTPUT:
Strategy for component to be activated: Strat_64, RefineSelectedTemplate

Received overall Strategy:
Strat_64, RefineSelectedTemplate
Received DODAssessment:
[] (IncludesBasicEvaluationInformation ( ( rqs_25, ... satisfies ( ( DOD_27, ( qr34, hard, ( r7 func:DetermineSquaredHeight )) true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true ), true )

No. of times activated: 3

OUTPUT:
Resulting RQInstance
rqs_27
internal present requirements:
(r7 func:DetermineSquaredHeight, true)
(r17 configurationID, 1035, post:PrecisionGreaterThanYield, true, refines r17 configurationID, true),
(r18 configurationID, 1035, func:DetermineSquaredHeight, true)
(r24 configurationID, 1035, func:DetermineSquaredHeight, true, refines r18 configurationID, true),
(r25 configurationID, 1035, post:PrecisionGreaterThanYield, true, refines r25 configurationID, true)
(r26 configurationID, 1035, func:DetermineSquaredHeight, true, refines r26 configurationID, true)
(r21 configurationID_1037, post:PrecisionGreaterThanYield, true, true)
(r22 configurationID_1037, post:PrecisionGreaterThanYield, true (refines r21), true)
(r28 configurationID_1037, func:Subtract, true)

Enacted Requirements:
 IsToBeSatisfied(r23 configurationID_1035, post:PrecisionGreaterThanYield, true (refines r17))
 IsToBeSatisfied(r24 configurationID_1035, func:Square1 (refines r18))
 IsToBeSatisfied(r25 configurationID_1036, post:PrecisionGreaterThanYield, true (refines r19))
 IsToBeSatisfied(r26 configurationID_1036, func:Square1 (refines r20))
 IsToBeSatisfied(r27 configurationID_1037, post:PrecisionGreaterThanYield, true (refines r21))
 IsToBeSatisfied(r28 configurationID_1037, func:Subtract (refines r22))

Resulting Process Evaluation:
ProcessEvaluationCollection POC_88
processEvaluation(Strat_64, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

05-Oct-2009 23:45:27 designProcess.components.DPC old logStatus
WARNING: Status report of DPC

No. of times activated: 7

INPUT
Received DDE Process Evaluation:
ProcessEvaluationCollection POC_83
processEvaluation(Strat_64, RefineSelectedTemplate, Incomplete, DDE) Incomplete, more activations are needed, or input from RQSM

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_88
processEvaluation(Strat_64, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

OUTPUT
Strategy for component to be activated: Strat_64, RefineSelectedTemplate

WARNING: Status report of DDE

No. of times activated: 4

INPUT
Received overall Strategy:
Strat_64, RefineSelectedTemplate
Received RQSiinstance: rqs_27

OUTPUT
Resulting Overall DOD Assessments:
[...]

Resulting RQSiInstance Assessments:
<table>
<thead>
<tr>
<th>IncludesBasicEvaluationInformation</th>
<th>{ ( rqs_27, ... ) [ satisfies ( ( DOD, ... ) , DOD ) , ... ] }</th>
</tr>
</thead>
<tbody>
<tr>
<td>r7, true, true</td>
<td>IncludesBasicEvaluationInformation</td>
</tr>
<tr>
<td>configurationID_1035, post:PrecisionGreaterThanYield, true (refines r17)]</td>
<td>true, false</td>
</tr>
<tr>
<td>configurationID_1035, func:Square1 (refines r18)]</td>
<td>true, false</td>
</tr>
<tr>
<td>configurationID_1036, post:PrecisionGreaterThanYield, true (refines r19)]</td>
<td>false, false</td>
</tr>
<tr>
<td>configurationID_1036, func:Square1 (refines r20)]</td>
<td>true, false</td>
</tr>
<tr>
<td>configurationID_1036, post:PrecisionGreaterThanYield, true (refines r21)]</td>
<td>true, false</td>
</tr>
<tr>
<td>configurationID_1036, func:Subtract (refines r22)]</td>
<td>false, false</td>
</tr>
<tr>
<td>configurationID_1036, post:PrecisionGreaterThanYield, true (refines r23)]</td>
<td>true, false</td>
</tr>
<tr>
<td>configurationID_1036, func:Subtract (refines r24)]</td>
<td>false, false</td>
</tr>
<tr>
<td>configurationID_1036, post:PrecisionGreaterThanYield, true (refines r25)]</td>
<td>true, false</td>
</tr>
<tr>
<td>configurationID_1036, func:Subtract (refines r26)]</td>
<td>false, false</td>
</tr>
<tr>
<td>configurationID_1036, post:PrecisionGreaterThanYield, true (refines r27)]</td>
<td>true, false</td>
</tr>
<tr>
<td>configurationID_1036, func:Subtract (refines r28)]</td>
<td>false, false</td>
</tr>
<tr>
<td>configurationID_1036, post:PrecisionGreaterThanYield, true (refines r29)]</td>
<td>true, false</td>
</tr>
<tr>
<td>configurationID_1036, func:Subtract (refines r30)]</td>
<td>false, false</td>
</tr>
<tr>
<td>configurationID_1036, post:PrecisionGreaterThanYield, true (refines r31)]</td>
<td>true, false</td>
</tr>
<tr>
<td>configurationID_1036, func:Subtract (refines r32)]</td>
<td>false, false</td>
</tr>
<tr>
<td>configurationID_1036, post:PrecisionGreaterThanYield, true (refines r33)]</td>
<td>true, false</td>
</tr>
<tr>
<td>configurationID_1036, func:Subtract (refines r34)]</td>
<td>false, false</td>
</tr>
<tr>
<td>configurationID_1036, post:PrecisionGreaterThanYield, true (refines r35)]</td>
<td>true, false</td>
</tr>
<tr>
<td>configurationID_1036, func:Subtract (refines r36)]</td>
<td>false, false</td>
</tr>
<tr>
<td>configurationID_1036, post:PrecisionGreaterThanYield, true (refines r37)]</td>
<td>true, false</td>
</tr>
<tr>
<td>configurationID_1036, func:Subtract (refines r38)]</td>
<td>false, false</td>
</tr>
</tbody>
</table>
Trace of Multi-Level Reconfiguration Example

Resulting DOD: DOD_20
- t3DetermineSquaredHeight sl1Square (configurationID_1035) filled with wsMathSquare1
- sl2Square (configurationID_1036) filled with wsMathSquare1
- sl3Subtract (configurationID_1037) not filled

Resulting Process Evaluation:
- ProcessEvaluationCollection POC_93
  - processEvaluation (Strat_64, RefineSelectedTemplate, Incomplete, DODM) Incomplete, more activations are needed, or input from RQSM

05-Oct-2009 23:45:27. designProcess components DPC_old logStatus
WARNING: Status—report of DPC

No. of times activated: 8

- INPUT
  - Received DODM Process Evaluation:
    - ProcessEvaluationCollection POC_93
      - processEvaluation (Strat_64, RefineSelectedTemplate, Incomplete, DODM) Incomplete, more activations are needed, or input from RQSM

- OUTPUT
  - Strategy for component to be activated: Strat_64, RefineSelectedTemplate

05-Oct-2009 23:45:27. designProcess components RQSM logStatus
WARNING: Status—report of RQSM

No. of times activated: 4

- INPUT
  - Received overall Strategy:
    - Strat_64, RefineSelectedTemplate
  - Received DODM Assessment:
    - || [IncludesBasicEvaluationInformation {(qs_27, . . . , satisfies ((DOD_29, (qr51, hard, (r38 configurationID_1036, func:DetermineSquaredHeight)), true), [...], true), true)}.{InsideBasicEvaluationInformation {(qs_27, . . . , satisfies ((DOD_29, (qr52, hard, (r39 configurationID_1037, post:PrecisionGreaterThanYield), true)), false), [...], true), true)}.{InsideBasicEvaluationInformation {(qs_27, . . . , satisfies ((DOD_29, (qr53, hard, (r40 configurationID_1037, func:Subtract)), false), [...], true), true)}], false)

- OUTPUT
  - Strategy for component to be activated: Strat_64, RefineSelectedTemplate
J.6 Template-based Configuration (2nd) 215

... , true ) , true ) ,

I n c l u d e s B a s i c E v a l u a t i o n I n f o r m a t i o n ( ( rqs 27 , . . . [ s a t i s f i e s ( ( DOD 29 , ( qr53 , hard , [ r40 configurationID 1037 , func: Subtract ] ) ) , f a l s e ) ,

... , true ) , true ) ,

--- OUTPUT ---

R e s u l t i n g R Q S i n s t a n c e

rqs 28

in t e r n a l p r e s e n t r e q u i r e m e n t s :

( r7 configurationID 1035 , post: PrecisionGreaterThanYield , true ( r e f i n e s r17 ) ) ,

( r23 configurationID 1035 , post: PrecisionGreaterThanYield , true ( r e f i n e s r17 ) , true ) ,

( r24 configurationID 1035 , func: Square1 ( r e f i n e s r18 ) , true ) ,

( r25 configurationID 1036 , post: PrecisionGreaterThanYield , true ( r e f i n e s r19 ) , true ) ,

( r20 configurationID 1036 , func: Square1 , true ) ,

( r26 configurationID 1036 , func: Square1 ( r e f i n e s r20 ) , true ) ,

( r21 configurationID 1037 , post: PrecisionGreaterThanYield , true ( r e f i n e s r21 ) , true ) ,

( r27 configurationID 1037 , post: PrecisionGreaterThanYield , true ( r e f i n e s r21 ) , true ) ,

( r28 configurationID 1037 , func: Subtract ( r e f i n e s r22 ) , true )

E n a c t e d R e q u i r e m e n t s:

I s T o B e S a t i s f i e d ( r7 func: DetermineSquaredHeight )

I s T o B e S a t i s f i e d ( r23 configurationID 1035 , post: PrecisionGreaterThanYield , true ( r e f i n e s r17 ) )

I s T o B e S a t i s f i e d ( r24 configurationID 1035 , func: Square1 ( r e f i n e s r18 ) )

I s T o B e S a t i s f i e d ( r25 configurationID 1036 , post: PrecisionGreaterThanYield , true ( r e f i n e s r19 ) )

I s T o B e S a t i s f i e d ( r20 configurationID 1036 , func: Square1 )

I s T o B e S a t i s f i e d ( r26 configurationID 1036 , func: Square1 ( r e f i n e s r20 ) )

I s T o B e S a t i s f i e d ( r21 configurationID 1037 , post: PrecisionGreaterThanYield , true ( r e f i n e s r21 ) )

I s T o B e S a t i s f i e d ( r27 configurationID 1037 , post: PrecisionGreaterThanYield , true ( r e f i n e s r21 ) )

I s T o B e S a t i s f i e d ( r28 configurationID 1037 , func: Subtract ( r e f i n e s r22 ) )

R e s u l t i n g P r o c e s s E v a l u a t i o n:

P r o c e s s E v a l u a t i o n C o l l e c t i o n POC 09

processEvaluation ( Strat 64 , RefineSelectedTemplate , Succeeded , RQSM) Succeeded

05 - O c t - 2009 23:45:27 designProcess.components.DPC obsolete logStatus

W A R N I N G : S t a t u s - r e p o r t o f DPC

--- INPUT ---

R e c e i v e d D O M P r o c e s s E v a l u a t i o n:

P r o c e s s E v a l u a t i o n C o l l e c t i o n POC 09

processEvaluation ( Strat 64 , RefineSelectedTemplate , Incomplete , DOM) Incomplete , more activations are needed , or input from RQSM

R e c e i v e d R Q S M P r o c e s s E v a l u a t i o n:

P r o c e s s E v a l u a t i o n C o l l e c t i o n POC 09

processEvaluation ( Strat 64 , RefineSelectedTemplate , Succeeded , RQSM) Succeeded

--- OUTPUT ---

S t r a t e g y f o r c o m p o n e n t t o b e a c t i v a t e d : Strat 64 , RefineSelectedTemplate

05 - O c t - 2009 23:45:27 designProcess.components.DOM logStatus

W A R N I N G : S t a t u s - r e p o r t o f DOM

--- INPUT ---

R e c e i v e d o v e r a l l S t r a t e g y : Strat 64 , RefineSelectedTemplate

R e c e i v e d R Q S i n s t a n c e :

rqs 28

--- OUTPUT ---

R e s u l t i n g O v e r a l l D o d A s s e s s m e n t s:

[ ]

R e s u l t i n g R Q S i n s t a n c e A s s e s s m e n t s:

[ IncludesBasicEvaluationInformation ( [ rqs 28 , . . . [ s a t i s f i e s ( ( DOD 31 , ( qr74 , hard , [ r7 func: DetermineSquaredHeight ] ) ) , true ) ,

... , true ) , true ).

IncludesBasicEvaluationInformation ( [ rqs 28 , . . . [ s a t i s f i e s ( ( DOD 31 , ( qr75 , hard , [ r23 configurationID 1035 , post: PrecisionGreaterThanYield , true ( r e f i n e s r17 ) ] ) ] ) , false ) ,

... , true ) , true ).

IncludesBasicEvaluationInformation ( [ rqs 28 , . . . [ s a t i s f i e s ( ( DOD 31 , ( qr76 , hard , [ r24 configurationID 1035 , func: Square1 ( r e f i n e s r18 ) ] ) ) , true ) ,

... , true ) , true ).

IncludesBasicEvaluationInformation ( [ rqs 28 , . . . [ s a t i s f i e s ( ( DOD 31 , ( qr77 , hard , [ r25 configurationID 1036 , post: PrecisionGreaterThanYield , true ( r e f i n e s r19 ) ] ) ) , false ) ,

... , true ) , true ).

]
Check pre- and post-conditions

WARNING: Status

No. of times activated: 10

Refined Selected Template, Succeeded, DODM Succeeded

Post Conditions

Refined Selected Template, Succeeded, RQSM Succeeded

216 Trace of Multi-Level Reconfiguration Example
J.6 Template-based Configuration (2nd)

[ ... true ), true ).

IncludesBasicEvaluationInformation ([ rqs_28 ... ] satisfies ( ( DOD_31, ( qr79, hard, [ r27 configurationID_1037, post: PrecisionGreaterThanYield, true ( refines r21 ) ] ) ) ) ) , false ),

... true , true ).

IncludesBasicEvaluationInformation ([ rqs_28 ... ] satisfies ( ( DOD_31, ( qr80, hard, [ r28 configurationID_1037, func: Subtract ( refines r22 ) ] ) ) ) ) , true ),

... true , true ).

IncludesBasicEvaluationInformation ([ rqs_28 ... ] satisfies ( ( DOD_31, ( qr68, hard, [ r41 configurationID_1035, post: PrecisionGreaterThanYield, true ) ] ) ) ) , false ),

... true , true ).

IncludesBasicEvaluationInformation ([ rqs_28 ... ] satisfies ( ( DOD_31, ( qr69, hard, [ r42 configurationID_1035, func: Square1 ] ) ) ) ) , false ),

... true , true ).

IncludesBasicEvaluationInformation ([ rqs_28 ... ] satisfies ( ( DOD_31, ( qr70, hard, [ r43 configurationID_1035, post: PrecisionGreaterThanYield, true ) ] ) ) ) , false ),

... true , true ).

IncludesBasicEvaluationInformation ([ rqs_28 ... ] satisfies ( ( DOD_31, ( qr71, hard, [ r44 configurationID_1036, func: Square1 ] ) ) ) ) , false ),

... true , true ).

IncludesBasicEvaluationInformation ([ rqs_28 ... ] satisfies ( ( DOD_31, ( qr72, hard, [ r45 configurationID_1037, post: PrecisionGreaterThanYield, true ) ] ) ) ) , false ),

... true , true ).

IncludesBasicEvaluationInformation ([ rqs_28 ... ] satisfies ( ( DOD_31, ( qr73, hard, [ r46 configurationID_1037, func: Subtract ] ) ) ) ) , false ),

... true , true ).

]

OUTPUT

Resulting RQsInstance

rqs_30

internal present requirements:

(r7 func:DetermineSquaredHeight, true)

(r17 configurationID_1035, post:PrecisionGreaterThanYield, true, true)

(r23 configurationID_1035, post:PrecisionGreaterThanYield, true ( refines r17 ), true)

(r18 configurationID_1035, func:Square1, true)

(r24 configurationID_1035, func:Square1 ( refines r18 ), true)

(r26 configurationID_1036, post:PrecisionGreaterThanYield, true ( refines r19 ), true)

(r20 configurationID_1036, func:Square1, true)

(r26 configurationID_1036, func:Square1 ( refines r20 ), true)

(r21 configurationID_1037, post:PrecisionGreaterThanYield, true, true)

(r27 configurationID_1037, post:PrecisionGreaterThanYield, true ( refines r21 ), true)

(r22 configurationID_1037, func:Subtract, true)

(r28 configurationID_1037, func:Subtract ( refines r22 ), true)

Enacted Requirements:

IsToBeSatisfied (r7 func:DetermineSquaredHeight)

IsToBeSatisfied (r23 configurationID_1035, post:PrecisionGreaterThanYield, true ( refines r17 ))

IsToBeSatisfied (r24 configurationID_1035, func:Square1 ( refines r18 ))

IsToBeSatisfied (r25 configurationID_1036, post:PrecisionGreaterThanYield, true ( refines r19 ))

IsToBeSatisfied (r26 configurationID_1036, func:Square1 ( refines r20 ))

IsToBeSatisfied (r27 configurationID_1037, post:PrecisionGreaterThanYield, true ( refines r21 ))

IsToBeSatisfied (r28 configurationID_1037, func:Subtract ( refines r22 ))

Resulting Process Evaluation:

ProcessEvaluationCollection POC_108

processEvaluation ( Strat_89 , TestPre_PostConditions , Succeeded , RQSM ) Succeeded


WARNING Status report of DPC

No. of times activated: 11

RESULT——

Received DODM Process Evaluation:

ProcessEvaluationCollection POC_103

processEvaluation ( Strat_64 , RefineSelectedTemplate , Succeeded , DODM ) Succeeded

Received RQSM Process Evaluation:

ProcessEvaluationCollection POC_108

processEvaluation ( Strat_89 , TestPre_PostConditions , Succeeded , RQSM ) Succeeded

——OUTPUT——

Strategy for component to be activated: Strat_89 , TestPre_PostConditions


WARNING Status report of DODM

No. of times activated: 6

——INPUT——

Received overall Strategy:
Trace of Multi-Level Reconfiguration Example

Received RQSinstance: rqs_30

RESULTING Overall Dod Assessments:

* Resulting RQSinstance Assessments:*

- Includes Basic Evaluation Information (rqs_30, satisfies ((DOD_33, (qr101, hard, [r7 configurationID_1035, post:PrecisionGreaterThanYield, true (refines r17)])), true), ...
- Includes Basic Evaluation Information (rqs_30, satisfies ((DOD_33, (qr102, hard, [r23 configurationID_1035, post:PrecisionGreaterThanYield, true (refines r18)])), true), ...
- Includes Basic Evaluation Information (rqs_30, satisfies ((DOD_33, (qr103, hard, [r24 configurationID_1035, func:DetermineSquaredHeight (refines r19)])), true), ...
- Includes Basic Evaluation Information (rqs_30, satisfies ((DOD_33, (qr104, hard, [r25 configurationID_1036, post:PrecisionGreaterThanYield, true (refines r20)])), true), ...
- Includes Basic Evaluation Information (rqs_30, satisfies ((DOD_33, (qr105, hard, [r26 configurationID_1036, func:Square1 (refines r21)])), true), ...
- Includes Basic Evaluation Information (rqs_30, violates ((DOD_33, (qr106, hard, [r27 configurationID_1036, post:PrecisionGreaterThanYield, true (refines r22)])), true), ...
- Includes Basic Evaluation Information (rqs_30, satisfies ((DOD_33, (qr107, hard, [r28 configurationID_1037, func:Square1 (refines r23)])), true), ...
- Includes Basic Evaluation Information (rqs_30, satisfies ((DOD_33, (qr108, hard, [r29 configurationID_1037, post:PrecisionGreaterThanYield, true (refines r24)])), true), ...
- Includes Basic Evaluation Information (rqs_30, satisfies ((DOD_33, (qr109, hard, [r47 configurationID_1038, post:PrecisionGreaterThanYield, true (refines r25)])), true), ...
- Includes Basic Evaluation Information (rqs_30, satisfies ((DOD_33, (qr110, hard, [r48 configurationID_1038, func:Square1 (refines r26)])), true), ...
- Includes Basic Evaluation Information (rqs_30, satisfies ((DOD_33, (qr111, hard, [r49 configurationID_1038, post:PrecisionGreaterThanYield, true (refines r27)])), true), ...
- Includes Basic Evaluation Information (rqs_30, satisfies ((DOD_33, (qr112, hard, [r50 configurationID_1038, func:Square1 (refines r28)])), true), ...
- Includes Basic Evaluation Information (rqs_30, violates ((DOD_33, (qr113, hard, [r51 configurationID_1039, post:PrecisionGreaterThanYield, true (refines r29)])), true), ...
- Includes Basic Evaluation Information (rqs_30, satisfies ((DOD_33, (qr114, hard, [r52 configurationID_1039, func:Square1 (refines r30)])), true), ...

Resulting DOD:

- DOD_33
  - t3DetermineSquaredHeight
  - s11Square (configurationID_1035) filled with wsMathSquare1
  - s12Square (configurationID_1036) filled with wsMathSquare1
  - s13Subtract (configurationID_1037) filled with wsMathSubtract2

RESULTING Process Evaluation:

- ProcessEvaluationCollection POC_113
  - processEvaluation (Strat_89, TestPrePostConditions, Failed, DODM) Failed because no successful strategies could be found

Incrementally fill open slots (backtracked)
J.6 Template-based Configuration (2nd) 219

No. of times activated: 6

INPUT

Received overall Strategy: RefineSelectedTemplate

Received DOD Assessment:

[...]

IncludesBasicEvaluationInformation (( qrs_30 ... satisfies ( DOD_33, ( qr101, hard, ( r7... ) ) ) ) , true ), [... , true], true )

IncludesBasicEvaluationInformation (( qrs_30 ... satisfies ( DOD_33, ( qr102, hard, ( r23... ) ) ) ) , true ), [... , true], true )

IncludesBasicEvaluationInformation (( qrs_30 ... satisfies ( DOD_33, ( qr103, hard, ( r24... ) ) ) ) , true ), [... , true], true )

IncludesBasicEvaluationInformation (( qrs_30 ... satisfies ( DOD_33, ( qr104, hard, ( r25... ) ) ) ) , true ), [... , true], true )

IncludesBasicEvaluationInformation (( qrs_30 ... satisfies ( DOD_33, ( qr105, hard, ( r26... ) ) ) ) , true ), [... , true], true )

IncludesBasicEvaluationInformation (( qrs_30 ... satisfies ( DOD_33, ( qr106, hard, ( r27... ) ) ) ) , true ), [... , true], true )

IncludesBasicEvaluationInformation (( qrs_30 ... satisfies ( DOD_33, ( qr107, hard, ( r28... ) ) ) ) , true ), [... , true], true )

IncludesBasicEvaluationInformation (( qrs_30 ... satisfies ( DOD_33, ( qr108, hard, ( r29... ) ) ) ) , true ), [... , true], true )

IncludesBasicEvaluationInformation (( qrs_30 ... satisfies ( DOD_33, ( qr109, hard, ( r30... ) ) ) ) , true ), [... , true], true )

IncludesBasicEvaluationInformation (( qrs_30 ... satisfies ( DOD_33, ( qr110, hard, ( r31... ) ) ) ) , true ), [... , true], true )

IncludesBasicEvaluationInformation (( qrs_30 ... satisfies ( DOD_33, ( qr111, hard, ( r32... ) ) ) ) , true ), [... , true], true )

IncludesBasicEvaluationInformation (( qrs_30 ... satisfies ( DOD_33, ( qr112, hard, ( r33... ) ) ) ) , true ), [... , true], true )

INPUT

Resulting RQSM instance

qrs_33 internal present requirements:

(r7 func: DetermineSquaredHeight, true)

(r17 configurationID_1035, post: PrecisionGreaterThanYield, true, true)

(r23 configurationID_1035, post: PrecisionGreaterThanYield, true (refines r17), true)

(r24 configurationID_1035, func: Subtract (refines r18), true)

(r19 configurationID_1036, post: PrecisionGreaterThanYield, true, true)

(r25 configurationID_1036, post: PrecisionGreaterThanYield, true (refines r19), true)

(r20 configurationID_1036, func: Subtract (refines r20), true)

(r26 configurationID_1036, func: Subtract (refines r20), true)

(r21 configurationID_1037, post: PrecisionGreaterThanYield, true, true)

(r27 configurationID_1037, post: PrecisionGreaterThanYield, true (refines r21), true)

(r22 configurationID_1037, func: Subtract (refines r22), true)

(r28 configurationID_1037, func: Subtract (refines r22), true)

Enacted Requirements:

IsToBeSatisfied (r7 func: DetermineSquaredHeight)

IsToBeSatisfied (r17 configurationID_1035, post: PrecisionGreaterThanYield, true (refines r17))

IsToBeSatisfied (r23 configurationID_1035, func: Subtract (refines r18))

IsToBeSatisfied (r24 configurationID_1035, func: Subtract (refines r18))

IsToBeSatisfied (r25 configurationID_1036, post: PrecisionGreaterThanYield, true (refines r19))

IsToBeSatisfied (r26 configurationID_1036, func: Subtract (refines r20))

IsToBeSatisfied (r27 configurationID_1037, post: PrecisionGreaterThanYield, true (refines r21))

IsToBeSatisfied (r28 configurationID_1037, func: Subtract (refines r22))

Resulting Process Evaluation:

ProcessEvaluationCollection: POC_118

processEvaluation (Strat_98, RefineSelectedTemplate, Succeeded, RQSM) Succeeded
Find a component satisfying the initial requirement set (backtracked)
---OUTPUT---
Strategy for component to be activated: Strat_107, GenerateMainTemplateOnFunctionality

WARNING: Status-report of RQSM

No. of times activated: 7

---INPUT---
Received overall Strategy:
Strat_107, GenerateMainTemplateOnFunctionality

Received DODAssessment:

[[[ [IncludesBasicEvaluationInformation([[rqs_33, ..., satisfies((DOD_36, (qr108, hard, [r7
func: DetermineSquaredHeight)]) ]), false), [ ... true]), true]

IncludesBasicEvaluationInformation([[rqs_33, ..., satisfies((DOD_36, (qr109, hard, [r23
configurationID_1035, post: PrecisionGreaterThanYield, true (refines r17)]) ]), false), [ ... true]), true]

IncludesBasicEvaluationInformation([[rqs_33, ..., satisfies((DOD_36, (qr110, hard, [r24
configurationID_1035, func: Square1 (refines r18)]) ]), false), [ ... true]), true]

IncludesBasicEvaluationInformation([[rqs_33, ..., satisfies((DOD_36, (qr111, hard, [r25
configurationID_1036, post: PrecisionGreaterThanYield, true (refines r19)]) ]), false), [ ... true]), true]

IncludesBasicEvaluationInformation([[rqs_33, ..., satisfies((DOD_36, (qr112, hard, [r26
configurationID_1036, func: Square1 (refines r20)]) ]), false), [ ... true]), true]

IncludesBasicEvaluationInformation([[rqs_33, ..., satisfies((DOD_36, (qr113, hard, [r27
configurationID_1037, post: PrecisionGreaterThanYield, true (refines r21)]) ]), false), [ ... true]), true]

IncludesBasicEvaluationInformation([[rqs_33, ..., satisfies((DOD_36, (qr114, hard, [r28
configurationID_1037, func: Subtract (refines r22)]) ]), false), [ ... true]), true]]

---OUTPUT---
Resulting RQSinstance
rqs_36

internal present requirements:
(r7 func: DetermineSquaredHeight, true)

Enacted Requirements:
IsToBeSatisfied(r7 func: DetermineSquaredHeight)

Resulting Process Evaluation:
ProcessEvaluationCollection POC_128
processEvaluation(Strategy_107, GenerateMainTemplateOnFunctionality, Succeeded, RQSM)

Succeeded

WARNING: Status-report of DODM

No. of times activated: 15

---INPUT---
Received DODM Process Evaluation:
ProcessEvaluationCollection POC_128
processEvaluation(Strategy_98, RefineSelectedTemplate, Failed, DODM) Failed because no successful strategies could be found

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_128
processEvaluation(Strategy_107, GenerateMainTemplateOnFunctionality, Succeeded, RQSM)

Succeeded

---OUTPUT---
Strategy for component to be activated: Strat_107, GenerateMainTemplateOnFunctionality

WARNING: Status-report of DODM

No. of times activated: 8

---INPUT---
Received overall Strategy:
Strat_107, GenerateMainTemplateOnFunctionality

Received RQSinstance:
rqs_36

---OUTPUT---
Resulting Overall DOD Assessments:

Resulting RQSinstance Assessments:

[]

IncludesBasicEvaluationInformation([[rqs_36, ..., satisfies((DOD_38, (qr122, hard, [r7
func: DetermineSquaredHeight)]) ]), true])

[ ... true]), true]
Incrementally fill open slots

05-Oct-2009 23:45:27 design_process components DPC cmd logStatus WARNING: Status report of DPC

No. of times activated: 16

INPUT:
Received DOD Process Evaluation:
ProcessEvaluationCollection POC_133
processEvaluation (Strat_107, GenerateMainTemplateOnFunctionality, Succeeded, DODM)
Succeeded

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_128
processEvaluation (Strat_107, GenerateMainTemplateOnFunctionality, Succeeded, RQSM)
Succeeded

OUTPUT:
Strategy for component to be activated: Strat_116, RefineSelectedTemplate

05-Oct-2009 23:45:27 design_process components RQSM logStatus WARNING: Status report of RQSM

No. of times activated: 8

INPUT:
Received overall Strategy:
Strat_116, RefineSelectedTemplate

[ ] [ IncludesBasicEvaluationInformation ([ rqs_36, ... [ satisfies ([ DOD_38, ( qr_16, hard, [ r53
configurationID_1039, post:PrecisionGreaterThanYield, true ] ) ] ), false ) ] ] [ true ] ]

[ ] [ IncludesBasicEvaluationInformation ([ rqs_36, ... [ satisfies ([ DOD_38, ( qr_117, hard, [ r54
configurationID_1039, func: Add ] ) ] ), false ) ] ] [ true ] ]

[ ] [ IncludesBasicEvaluationInformation ([ rqs_36, ... [ satisfies ([ DOD_38, ( qr_118, hard, [ r55
configurationID_1040, post:PrecisionGreaterThanYield, true ] ) ] ), false ) ] ] [ true ] ]

[ ] [ IncludesBasicEvaluationInformation ([ rqs_36, ... [ satisfies ([ DOD_38, ( qr_120, hard, [ r57
configurationID_1040, func: Subtract ] ) ] ), false ) ] ] [ true ] ]

[ ] [ IncludesBasicEvaluationInformation ([ rqs_36, ... [ satisfies ([ DOD_38, ( qr_121, hard, [ r58
configurationID_1041, func: Multiply ] ) ] ), false ) ] ] [ true ] ]
Resulting RQS instance:

\[
\text{Resulting RQS instance:}
\]

\[
\text{internal present requirements:}
\]

\[
\begin{align*}
( & \text{configID: 1039, post: Precision Greater Than Yield, true, true) } \\
( & \text{configID: 1039, post: Precision Greater Than Yield, true (refines r59), true) } \\
( & \text{configID: 1039, func: Add (refines r60), true) } \\
( & \text{configID: 1040, post: Precision Greater Than Yield, true, true) } \\
( & \text{configID: 1040, post: Precision Greater Than Yield, true (refines r61), true) } \\
( & \text{configID: 1041, post: Precision Greater Than Yield, true, true) } \\
( & \text{configID: 1041, post: Precision Greater Than Yield, true (refines r62), true) } \\
( & \text{configID: 1041, func: Multiply (refines r64), true) }
\end{align*}
\]

Enacted Requirements:

\[
\text{Is To Be Satisfied (r7 func: DetermineSquaredHeight)}
\]

\[
\text{Is To Be Satisfied (r65 configID 1039, post: Precision Greater Than Yield, true (refines r59), true)}
\]

\[
\text{Is To Be Satisfied (r66 configID 1039, func: Add (refines r60), true)}
\]

\[
\text{Is To Be Satisfied (r67 configID 1040, post: Precision Greater Than Yield, true, true)}
\]

\[
\text{Is To Be Satisfied (r68 configID 1040, func: Subtract (refines r62), true)}
\]

\[
\text{Is To Be Satisfied (r69 configID 1041, post: Precision Greater Than Yield, true (refines r63), true)}
\]

\[
\text{Is To Be Satisfied (r70 configID 1041, func: Multiply (refines r64), true)}
\]

Resulting Process Evaluation:

\[
\text{Process Evaluation Collection POC 138}
\]

\[
\text{processEvaluation (Strat 116, RefineSelectedTemplate, Succeeded, RQSM) Succeeded}
\]

05-Oct-2009 23:45:27 designProcess.components.DOD.m3 logStatus

WARNING: Status report of DPC

No. of times activated: 17

Received DODM Process Evaluation:

\[
\text{Process Evaluation Collection POC 133}
\]

\[
\text{processEvaluation (Strat 107, GenerateMainTemplateOnFunctionality, Succeeded, DODM) Succeeded}
\]

Received RQSM Process Evaluation:

\[
\text{Process Evaluation Collection POC 138}
\]

\[
\text{processEvaluation (Strat 116, RefineSelectedTemplate, Succeeded, RQSM) Succeeded}
\]

Strategy for component to be activated: Strat 116, RefineSelectedTemplate

05-Oct-2009 23:45:27 designProcess.components.DOD.m3 logStatus

WARNING: Status report of DODM

No. of times activated: 9

Received overall Strategy:

\[
\text{Strat 116, RefineSelectedTemplate}
\]

Received RQS instance:

\[
\text{rqs_39}
\]

Resulting Overall DOD Assessments:

\[
\text{Resulting RQS instance Assessments:}
\]

\[
\begin{align*}
( & \text{Includes Basic Evaluation Information ((rqs_39, ..., satisfies ((DOD_40, qr136, hard, [r7 func: DetermineSquaredHeight])}, true) } \\
( & \text{..., true), true) } \\
( & \text{Includes Basic Evaluation Information ((rqs_39, ..., satisfies ((DOD_40, qr137, hard, [r65 configurationID 1039, post: Precision Greater Than Yield], true (refines r59), true) }, false) } \\
( & \text{..., true), true) } \\
( & \text{Includes Basic Evaluation Information ((rqs_39, ..., satisfies ((DOD_40, qr138, hard, [r66 configurationID 1039, func: Add (refines r60), true) }, false) } \\
( & \text{..., true), true) } \\
( & \text{Includes Basic Evaluation Information ((rqs_39, ..., satisfies ((DOD_40, qr140, hard, [r68 configurationID 1040, post: Precision Greater Than Yield, true (refines r61), true) }, false) } \\
( & \text{..., true), true) } \\
( & \text{Includes Basic Evaluation Information ((rqs_39, ..., satisfies ((DOD_40, qr141, hard, [r67 configurationID 1040, func: Subtract (refines r62), true) }, false) } \\
( & \text{..., true), true) }
\end{align*}
\]
Trace of Multi-Level Reconfiguration Example

```plaintext
Resulting DOD:
DOD_40
1DetermineSquaredHeight sl_1Add(configurationID_1039) filled with wsMathAdd
sl_2Subtract(configurationID_1040) not filled
sl_3Multiply(configurationID_1041) not filled

Resulting Process Evaluation:
ProcessEvaluationCollection POC_143
processEvaluation(Strategy_116, RefineSelectedTemplate, Incomplete, DODM) Incomplete, more activations are needed, or input from RQSM

05-Oct-2009 23:45:27 designProcess components DPC old log Status
WARNING: Status=report of DPC

No. of times activated: 18

---INPUT---
Received DODM Process Evaluation:
ProcessEvaluationCollection POC_143
processEvaluation(Strategy_116, RefineSelectedTemplate, Incomplete, DODM) Incomplete, more activations are needed, or input from RQSM

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_138
processEvaluation(Strategy_116, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

---OUTPUT---
Strategy for component to be activated: Strategy_116, RefineSelectedTemplate

05-Oct-2009 23:45:27 designProcess components RQSM log Status
WARNING: Status=report of RQSM

No. of times activated: 9

---INPUT---
Received overall Strategy:
Strategy_116, RefineSelectedTemplate
Received DOD Assessment:
[] (IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr136, hard, r7 configurationID_1039, post:PrecisionGreaterThanYield, true \text{refines r60} \}\} \right), false), \ldots, true), true),
IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr137, hard, r65 configurationID_1039, post:PrecisionGreaterThanYield, true \text{refines r59} \}\} \right), false), \ldots, true), true),
IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr138, hard, r66 configurationID_1039, func:Add \text{refines r60} \}\} \right), true), \ldots, true), true),
IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr139, hard, r67 configurationID_1040, post:PrecisionGreaterThanYield, true \text{refines r61} \}\} \right), false), \ldots, true), true),
IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr140, hard, r68 configurationID_1040, func:Subtract \text{refines r62} \}\} \right), false), \ldots, true), true),
IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr141, hard, r69 configurationID_1041, post:PrecisionGreaterThanYield, true \text{refines r63} \}\} \right), false), \ldots, true), true),
IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr142, hard, r70 configurationID_1041, func:Multiply \text{refines r64} \}\} \right), false), \ldots, true), true),
```

---WARNING---
No. of times activated: 9
---INPUT---
Received overall Strategy:
Strategy_116, RefineSelectedTemplate
Received DOD Assessment:
[] (IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr136, hard, r7 configurationID_1039, post:PrecisionGreaterThanYield, true \text{refines r60} \}\} \right), false), \ldots, true), true),
IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr137, hard, r65 configurationID_1039, post:PrecisionGreaterThanYield, true \text{refines r59} \}\} \right), false), \ldots, true), true),
IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr138, hard, r66 configurationID_1039, func:Add \text{refines r60} \}\} \right), true), \ldots, true), true),
IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr139, hard, r67 configurationID_1040, post:PrecisionGreaterThanYield, true \text{refines r61} \}\} \right), false), \ldots, true), true),
IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr140, hard, r68 configurationID_1040, func:Subtract \text{refines r62} \}\} \right), false), \ldots, true), true),
IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr141, hard, r69 configurationID_1041, post:PrecisionGreaterThanYield, true \text{refines r63} \}\} \right), false), \ldots, true), true),
IncludesBasicEvaluationInformation(\{ rqs_39 \ldots \text{satisfies} \left( \text{DOD_40, \{ qr142, hard, r70 configurationID_1041, func:Multiply \text{refines r64} \}\} \right), false), \ldots, true), true),
```
J.6 Template-based Configuration (2nd)

```
[... true], true).

Inclu... (DOD40, (qr130, hard, [r71 configurationID1039, post:PrecisionGreaterThanYield, true]), false), [... true], true).

Inclu... (DOD40, (qr131, hard, [r72 configurationID1039, func:Add]), false), [... true], true).

Inclu... (DOD40, (qr132, hard, [r73 configurationID1040, post:PrecisionGreaterThanYield, true]), false), [... true], true).

Inclu... (DOD40, (qr133, hard, [r74 configurationID1040, func:Subtract]), false), [... true], true).

Inclu... (DOD40, (qr134, hard, [r75 configurationID1041, post:PrecisionGreaterThanYield, true]), false), [... true], true).

Inclu... (DOD40, (qr135, hard, [r76 configurationID1041, func:Multiply]), false), [... true], true).

−−− OUTPUT −−−

Resulting RQSinstance

rq...[116, RefineSelectedTemplate, Succeeded, RQSM] Succeeded

05−Oct−2009 23:45:27 designProcess.components.DOD logStatus

WARNING Status−report of DPC

No. of times activated: 19

−−− INPUT −−−

Received DOD Process Evaluation:

ProcessEvaluationCollection POC148

processEvaluation(Strat116, RefineSelectedTemplate, Incomplete, DOD) Incomplete, more activations are needed, or input from RQSM

Received RQSM Process Evaluation:

ProcessEvaluationCollection POC148

processEvaluation(Strat116, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

−−− OUTPUT −−−

Strategy for component to be activated: Strat116, RefineSelectedTemplate

05−Oct−2009 23:45:27 designProcess.components.DOXE logStatus

WARNING Status−report of DOXE

No. of times activated: 10

−−− INPUT −−−

Received overall Strategy:

Strat116, RefineSelectedTemplate

Received RQInstance:

rq...[116, RefineSelectedTemplate

−−− OUTPUT −−−

Resulting Overall Dod Assessments:
Trace of Multi-Level Reconfiguration Example

Resulting RQSM Instance Assessments:
- IncludesBasicEvaluationInformation somewsMathAdd (configurationID_1039) filled with wsMathAdd
- IncludesBasicEvaluationInformation somewsMathSubtract (configurationID_1040) filled with wsMathSubtract
- DeterminedSquaredHeight sl1Add (configurationID_1039) filled with wsMathAdd
- Subtract (configurationID_1040) post:PrecisionGreaterThanYield, true (refines r61)
- DeterminesSquaredHeight sl3Multiply (configurationID_1041) post:PrecisionGreaterThanYield, true (refines r62)

Resulting DOD:
DOD_42

Oct 2009 23:45:27 designProcess.components DPC old logStatus

No. of times activated: 20

 Kota

Oct 2009 23:45:27 designProcess.components RQSM logStatus

No. of times activated: 10

OUTPUT

Strategy for component to be activated: Strat_116, RefineSelectedTemplate

226
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr157, hard, [r65 configurationID_1039, post:PrecisionGreaterThanYield, true (refines r59)]), false), [... true], true).
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr158, hard, [r66 configurationID_1039, func:Add (refines r68)]), true), [... true], true).
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr159, hard, [r67 configurationID_1040, post:PrecisionGreaterThanYield, true (refines r61)]), false), [... true], true).
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr160, hard, [r68 configurationID_1040, func:Subtract (refines r62)]), true), [... true], true).
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr161, hard, [r69 configurationID_1041, post:PrecisionGreaterThanYield, true (refines r63)]), false), [... true], true).
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr162, hard, [r70 configurationID_1041, func:Multiply (refines r64)]), false), [... true], true).
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr163, hard, [r71 configurationID_1039, post:PrecisionGreaterThanYield, true]), false), [... true], true).
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr164, hard, [r72 configurationID_1039, func:Add (refines r59)]), true), [... true], true).
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr165, hard, [r73 configurationID_1039, post:PrecisionGreaterThanYield, true (refines r59)]), false), [... true], true).
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr166, hard, [r74 configurationID_1039, func:Subtract (refines r62)]), true), [... true], true).
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr167, hard, [r75 configurationID_1039, func:Multiply (refines r64)]), false), [... true], true).
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr168, hard, [r76 configurationID_1040, post:PrecisionGreaterThanYield, true]), false), [... true], true).
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr169, hard, [r77 configurationID_1040, func:Subtract (refines r62)]), true), [... true], true).
IncludesBasicEvaluationInformation((rqs_41,... satisifies ((DOD_42, (qr170, hard, [r78 configurationID_1040, func:Multiply (refines r64)]), false), [... true], true).
Resulting RQSIstance
rqs_42

internal present requirements:
(r7 func:DetermineSquaredHeight, true)
(r59 configurationID_1039, post:PrecisionGreaterThanYield, true, true)
(r60 configurationID_1039, post:PrecisionGreaterThanYield, true (refines r59), true)
(r61 configurationID_1040, func:Add (refines r60), true)
(r62 configurationID_1040, func:Subtract (refines r62), true)
(r63 configurationID_1040, func:Multiply (refines r64), true)
(r64 configurationID_1041, post:PrecisionGreaterThanYield, true)
(r65 configurationID_1041, post:PrecisionGreaterThanYield, true (refines r66), true)
(r66 configurationID_1041, func:Subtract (refines r62), true)
(r67 configurationID_1041, func:Multiply (refines r64), true)
(r68 configurationID_1041, func:Subtract (refines r62), true)
(r69 configurationID_1041, func:Multiply (refines r64), true)
(r70 configurationID_1041, func:Subtract (refines r62), true)

Enacted Requirements:
IstBeSatisfied(r7 func:DetermineSquaredHeight)
IstBeSatisfied(r65 configurationID_1039, post:PrecisionGreaterThanYield, true (refines r59)
IstBeSatisfied(r66 configurationID_1039, func:Add (refines r60))
IstBeSatisfied(r67 configurationID_1040, post:PrecisionGreaterThanYield, true (refines r61)
IstBeSatisfied(r68 configurationID_1040, func:Subtract (refines r62))
IstBeSatisfied(r69 configurationID_1041, post:PrecisionGreaterThanYield, true (refines r63)
IstBeSatisfied(r70 configurationID_1041, func:Multiply (refines r64))

Resulting Process Evaluation
ProcessEvaluationCollection POC_158
processEvaluation (=Strat_116, RefineSelectedTemplate, Succeeded, RQSM) Succeeded

05-Oct-2009 23:45:27 designProcess.components.DPC old logStatus
WARNING Status-report of DPC

No. of times activated: 21

Received DODM Process Evaluation:
ProcessEvaluationCollection POC_153
processEvaluation (=Strat_116, RefineSelectedTemplate, Incomplete, DODM) Incomplete, more activations are needed, or input from RQSM

Received RQSM Process Evaluation:
Trace of Multi-Level Reconfiguration Example

Process Evaluation Collection POC\textsubscript{158}
\texttt{processEvaluation(Strat\_116, \textbf{RefineSelectedTemplate}, Succeeded, RQSM) Succeeded}

\begin{itemize}
  \item \texttt{OUTPUT}:
  \begin{itemize}
    \item Strategy for component to be activated: Strat\_116, \textbf{RefineSelectedTemplate}
  \end{itemize}
\end{itemize}


\begin{itemize}
  \item No. of times activated: 11
  \item \texttt{INPUT}:
    \begin{itemize}
      \item Received overall Strategy: Strat\_116, \textbf{RefineSelectedTemplate}
      \item Received RQ\textsubscript{instance}:
    \end{itemize}
\end{itemize}

\texttt{OUTPUT}:
\begin{itemize}
  \item Resulting Overall Dod Assessments:
  \begin{itemize}
    \item \texttt{IncludesBasicEvaluationInformation} (\texttt{[rqs\_42 \ldots \{satisfies (\texttt{DOD44,} (\texttt{qr176,} hard, \{r7 func:DetermineSquaredHeight\}) \},true),\ldots,\{true\},true])}
    \item \texttt{IncludesBasicEvaluationInformation} (\texttt{[rqs\_42 \ldots \{satisfies (\texttt{DOD44,} (\texttt{qr177,} hard, \{r65 configurationID\_1039, post:PrecisionGreaterThanYield, true (refines r59)\}) \},false),\ldots,\{true\},true])}
    \item \texttt{IncludesBasicEvaluationInformation} (\texttt{[rqs\_42 \ldots \{satisfies (\texttt{DOD44,} (\texttt{qr178,} hard, \{r66 configurationID\_1039, func:Add (refines r60)}) \},true),\ldots,\{true\},true])}
    \item \texttt{IncludesBasicEvaluationInformation} (\texttt{[rqs\_42 \ldots \{satisfies (\texttt{DOD44,} (\texttt{qr179,} hard, \{r67 configurationID\_1040, post:PrecisionGreaterThanYield, true (refines r61)\}) \},false),\ldots,\{true\},true])}
    \item \texttt{IncludesBasicEvaluationInformation} (\texttt{[rqs\_42 \ldots \{satisfies (\texttt{DOD44,} (\texttt{qr180,} hard, \{r68 configurationID\_1040, func:Subtract (refines r62)}) \},true),\ldots,\{true\},true])}
    \item \texttt{IncludesBasicEvaluationInformation} (\texttt{[rqs\_42 \ldots \{satisfies (\texttt{DOD44,} (\texttt{qr181,} hard, \{r69 configurationID\_1041, post:PrecisionGreaterThanYield, true (refines r63)\}) \},false),\ldots,\{true\},true])}
    \item \texttt{IncludesBasicEvaluationInformation} (\texttt{[rqs\_42 \ldots \{satisfies (\texttt{DOD44,} (\texttt{qr182,} hard, \{r70 configurationID\_1041, func:Multiplies (refines r64)}) \},true),\ldots,\{true\},true])}
    \item \texttt{IncludesBasicEvaluationInformation} (\texttt{[rqs\_42 \ldots \{satisfies (\texttt{DOD44,} (\texttt{qr183,} hard, \{r83 configurationID\_1039, post:PrecisionGreaterThanYield, true) \},false),\ldots,\{true\},true])}
    \item \texttt{IncludesBasicEvaluationInformation} (\texttt{[rqs\_42 \ldots \{satisfies (\texttt{DOD44,} (\texttt{qr184,} hard, \{r84 configurationID\_1039, func:Add \}) \},false),\ldots,\{true\},true])}
    \item \texttt{IncludesBasicEvaluationInformation} (\texttt{[rqs\_42 \ldots \{satisfies (\texttt{DOD44,} (\texttt{qr185,} hard, \{r85 configurationID\_1040, post:PrecisionGreaterThanYield, true) \},false),\ldots,\{true\},true])}
    \item \texttt{IncludesBasicEvaluationInformation} (\texttt{[rqs\_42 \ldots \{satisfies (\texttt{DOD44,} (\texttt{qr186,} hard, \{r86 configurationID\_1040, func:Subtract \}) \},false),\ldots,\{true\},true])}
    \item \texttt{IncludesBasicEvaluationInformation} (\texttt{[rqs\_42 \ldots \{satisfies (\texttt{DOD44,} (\texttt{qr187,} hard, \{r87 configurationID\_1041, post:PrecisionGreaterThanYield, true) \},false),\ldots,\{true\},true])}
    \item \texttt{IncludesBasicEvaluationInformation} (\texttt{[rqs\_42 \ldots \{satisfies (\texttt{DOD44,} (\texttt{qr188,} hard, \{r88 configurationID\_1041, func:Multiplies \}) \},false),\ldots,\{true\},true])}
  \end{itemize}
\end{itemize}

Resulting Dod:
\texttt{DOD\_4414DetermineSquaredHeight s1Add (configurationID\_1039) filled with wsMathAdd s2Subtract (configurationID\_1040) filled with wsMathSubtract2 s3Multiply (configurationID\_1041) filled with wsMathMultiply}

\texttt{Resulting Process Evaluation:}
\texttt{ProcessEvaluationCollection POC\_163}
\texttt{processEvaluation(Strat\_116, \textbf{RefineSelectedTemplate}, Succeeded, DOEM) Succeeded}

Check pre- and post-conditions


\begin{itemize}
  \item No. of times activated: 22
  \item \texttt{INPUT}:
    \begin{itemize}
      \item Received DOEM Process Evaluation:
    \end{itemize}
\end{itemize}
J.6 Template-based Configuration (2nd) 229

ProcessEvaluationCollection POC_163
processEvaluation(Strat_116 , RefineSelectedTemplate , Succeeded , DODM) Succeeded

Received RQSM Process Evaluation
ProcessEvaluationCollection POC_158
processEvaluation(Strat_116 , RefineSelectedTemplate , Succeeded , RQSM) Succeeded

---OUTPUT---
Strategy for component to be activated: Strat_141 , TestPre_PostConditions

WARNING Status report of RQSM

No. of times activated: 11
---INPUT---
Received overall Strategy: Strat_141 , TestPre_PostConditions
Received DODAssessment:

[ ] [ IncludesBasicEvaluationInformation ({ rq5_42 ... | satisfies ((DOD_44 , (qr176 , hard , [ r7 func:DetermineSquaredHeight ))) , true ) , ... | true } , true ) ,
IncludesBasicEvaluationInformation ({ rq5_42 ... | satisfies ((DOD_44 , (qr177 , hard , [ r65 configurationID_1039 , post:PrecisionGreaterThanYield , true ( refines r59 ))) , false ) , ... | true } , true ) ,
IncludesBasicEvaluationInformation ({ rq5_42 ... | satisfies ((DOD_44 , (qr178 , hard , [ r66 configurationID_1039 , func:Add ( refines r60 ))) , true ) , ... | true } , true ) ,
IncludesBasicEvaluationInformation ({ rq5_42 ... | satisfies ((DOD_44 , (qr179 , hard , [ r67 configurationID_1040 , post:PrecisionGreaterThanYield , true ( refines r61 ))) , false ) , ... | true } , true ) ,
IncludesBasicEvaluationInformation ({ rq5_42 ... | satisfies ((DOD_44 , (qr180 , hard , [ r68 configurationID_1040 , func:Subtract ( refines r62 ))) , true ) , ... | true } , true ) ,
IncludesBasicEvaluationInformation ({ rq5_42 ... | satisfies ((DOD_44 , (qr181 , hard , [ r69 configurationID_1041 , post:PrecisionGreaterThanYield , true ( refines r63 ))) , false ) , ... | true } , true ) ,
IncludesBasicEvaluationInformation ({ rq5_42 ... | satisfies ((DOD_44 , (qr182 , hard , [ r70 configurationID_1041 , func:Multiply ( refines r64 ))) , true ) , ... | true } , true ) ,
IncludesBasicEvaluationInformation ({ rq5_42 ... | satisfies ((DOD_44 , (qr170 , hard , [ r83 configurationID_1039 , post:PrecisionGreaterThanYield , true ) ) ) , false ) , ... | true } , true ) ,
IncludesBasicEvaluationInformation ({ rq5_42 ... | satisfies ((DOD_44 , (qr171 , hard , [ r84 configurationID_1039 , func:Add () ) ) ) , false ) , ... | true } , true ) ,
IncludesBasicEvaluationInformation ({ rq5_42 ... | satisfies ((DOD_44 , (qr172 , hard , [ r85 configurationID_1040 , post:PrecisionGreaterThanYield , true ) ) ) , false ) , ... | true } , true ) ,
IncludesBasicEvaluationInformation ({ rq5_42 ... | satisfies ((DOD_44 , (qr173 , hard , [ r86 configurationID_1040 , func:Subtract () ) ) ) , false ) , ... | true } , true ) ,
IncludesBasicEvaluationInformation ({ rq5_42 ... | satisfies ((DOD_44 , (qr174 , hard , [ r87 configurationID_1041 , post:PrecisionGreaterThanYield , true ) ) ) , false ) , ... | true } , true ) ,
IncludesBasicEvaluationInformation ({ rq5_42 ... | satisfies ((DOD_44 , (qr175 , hard , [ r88 configurationID_1041 , func:Multiply () ) ) ) , false ) , ... | true } , true ) ,

---OUTPUT---
Resulting RQSInstance
rq5_44
internal present requirements:
(r7 func:DetermineSquaredHeight , true )
(r59 configurationID_1039 , post:PrecisionGreaterThanYield , true )
(r65 configurationID_1039 , post:PrecisionGreaterThanYield , true ( refines r59 , true ) )
(r60 configurationID_1039 , func:Add ( refines r60 , true ) )
(r61 configurationID_1040 , post:PrecisionGreaterThanYield , true ( refines r61 , true ) )
(r62 configurationID_1040 , func:Subtract ( refines r62 , true ) )
(r68 configurationID_1040 , func:Subtract ( refines r62 , true ) )
(r63 configurationID_1041 , post:PrecisionGreaterThanYield , true ( refines r63 , true ) )
(r69 configurationID_1041 , post:PrecisionGreaterThanYield , true ( refines r63 , true ) )
(r70 configurationID_1041 , func:Multiply ( refines r64 , true ) )

Enacted Requirements:
IsToBeSatisfied(r7 func:DetermineSquaredHeight)
IsToBeSatisfied(r65 configurationID_1039 , post:PrecisionGreaterThanYield , true ( refines r59 , true ) )
IsToBeSatisfied(r66 configurationID_1039 , func:Add ( refines r60 , true ) )
IsToBeSatisfied(r67 configurationID_1040 , post:PrecisionGreaterThanYield , true ( refines r61 , true ) )
Trace of Multi-Level Reconfiguration Example

```
IsToBeSatisfied(r68 configurationID,1040,func:Subtract(refines r62))
IsToBeSatisfied(r69 configurationID,1041,post:PrecisionGreaterThanYield,refines r63)
IsToBeSatisfied(r70 configurationID,1041,func:Multiply(refines r64))

Resulting Process Evaluation:
ProcessEvaluationCollection POC_168
processEvaluation(Strat_141, TestPre,PostConditions, Succeeded, RQSM) Succeeded

05-OCT-2009 23:45:27 designProcess components DPC,old logStatus
WARNING: Status—report of DPC

No. of times activated: 23

INPUT:
Received DDOM Process Evaluation:
ProcessEvaluationCollection POC_168
processEvaluation(Strat_116, RefSelectedTemplate, Succeeded, DDOM) Succeeded

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_168
processEvaluation(Strat_141, TestPre,PostConditions, Succeeded, RQSM) Succeeded

OUTPUT:
Strategy for component to be activated: Strat_141, TestPre,PostConditions

05-OCT-2009 23:45:27 designProcess components DDCM logStatus
WARNING: Status—report of DOCM

No. of times activated: 12

INPUT:
Received overall Strategy:
Strat_141, TestPre,PostConditions
Received RQInstances:
rr4,44

OUTPUT:
Resulting Overall Dod Assessments:
[]
Resulting RQInstances Assessments:
[]
```

---

```
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr203,hard,...func:DetermineSquaredHeight))true)
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr204,hard,...configurationID,1039,post:PrecisionGreaterThanYield,refines r59))true)
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr205,hard,...configurationID,1039,func:Add(refines r60))true)
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr206,hard,...configurationID,1040,post:PrecisionGreaterThanYield,refines r61))true)
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr207,hard,...configurationID,1040,func:Subtract(refines r62))true)
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr208,hard,...configurationID,1040,post:PrecisionGreaterThanYield,refines r63))true)
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr209,hard,...configurationID,1041,func:Multiply(refines r64))true)
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr197,hard,...configurationID,1039,post:PrecisionGreaterThanYield,refines r59))true)
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr198,hard,...configurationID,1039,func:Add))false)
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr199,hard,...configurationID,1040,post:PrecisionGreaterThanYield,refines r59))false)
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr200,hard,...configurationID,1040,func:Subtract))false)
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr201,hard,...configurationID,1041,post:PrecisionGreaterThanYield,refines r64))false)
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr202,hard,...configurationID,1041,func:Multiply))false)
includesBasicEvaluationInformation(rr4,44,...satisfies((DOD,46,qr203,hard,...configurationID,1041,func:Add))false)

Resulting DOD:
DOD_46
```
J.7 Integration

Resulting Process Evaluation:
ProcessEvaluationCollection POC_173
processEvaluation (Strat_141, TestPre, PostConditions, Succeeded, DODM) Succeeded

WARNING: Status REPORT of DPC

No. of times activated: 24

---INPUT---
Received DODM Process Evaluation:
ProcessEvaluationCollection POC_173
processEvaluation (Strat_141, TestPre, PostConditions, Succeeded, DODM) Succeeded

Received RQSM Process Evaluation:
ProcessEvaluationCollection POC_168
processEvaluation (Strat_141, TestPre, PostConditions, Succeeded, RQSM) Succeeded

---OUTPUT---
Strategy for component to be activated: null

INFO: Successfully found a configuration

INFO: Resulting Configuration = DOD

 Failed Component = wsMathSubtract1

INFO: Final configuration, after integration:

Failed Component = wsMathArea

INFO: Final configuration, after integration:

Failed Component = wsMathArea

J.7 Integration

INFO: Starting Integration
Design Finished

Initial Configuration: DOD

Failed Component = wsMathSUB}
Trace of Multi-Level Reconfiguration Example
Samenvatting

De hypothese van dit proefschrift is dat automatische herconfiguratie van complexe services mogelijk is door gebruik van lokale kennis. Lokale kennis maakt het mogelijk om verandering lokaal te houden zonder de functionele werking te veranderen zodat de aanpassingen in de structuur minimaal blijven. Om een systeem te maken dat zelfstandig geautomatiseerd herconfiguratie kan plegen op basis van lokale kennis, is propagatie nodig, waarmee het mogelijk wordt om op basis van lokale kennis te redeneren over de gevolgen die niet lokaal zijn, bijvoorbeeld voor Quality of Service eigenschappen. De volgende onderzoeksvragen worden in hoofdstuk 1 gesteld:

1. Hoe kan lokale kennis gerepresenteerd worden en in een configuratie worden opgenomen?

2. Welke processen zijn nodig voor geautomatiseerde herconfiguratie die gebruik maakt van lokale kennis?

Hoofdstuk 2 presenteert gerelateerd onderzoek in geautomatiseerde herconfiguratie, en concludeert dat de verschillende aanpakken elk sterke en zwakke kanten hebben. Component-gebaseerde vervanging is een aanpak die dicht bij de originele configuratie blijft, maar hangt zwaar op de beschikbaarheid van een service die precies overeenkomt met de te vervangen service. Herconfiguratie op een enkelvoudig niveau maakt meer geavanceerde aanpassingen mogelijk, maar hiervoor moet elke configuratie opnieuw geannoteerd worden met specifieke kennis. Herconfiguratie op meervoudige niveaus maakt het mogelijk om te variëren in de omvang van de structuur van de configuratie die wordt aangepast, maar heeft wel meer geannoteerde compositieele structuren nodig. Configuratie als herconfiguratie is vaak gelimiteerd doordat de vereisten niet expliciet zijn gemaakt voor de configuratie die moet worden aangepast. Een hybride aanpak zou ideaal zijn, daar deze zowel kan variëren in de focus van welk deel van de structuur veranderd wordt, als in de manieren waarop dit deel wordt veranderd. De aanpak die in dit proefschrift beschreven staat, is een hybride aanpak.
In hoofdstuk 3 wordt de representatie van lokale kennis besproken, en hoe deze kan worden opgenomen in een configuratie. De volgende structuren worden hiervoor gedefinieerd: template in Sectie 3.2, slot in Sectie 3.2 en template-gebaseerde configuratie in Sectie 3.3. Een template definieert en anoteert de vereisten die geassocieerd zijn met een complexe web service in meer detail dan een enkele service, maar in minder details dan voor een gehele complexe service. Een template representeert een enkelvoudig niveau van een configuratie. De slots van een template specificeren de vereisten voor elke component (een web service of weer een andere template) die gebruikt wordt binnen een template. Door het invullen van slots met specifieke web services en templates wordt een template-gebaseerde configuratie gedefinieerd. Propagatie, beschreven in Sectie 3.4, maakt het mogelijk om te redeneren over vereisten die over meerdere niveaus van een configuratie afhankelijkheden hebben, bijvoorbeeld Quality of Service.

Het proces dat nodig is voor geautomatiseerde herconfiguratie met gebruik van lokale kennis wordt besproken in hoofdstuk 4. Een generiek herconfiguratie proces wordt beschreven, met de volgende subprocessen: redeneren over de focus in Sectie 4.2.1, redeneren over de vereisten in Sectie 4.2.2, template-gebaseerde configuratie in Sectie 4.2.3 en integratie in Sectie 4.2.4. Er is een prototype van het herconfiguratie proces ontwikkeld, welk staat beschreven in Sectie 4.3, met verdere details in de appendices D tot H.

Twee toepassingen van de gepresenteerde aanpak zijn opgenomen in dit proefschrift: classificatie-services in hoofdstuk 5 en mathematische services in hoofdstuk 6. Herconfiguratie op een enkelvoudig niveau is geïllustreerd in hoofdstuk 5, waarin wordt aangegeven dat deze aanpak minstens even sterk is als bestaande aanpakken. Herconfiguratie op meervoudige niveaus wordt beschreven in hoofdstuk 6, waar de toegevoegde waarde wordt getoond van het geleidelijk uitbreiden van het bereik van de adaptaties, en ook van het lokale redeneren over eigenschappen gerelateerd aan de Quality of Service. Beide scenario’s worden door het geïmplementeerde prototype ondersteund. Gezamenlijk tonen deze toepassingen aan dat (1) de gedefinieerde structuren voor het representeren en opnemen van lokale kennis in een configuratie generiek genoeg zijn om gebruikt te worden op bestaande services uit twee verschillende domeinen, (2) het gedefinieerde herconfiguratie-proces generiek genoeg is om herconfiguratie-problemen uit twee verschillende domeinen aan te kunnen, en dat het zowel herconfiguratie-problemen aankan op een enkelvoudig niveau, zoals beschreven in Sectie 5.3, als op meervoudige niveaus, zoals beschreven in Sectie 6.3.
SIKS Dissertation Series

1998

1998-1 Johan van den Akker (CWI)
DEGAS - An Active, Temporal Database of Autonomous Objects

1998-2 Floris Wiesman (UM)
Information Retrieval by Graphically Browsing Meta-Information

1998-3 Ans Steuten (TUD)
A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective

1998-4 Dennis Breuker (UM)
Memory versus Search in Games

1998-5 E.W. Oskamp (RUL)
Computerondersteuning bij Straftoemeting

1999

1999-1 Mark Sloof (VU)
Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products

1999-2 Rob Potharst (EUR)
Classification using decision trees and neural nets

1999-3 Don Beal (UM)
The Nature of Minimax Search

1999-4 Jacques Penders (UM)
The practical Art of Moving Physical Objects

1999-5 Aldo de Moor (KUB)
Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems

1999-6 Niek J.E. Wijngaards (VU)
Re-design of compositional systems

1999-7 David Spelt (UT)
Verification support for object database design

1999-8 Jacques H.J. Lenting (UM)
Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation

2000

2000-1 Frank Niessink (VU)
Perspectives on Improving Software Maintenance

2000-2 Koen Holtman (TUE)
Prototyping of CMS Storage Management

2000-3 Carolien M.T. Metselaar (UVA)
Sociaal-organisatorische gevolgen van kennis-technologie; een procesbenadering en actielperspectief

2000-4 Geert de Haan (VU)
ETAG, A Formal Model of Competence Knowledge for User Interface Design

2000-5 Ruud van der Pol (UM)
Knowledge-based Query Formulation in Information Retrieval

2000-6 Rogier van Eijk (UU)
Programming Languages for Agent Communication

2000-7 Niels Peck (UU)
Decision-theoretic Planning of Clinical Patient Management

2000-8 Veerle Coup (EUR)
Sensitivity Analysis of Decision-Theoretic Networks

2000-9 Florian Waas (CWI)
Principles of Probabilistic Query Optimization

2000-10 Niels Nes (CWI)
Image Database Management System Design Considerations, Algorithms and Architecture

2000-11 Jonas Karlsson (CWI)
Scalable Distributed Data Structures for Database Management
2001

2001-1 Silja Renooij (UU)
Qualitative Approaches to Quantifying Probabilistic Networks

2001-2 Koen Hindriks (UU)
Agent Programming Languages: Programming with Mental Models

2001-3 Maarten van Someren (UvA)
Learning as problem solving

2001-4 Evgueni Smirnov (UM)
Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets

2001-5 Jacco van Ossenbruggen (VU)
Processing Structured Hypermedia: A Matter of Style

2001-6 Martijn van Welie (VU)
Task-based User Interface Design

2001-7 Bastiaan Schönhage (VU)
Diva: Architectural Perspectives on Information Visualization

2001-8 Pascal van Eck (VU)
A Compositional Semantic Structure for Multi-Agent Systems Dynamics

2001-9 Pieter Jan ‘t Hoen (RUL)
Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes

2001-10 Maarten Sierhuis (UvA)
Modelling and Simulating Work Practice: BRAHMS: a multiagent modeling and simulation language for work practice analysis and design

2001-11 Tom M. van Engers (VUA)
Knowledge Management: The Role of Mental Models in Business Systems Design

2002

2002-01 Nico Lassing (VU)
Architecture-Level Modifiability Analysis

2002-02 Roelof van Zwol (UT)
Modelling and searching web-based document collections

2002-03 Henk Ernst Blok (UT)
Database Optimization Aspects for Information Retrieval

2002-04 Juan Roberto Castelo Valdueza (UU)
The Discrete Acyclic Digraph Markov Model in Data Mining

2002-05 Radu Serban (VU)
The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents

2002-06 Laurens Mommers (UL)
Applied legal epistemology; Building a knowledge-based ontology of the legal domain

2002-07 Peter Bonez (CWI)
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications

2002-08 Jaap Gordijn (VU)
Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas

2002-09 Willem-Jan van den Heuvel(KUB)
Integrating Modern Business Applications with Objectified Legacy Systems

2002-10 Brian Sheppard (UM)
Towards Perfect Play of Scrabble

2002-11 Wouter C.A. Wijngaards (VU)
Agent Based Modelling of Dynamics: Biological and Organisational Applications

2002-12 Albrecht Schmidt (UvA)
Processing XML in Database Systems

2002-13 Hongjing Wu (TUE)
A Reference Architecture for Adaptive Hypermedia Applications

2002-14 Wieke de Vries (UU)
Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems

2002-15 Rik Eshuis (UT)
Semantics and Verification of UML Activity Diagrams for Workflow Modelling

2002-16 Pieter van Langen (VU)
The Anatomy of Design: Foundations, Models and Applications

2002-17 Stefan Manegold (UVA)
Understanding, Modeling, and Improving Main-Memory Database Performance

2003

2003-01 Heiner Stuckenschmidt (VU)
Ontology-Based Information Sharing in Weakly Structured Environments

2003-02 Jan Broersen (VU)
Modal Action Logics for Reasoning About Reactive Systems

2003-03 Martijn Schuemie (TUD)
Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy
2003-04 Milan Petkovic (UT)
Content-Based Video Retrieval Supported by Database Technology

2003-05 Jos Lehmann (UVA)
Causation in Artificial Intelligence and Law - A modelling approach

2003-06 Boris van Schooten (UT)
Development and specification of virtual environments

2003-07 Machiel Jansen (UvA)
Formal Explorations of Knowledge Intensive Tasks

2003-08 Yongping Ran (UM)
Repair Based Scheduling

2003-09 Rens Kortmann (UM)
The resolution of visually guided behaviour

2003-10 Andreas Linche (UvT)
Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture

2003-11 Simon Keizer (UT)
Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks

2003-12 Roeland Ordelman (UT)
Dutch speech recognition in multimedia information retrieval

2003-13 Jeroen Donkers (UM)
Nosce Hostem - Searching with Opponent Models

2003-14 Stijn Hoppenbrouwers (KUN)
Freezing Language: Conceptualisation Processes across ICT-Supported Organisations

2003-15 Mathijs de Weerdt (TUD)
Plan Merging in Multi-Agent Systems

2003-16 Menzo Windhouwer (CWI)
Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses

2003-17 David Jansen (UT)
Extensions of Statecharts with Probability, Time, and Stochastic Timing

2003-18 Levente Kocsis (UM)
Learning Search Decisions

2004

2004-01 Virginia Dignum (UU)
A Model for Organizational Interaction: Based on Agents, Founded in Logic

2004-02 Lai Xu (UvT)
Monitoring Multi-party Contracts for E-business

2004-03 Perry Groot (VU)
A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving

2004-04 Chris van Aart (UVA)
Organizational Principles for Multi-Agent Architectures

2004-05 Viara Popova (EUR)
Knowledge discovery and monotonicity

2004-06 Bart-Jan Hommes (TUD)
The Evaluation of Business Process Modeling Techniques

2004-07 Elise Boltjes (UM)
Voorbeeld onderwijs; voorbeeldgestuurd onderwijs, een stap naar abstract denken, vooral voor meisjes

2004-08 Joop Verbeek (UM)
Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politie gegevensuitwisseling en digitale expertise

2004-09 Martin Caminada (VU)
For the Sake of the Argument; explorations into argument-based reasoning

2004-10 Suzanne Kabel (UVA)
Knowledge-rich indexing of learning-objects

2004-11 Michel Klein (VU)
Change Management for Distributed Ontologies

2004-12 The Duy Bui (UT)
Creating emotions and facial expressions for embodied agents

2004-13 Wojciech Jamroga (UT)
Using Multiple Models of Reality: On Agents who Know how to Play

2004-14 Paul Harrenstein (UU)
Logic in Conflict. Logical Explorations in Strategic Equilibrium

2004-15 Arno Knobbe (UU)
Multi-Relational Data Mining

2004-16 Federico Divina (VU)
Hybrid Genetic Relational Search for Inductive Learning

2004-17 Mark Winands (UM)
Informed Search in Complex Games

2004-18 Vania Bessa Machado (UvA)
Supporting the Construction of Qualitative Knowledge Models

2004-19 Thijs Westerveld (UT)
Using generative probabilistic models for multimedia retrieval
2004-20 Madelon Evers (Nyenrode)
   Learning from Design: facilitating multidisciplinary design teams

2005

2005-01 Floor Verdenius (UVA)
   Methodological Aspects of Designing Induction-Based Applications

2005-02 Erik van der Werf (UM)
   AI techniques for the game of Go

2005-03 Franc Grootjen (RUN)
   A Pragmatic Approach to the Conceptualisation of Language

2005-04 Nirvana Meratnia (UT)
   Towards Database Support for Moving Object data

2005-05 Gabriel Infante-Lopez (UVA)
   Two-Level Probabilistic Grammars for Natural Language Parsing

2005-06 Pieter Spronck (UM)
   Adaptive Game AI

2005-07 Flavius Frasincar (TUE)
   Hypermedia Presentation Generation for Semantic Web Information Systems

2005-08 Richard Vdovjak (TUE)
   A Model-driven Approach for Building Distributed Ontology-based Web Applications

2005-09 Jeen Broekstra (VU)
   Storage, Querying and Inferencing for Semantic Web Languages

2005-10 Anders Bouwer (UVA)
   Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments

2005-11 Elth Ogston (VU)
   Agent Based Matchmaking and Clustering - A Decentralized Approach to Search

2005-12 Csaba Boer (EUR)
   Distributed Simulation in Industry

2005-13 Fred Hamburg (UL)
   Een Computermmodel voor het Ondersteunen van Euthanasiebeslissingen

2005-14 Borys Omeljenko (VU)
   Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics

2005-15 Tibor Bosse (VU)
   Analysis of the Dynamics of Cognitive Processes

2005-16 Joris Graaumans (UU)
   Usability of XML Query Languages

2005-17 Boris Shishkov (TUD)
   Software Specification Based on Re-usable Business Components

2005-18 Danielle Sent (UU)
   Test-selection strategies for probabilistic networks

2005-19 Michel van Dartel (UM)
   Situated Representation

2005-20 Cristina Coteanu (UL)
   Cyber Consumer Law, State of the Art and Perspectives

2005-21 Wijnand Derks (UT)
   Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

2006

2006-01 Samuil Angelov (TUE)
   Foundations of B2B Electronic Contracting

2006-02 Cristina Chisalita (VU)
   Contextual issues in the design and use of information technology in organizations

2006-03 Noor Christoph (UVA)
   The role of metacognitive skills in learning to solve problems

2006-04 Marta Sabou (VU)
   Building Web Service Ontologies

2006-05 Cees Pierik (UU)
   Validation Techniques for Object-Oriented Proof Outlines

2006-06 Ziv Baida (VU)
   Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modelling

2006-07 Marko Smiljanic (UT)
   XML schema matching – balancing efficiency and effectiveness by means of clustering

2006-08 Eelco Herder (UT)
   Forward, Back and Home Again - Analyzing User Behavior on the Web

2006-09 Mohamed Wahdan (UM)
   Automatic Formulation of the Auditor’s Opinion

2006-10 Ronny Siebes (VU)
   Semantic Routing in Peer-to-Peer Systems

2006-11 Joeri van Ruth (UT)
   Flattening Queries over Nested Data Types

2006-12 Bert Bongers (VU)
   Interactivation - Towards an e-cology of people, our technological environment, and the arts
2006-13 Henk-Jan Lebbink (UU)
Dialogue and Decision Games for Information Exchanging Agents

2006-14 Johan Hoorn (VU)
Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change

2006-15 Rainer Malik (UU)
CONAN: Text Mining in the Biomedical Domain

2006-16 Carsten Riggelsen (UU)
Approximation Methods for Efficient Learning of Bayesian Networks

2006-17 Stacey Nagata (UU)
User Assistance for Multitasking with Interruptions on a Mobile Device

2006-18 Valentin Zhizhikun (UVA)
Graph transformation for Natural Language Processing

2006-19 Birna van Riemsdijk (UU)
Cognitive Agent Programming: A Semantic Approach

2006-20 Marina Velikova (UvT)
Monotone models for prediction in data mining

2006-21 Bas van Gils (RUN)
Aptness on the Web

2006-22 Paul de Vrieze (RUN)
Fundamentals of Adaptive Personalisation

2006-23 Ion Juvina (UU)
Development of Cognitive Model for Navigating on the Web

2006-24 Laura Hollink (VU)
Semantic Annotation for Retrieval of Visual Resources

2006-25 Madalina Drugan (UU)
Conditional log-likelihood MDL and Evolutionary MCMC

2006-26 Vojkan Mihajlovic (UT)
Score Region Algebra: A Flexible Framework for Structured Information Retrieval

2006-27 Stefano Bocconi (CWI)
Voz Populi: generating video documentaries from semantically annotated media repositories

2006-28 Borkur Sigurbjornsson (UVA)
Focused Information Access using XML Element Retrieval

2007

2007-01 Kees Leune (UvT)
Access Control and Service-Oriented Architectures

2007-02 Wouter Teepe (RUG)
Reconciling Information Exchange and Confidentiality: A Formal Approach

2007-03 Peter Mika (VU)
Social Networks and the Semantic Web

2007-04 Jurriaan van Diggelen (UU)
Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach

2007-05 Bart Schermer (UL)
Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance

2007-06 Gilad Mishne (UVA)
Applied Text Analytics for Blogs

2007-07 Natasa Jovanovic’ (UT)
To Whom It May Concern - Addressee Identification in Face-to-Face Meetings

2007-08 Mark Hoogendoorn (VU)
Modeling of Change in Multi-Agent Organizations

2007-09 David Mobach (VU)
Agent-Based Mediated Service Negotiation

2007-10 Huib Aldewereld (UU)
Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols

2007-11 Natalia Stash (TUE)
Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System

2007-12 Marcel van Gerven (RUN)
Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty

2007-13 Rutger Rienks (UT)
Meetings in Smart Environments; Implications of Progressing Technology

2007-14 Nick Bergboer (UM)
Context-Based Image Analysis

2007-15 Joyca Lacroix (UM)
NIM: a Situated Computational Memory Model

2007-16 Davide Grossi (UU)
Designing Invisible Handcuffs: Formal investigations in Institutions and Organizations for Multi-agent Systems
<table>
<thead>
<tr>
<th>Year</th>
<th>Candidate</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-17</td>
<td>Theodore Charitos (UU)</td>
<td>Reasoning with Dynamic Networks in Practice</td>
</tr>
<tr>
<td>2007-18</td>
<td>Bart Orriens (UvT)</td>
<td>On the development an management of adaptive business collaborations</td>
</tr>
<tr>
<td>2007-19</td>
<td>David Levy (UM)</td>
<td>Intimate relationships with artificial partners</td>
</tr>
<tr>
<td>2007-20</td>
<td>Slinger Jansen (UU)</td>
<td>Customer Configuration Updating in a Software Supply Network</td>
</tr>
<tr>
<td>2007-21</td>
<td>Karianne Vermaas (UU)</td>
<td>Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005</td>
</tr>
<tr>
<td>2007-22</td>
<td>Zlatko Zlatev (UT)</td>
<td>Goal-oriented design of value and process models from patterns</td>
</tr>
<tr>
<td>2007-23</td>
<td>Peter Barna (TUE)</td>
<td>Specification of Application Logic in Web Information Systems</td>
</tr>
<tr>
<td>2007-24</td>
<td>Georgina Ramrez Camps (CWI)</td>
<td>Structural Features in XML Retrieval</td>
</tr>
<tr>
<td>2008-01</td>
<td>Katalin Boer-Sorbn (EUR)</td>
<td>Agent-Based Simulation of Financial Markets: A modular, continuous-time approach</td>
</tr>
<tr>
<td>2008-03</td>
<td>Vera Hollink (UVA)</td>
<td>Optimizing hierarchical menus: a usage-based approach</td>
</tr>
<tr>
<td>2008-04</td>
<td>Ander de Keijzer (UT)</td>
<td>Management of Uncertain Data - towards unattended integration</td>
</tr>
<tr>
<td>2008-05</td>
<td>Bela Mutschler (UT)</td>
<td>Modeling and simulating causal dependencies on process-aware information systems from a cost perspective</td>
</tr>
<tr>
<td>2008-06</td>
<td>Arjen Hommersom (RUN)</td>
<td>On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective</td>
</tr>
<tr>
<td>2008-07</td>
<td>Peter van Rosmalen (OU)</td>
<td>Supporting the tutor in the design and support of adaptive e-learning</td>
</tr>
<tr>
<td>2008-08</td>
<td>Janneke Bolt (UU)</td>
<td>Bayesian Networks: Aspects of Approximate Inference</td>
</tr>
<tr>
<td>2008-09</td>
<td>Christof van Nimwegen (UU)</td>
<td>The paradox of the guided user: assistance can be counter-effective</td>
</tr>
<tr>
<td>2008-10</td>
<td>Wauter Bosma (UT)</td>
<td>Discourse oriented summarization</td>
</tr>
<tr>
<td>2008-11</td>
<td>Vera Kartseva (VU)</td>
<td>Designing Controls for Network Organizations: A Value-Based Approach</td>
</tr>
<tr>
<td>2008-12</td>
<td>Jozsef Farkas (RUN)</td>
<td>A Semiotically Oriented Cognitive Model of Knowledge Representation</td>
</tr>
<tr>
<td>2008-13</td>
<td>Caterina Carraciolo (UVA)</td>
<td>Topic Driven Access to Scientific Handbooks</td>
</tr>
<tr>
<td>2008-14</td>
<td>Arthur van Bunningen (UT)</td>
<td>Context-Aware Querying: Better Answers with Less Effort</td>
</tr>
<tr>
<td>2008-16</td>
<td>Henriette van Vugt (VU)</td>
<td>Embodied agents from a user’s perspective</td>
</tr>
<tr>
<td>2008-17</td>
<td>Martin Op ’t Land (TUD)</td>
<td>Applying Architecture and Ontology to the Splitting and Allying of Enterprises</td>
</tr>
<tr>
<td>2008-18</td>
<td>Guido de Croon (UM)</td>
<td>Adaptive Active Vision</td>
</tr>
<tr>
<td>2008-19</td>
<td>Henning Rode (UT)</td>
<td>From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search</td>
</tr>
<tr>
<td>2008-20</td>
<td>Rex Arendsen (UVA)</td>
<td>Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven</td>
</tr>
<tr>
<td>2008-21</td>
<td>Kristian Balog (UVA)</td>
<td>People Search in the Enterprise</td>
</tr>
<tr>
<td>2008-22</td>
<td>Henk Koning (UU)</td>
<td>Communication of IT-Architecture</td>
</tr>
<tr>
<td>2008-23</td>
<td>Stefan Visscher (UU)</td>
<td>Bayesian network models for the management of ventilator-associated pneumonia</td>
</tr>
<tr>
<td>Year</td>
<td>Title</td>
<td>Author(s)</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>2008</td>
<td>Using background knowledge in ontology matching</td>
<td>Zharko Aleksovski (VU)</td>
</tr>
<tr>
<td>2008</td>
<td>Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency</td>
<td>Geert Jonker (UU)</td>
</tr>
<tr>
<td>2008</td>
<td>Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled</td>
<td>Marijn Huijbregts (UT)</td>
</tr>
<tr>
<td>2008</td>
<td>Design and Implementation Strategies for IMS Learning Design</td>
<td>Hubert Vogten (OU)</td>
</tr>
<tr>
<td>2008</td>
<td>On the Use of Independence Relations in Bayesian Networks</td>
<td>Billko Flesch (RUN)</td>
</tr>
<tr>
<td>2008</td>
<td>Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans</td>
<td>Dennis Reidsma (UT)</td>
</tr>
<tr>
<td>2008</td>
<td>Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content</td>
<td>Wouter van Atteveldt (VU)</td>
</tr>
<tr>
<td>2008</td>
<td>Pro-Active Medical Information Retrieval</td>
<td>Loes Braun (UM)</td>
</tr>
<tr>
<td>2008</td>
<td>Toward Affective Dialogue Management using Partially Observable Markov Decision Processes</td>
<td>Trung H. Bui (UT)</td>
</tr>
<tr>
<td>2008</td>
<td>Scientific Workflow Design; theoretical and practical issues</td>
<td>Frank Terpstra (UVA)</td>
</tr>
<tr>
<td>2008</td>
<td>Studies in Frequent Tree Mining</td>
<td>Jeroen de Knijf (UU)</td>
</tr>
<tr>
<td>2008</td>
<td>Dendritic morphologies: function shapes structure</td>
<td>Ben Torben Nielsen (UvT)</td>
</tr>
<tr>
<td>2009</td>
<td>Symmetric Causal Independence Models</td>
<td>Rasa Jurgelenaite (RUN)</td>
</tr>
<tr>
<td>2009</td>
<td>Evaluating Ontology-Alignment Techniques</td>
<td>Willem Robert van Hage (VU)</td>
</tr>
<tr>
<td>2009</td>
<td>Toward Affective Dialogue Management using Partially Observable Markov Decision Processes</td>
<td>Hans Stol (UvT)</td>
</tr>
<tr>
<td>2009</td>
<td>Scientific Workflow Design; theoretical and practical issues</td>
<td>Frank Terpstra (UVA)</td>
</tr>
<tr>
<td>2009</td>
<td>Studies in Frequent Tree Mining</td>
<td>Jeroen de Knijf (UU)</td>
</tr>
<tr>
<td>2009</td>
<td>Dendritic morphologies: function shapes structure</td>
<td>Ben Torben Nielsen (UvT)</td>
</tr>
<tr>
<td>2009</td>
<td>Operating Guidelines for Services</td>
<td>Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin)</td>
</tr>
<tr>
<td>2009</td>
<td>From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)</td>
<td>Maksym Korotkiy (VU)</td>
</tr>
<tr>
<td>2009</td>
<td>Ontology Representation - Design Patterns and Ontologies that Make Sense</td>
<td>Rinke Hoekstra (UVA)</td>
</tr>
<tr>
<td>2009</td>
<td>New Architectures in Computer Chess</td>
<td>Laurents van der Maaten (UvT)</td>
</tr>
<tr>
<td>2009</td>
<td>Feature Extraction from Visual Data</td>
<td>Fabian Groffen (CWI)</td>
</tr>
<tr>
<td>2009</td>
<td>Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets</td>
<td>Valentin Robu (CWI)</td>
</tr>
<tr>
<td>2009</td>
<td>Adjustable Autonomy: Controlling Influences on Decision Making</td>
<td>Bob van der Vecht (UU)</td>
</tr>
<tr>
<td>2009</td>
<td>Ranking and Reliable Classification</td>
<td>Stijn Vanderlooy (UM)</td>
</tr>
<tr>
<td>Year</td>
<td>Author(s)</td>
<td>Title</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>2009-22</td>
<td>Pavel Serdyukov (UT)</td>
<td>Search For Expertise: Going beyond direct evidence</td>
</tr>
<tr>
<td>2009-23</td>
<td>Peter Hofgesang (VU)</td>
<td>Modelling Web Usage in a Changing Environment</td>
</tr>
<tr>
<td>2009-24</td>
<td>Annerieke Heuvelink (VUA)</td>
<td>Cognitive Models for Training Simulations</td>
</tr>
<tr>
<td>2009-25</td>
<td>Alex van Ballegooij (CWI)</td>
<td>RAM: Array Database Management through Relational Mapping</td>
</tr>
<tr>
<td>2009-26</td>
<td>Fernando Koch (UU)</td>
<td>An Agent-Based Model for the Development of Intelligent Mobile Services</td>
</tr>
<tr>
<td>2009-27</td>
<td>Christian Glahn (OU)</td>
<td>Contextual Support of Social Engagement and Reflection on the Web</td>
</tr>
<tr>
<td>2009-28</td>
<td>Sander Evers (UT)</td>
<td>Sensor Data Management with Probabilistic Models</td>
</tr>
<tr>
<td>2009-29</td>
<td>Stanislav Pokraev (UT)</td>
<td>Model-Driven Semantic Integration of Service-Oriented Applications</td>
</tr>
<tr>
<td>2009-30</td>
<td>Marcin Żukowski (CWI)</td>
<td>Balancing vectorized query execution with bandwidth-optimized storage</td>
</tr>
<tr>
<td>2009-31</td>
<td>Sofiya Katrenko (UVA)</td>
<td>A Closer Look at Learning Relations from Text</td>
</tr>
<tr>
<td>2009-32</td>
<td>Rik Farenhorst (VU) and Remco de Boer (VU)</td>
<td>Architectural Knowledge Management: Supporting Architects and Auditors</td>
</tr>
<tr>
<td>2009-33</td>
<td>Khiet Truong (UT)</td>
<td>How Does Real Affect Affect Affect Recognition in Speech?</td>
</tr>
<tr>
<td>2009-34</td>
<td>Inge van de Weerd (UU)</td>
<td>Advancing in Software Product Management: An Incremental Method Engineering Approach</td>
</tr>
<tr>
<td>2009-35</td>
<td>Wouter Koolewijn (UL)</td>
<td>Privacy en Politiegegevens: Over geautomatiseerde normatieve informatie-uitwisseling</td>
</tr>
<tr>
<td>2009-36</td>
<td>Marco Kalz (OUN)</td>
<td>Placement Support for Learners in Learning Networks</td>
</tr>
<tr>
<td>2009-37</td>
<td>Hendrik Drachsler (OUN)</td>
<td>Navigation Support for Learners in Informal Learning Networks</td>
</tr>
<tr>
<td>2009-38</td>
<td>Riina Vuorikari (OU)</td>
<td>Tags and self-organisation: a metadata ecology for learning resources in a multilingual context</td>
</tr>
<tr>
<td>2009-40</td>
<td>Stephan Raaijmakers (UvT)</td>
<td>Multinomial Language Learning: Investigations into the Geometry of Language</td>
</tr>
<tr>
<td>2009-41</td>
<td>Igor Berezhnyy (UvT)</td>
<td>Digital Analysis of Paintings</td>
</tr>
<tr>
<td>2009-42</td>
<td>Toine Bogers (UvT)</td>
<td>Recommender Systems for Social Bookmarking</td>
</tr>
<tr>
<td>2009-43</td>
<td>Virginia Nunes Leal Franqueira (UT)</td>
<td>Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients</td>
</tr>
<tr>
<td>2009-44</td>
<td>Roberto Santana Tapia (UT)</td>
<td>Assessing Business-IT Alignment in Networked Organizations</td>
</tr>
<tr>
<td>2009-45</td>
<td>Jilles Vreeken (UU)</td>
<td>Making Pattern Mining Useful</td>
</tr>
<tr>
<td>2009-46</td>
<td>Loredana Afanasiev (UvA)</td>
<td>Querying XML: Benchmarks and Recursion</td>
</tr>
</tbody>
</table>

**2010**

<table>
<thead>
<tr>
<th>Year</th>
<th>Author(s)</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010-01</td>
<td>Matthijs van Leeuwen (UU)</td>
<td>Patterns that Matter</td>
</tr>
<tr>
<td>2010-02</td>
<td>Ingo Wassink (UT)</td>
<td>Work flows in Life Science</td>
</tr>
<tr>
<td>2010-04</td>
<td>Olga Kulyk (UT)</td>
<td>Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments</td>
</tr>
<tr>
<td>2010-05</td>
<td>Claudia Hauff (UT)</td>
<td>Predicting the Effectiveness of Queries and Retrieval Systems</td>
</tr>
<tr>
<td>2010-06</td>
<td>Sander Bakkes (UvT)</td>
<td>Rapid Adaptation of Video Game AI</td>
</tr>
<tr>
<td>2010-07</td>
<td>Wim Fikkert (UT)</td>
<td>A Gesture interaction at a Distance</td>
</tr>
<tr>
<td>2010-08</td>
<td>Krzysztof Siewicz (UL)</td>
<td>Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments</td>
</tr>
</tbody>
</table>
2010-09 Hugo Kielman (UL)
A Politieele gegevensverwerking en Privacy, Naar een effectieve waarborging

2010-10 Rebecca Ong (UL)
Mobile Communication and Protection of Children

2010-11 Adriaan Ter Mors (TUD)
The world according to MARP: Multi-Agent Route Planning

2010-12 Susan van den Braak (UU)
Sensemaking software for crime analysis

2010-13 Gianluigi Folino (RUN)
High Performance Data Mining using Bio-inspired techniques

2010-14 Sander van Splunter (VU)
Automated Web Service Reconfiguration

2010-15 Lianne Bodenstaff (UT)
Managing Dependency Relations in Inter-Organizational Models

2010-16 Sicco Verwer (TUD)
Efficient Identification of Timed Automata, theory and practice

2010-17 Spyros Kotoulas (VU)
Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications