UNIVERSITÄT
KOBLENZ · LANDAU
Institut für Softwaretechnik

FB 4
Informatik

Model-Driven Software Migration
Extending SOMA

Andreas Fuhr
Tassilo Horn
Andreas Winter

Nr. 16/2009

**Arbeitsberichte aus dem
Fachbereich Informatik**

Universität Koblenz-Landau • Campus Koblenz • Universitätsstraße 1 • 56070 Koblenz
Telefon +49 261 287-0 • http://www.uni-koblenz.de

Arbeitsberichte des Fachbereichs Informatik

ISSN (Print): 1864-0346
ISSN (Online): 1864-0850

Kontaktdaten der Verfasser

Andreas Fuhr, Tassilo Horn, Andreas Winter
Institut für Softwaretechnik
Fachbereich Informatik
Universität Koblenz-Landau
Universitätsstraße 1
D-56070 Koblenz
EMail: afuhr@uni-koblenz.de, horn@uni-koblenz.de, winter@uni-koblenz.de

Model-Driven Software Migration Extending SOMA

Andreas Fuhr, Tassilo Horn, Andreas Winter
University of Koblenz-Landau
{afuhr,horn,winter}@uni-koblenz.de

This paper proposes model-driven techniques to extend IBM's SOMA method towards migrating legacy systems into Service-Oriented Architectures (SOA). The proposal explores how graph-based querying and transformation techniques enable the integration of legacy assets into a new SOA. The presented approach is applied to the identification and migration of services in an open source Java software system.

1 Introduction

Today, almost every company runs systems that have been implemented a long time ago. These systems, and even those that have been developed in the last years, are still under adaptation and maintenance to address current needs. Very often, adapting legacy software systems to new requirements needs to make use of new technological advances. Business value of existing systems can only be preserved by transferring these legacy systems into new technological surroundings. Migrating legacy systems, i. e. transferring software systems to a new environment without changing the functionality [32], enables already proven applications to stay on stream instead of passing away after some suspensive servicing [30].

A technological advance promising better reusability of software assets in new application areas is provided by *Service-Oriented Architectures (SOA)*. SOA is viewed as an abstract, business-driven approach decomposing software into loosely-coupled *services* that enables reusing existing software assets for rapidly changing business needs [20]. A service is viewed as an encapsulated, reusable and business-aligned asset that comes with a well-defined *service specification* providing an interface description of the requested functionality. The service specification is implemented by a service component which is realized by a *service provider*. Its functionality is used by *service consumers* [2].

Migrating legacy systems to services enables both, the reuse of already established and proven software components and the integration with newly created services, including their orchestration to support changing business needs. The work presented here is part of the SOAMIG[1] project, which addresses the migration of legacy software systems to Service-Oriented Architectures, based on model driven technologies and code transformation.

[1] This work is partially funded by the German Ministry of Education and Research (BMBF) grant 01IS09017C. See http://www.soamig.de for further information.

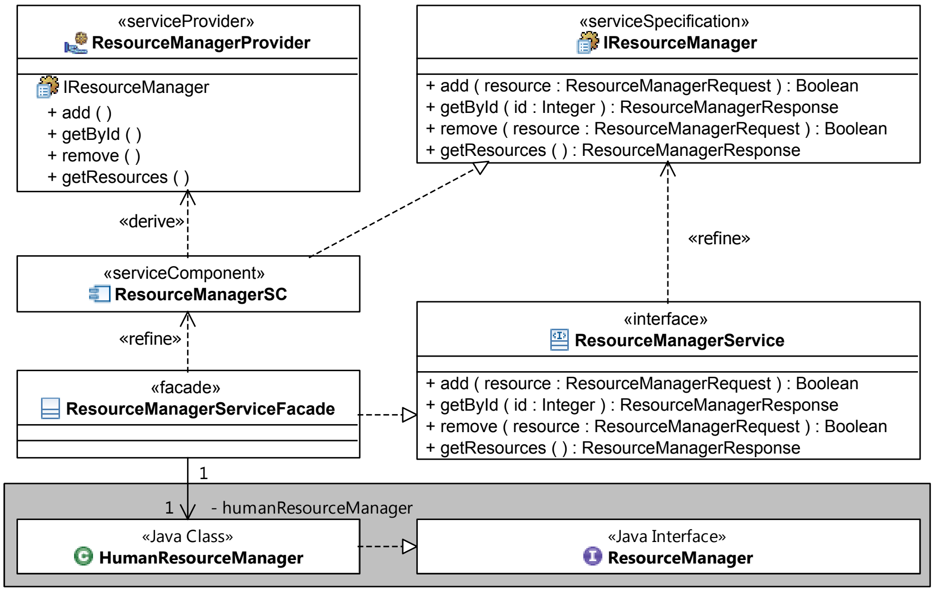

Figure 1: Target architecture of SOA

Software development and maintenance projects require a clearly defined methodology. E. g. Chicken Little [7] provides an incremental approach. The *ReMiP (Reference Migration Process)* provides a generic process model for software migration [1, 19]. Major activities in all software migration processes, dealing with the legacy code, include *legacy analysis* and *legacy conversion*. Legacy analysis aims at understanding legacy systems and identifying software assets worth to be transfered into the new environment. Legacy conversion supports the technical migration of legacy assets by wrapping or transformation.

An end-to-end method to develop SOA systems is given by the *SOMA method* developed by IBM [2]. Service-Oriented Modeling and Architecture (SOMA) includes seven incremental and iterative phases describing how to identify, specify and implement services. In the first place, SOMA is designed to develop SOAs from scratch and does not provide a direct support for integration of legacy assets. However, SOMA is strongly focused on extensibility and allows to include additional techniques to support specific needs. Thus, by extending SOMA with legacy analysis and conversion, SOMA will provide a comprehensive methodology to SOA development including a broad reuse of legacy code assets.

The extension of SOMA towards reusing legacy assets by migration is based on a model-driven strategy. Models (including code) represent different views on software systems including business process models, software architecture and programming code. In particular, migrating legacy systems to SOA requires an integrated view on business processes, architecture and code [35]. Legacy analysis and legacy conversion is based on querying and transforming models, describing appropriate views of the legacy system.

This paper introduces the application of TGraph technology, a graph-based representation funding on TGraphs [12]. It includes querying and transformation techniques to support legacy analysis and legacy conversion within the appropriate SOMA phases. TGraph technology will support identification, specification, realization and implementation of services obtained from legacy systems [16].

The integration of graph-based reengineering and migration techniques to SOMA is explained by identifying and transforming an exemplary service in the open source software GanttProject [17]. It will be described

- how TGraph technology is applied to represent and analyze legacy code supporting service identification and realization decisions,
- how SOMA is applied to specify and design services and
- how TGraph technology is applied to transfer legacy code into a service implementation.

Figure 1 shows the service-oriented target architecture, modeling the embedding of a service providing capabilities to *manage project resources*, obtained from a legacy system supporting project schedules. The upper part of the class diagram shows the service framework that will be created during forward engineering using the extended SOMA approach. The bottom of the diagram (gray box) shows legacy classes and interfaces that will be transformed into the service implementation. Combining both parts via the facade class will provide a fully executable and functional service [15, 16].

The paper is organized as follows: Section 2 describes the SOMA method in more detail and motivates where SOMA has to be extended by model driven reengineering techniques. Section 3 describes the TGraph technology to provide legacy analysis and legacy conversion. In Section 4 the integrated method is applied to identify, to specify, to realize and to implement the resource management service by reusing the GanttProject legacy code. Section 5 shortly, contrasts the integrated SOA migration approach presented here, with current work in model-driven software analysis and migration. Finally, Section 6 summarizes and reflects the obtained results.

2 Service-Oriented Modeling and Architecture (SOMA)

SOMA [2] is an iterative and incremental method to design and implement service-oriented systems, developed by IBM and still under research (latest published version: 2.4). SOMA describes how to plan, to design, to implement and to deploy SOA systems. SOMA is designed extensible to be able to include additional, specialized techniques supporting specific project needs. The following subsections shortly describe the SOMA phases and outline where SOMA has to be extended towards providing software migration, as well.

2.1 Business Modeling and Solution Management

During *Business Modeling* the business at the beginning of a project is analyzed. Business goals and the business vision are identified, as well as business actors and business use cases.

Solution Management adapts the SOMA method to the project needs. This includes choosing additional techniques to solve project-specific problems (like adding migration techniques in a migration project).

SOA migration does not require to extend theses initial SOMA phases.

2.2 Service Identification

During *Service Identification*, SOMA uses three complementary techniques to identify *service candidates*, i. e functionality that may be implemented as service later in the new architecture.

Domain Decomposition is a top-down method decomposing the business domain into functional areas and analyzing the business processes to identify service candidates.

Goal-Service Modeling identifies service candidates by exploring the business goals and subgoals.

Legacy Asset Analysis finally explores the functionality of legacy systems. Documentation, APIs or interfaces are analyzed to identify service candidates. The source code is only analyzed coarse-grained, meaning it is analyzed which functionality exists and not how it is actually implemented.

All three techniques are performed incrementally and iteratively. For each identified candidate, an initial service specification is created and a trace to the source of identification is established.

Extending Service Identification

SOMA does not describe how to analyze legacy systems. At this point, additional methods and techniques have to be included. In Section 4.2, we extend SOMA by a model-driven technique to reverse-engineer legacy code into an appropriate TGraph, which enables queries and transformations to identify service candidates.

2.3 Service Specification

Service Specification deals with describing the service design in detail. The service specification is refined, messages and message flows are designed and services are composed. At the end of this phase, a comprehensive description of the service design exists.

SOMA uses an UML profile for Service-Oriented Architectures to describe the service design. Later, the specification will be transformed into WSDL code for implementing the service as a Web Service (as is proposed by SOMA).

Extending Service Specification

Service Specification describes the service in detail. To gather the information needed for the design, messages and message parameters can be derived from legacy code. We extend SOMA to identify useful legacy code in Section 4.3.

2.4 Service Realization

Service Realization decides which services will be implemented in the current iteration and constitutes how to implement them. First, a *Service Litmus Test (SLT)* is executed to identify service candidates that should be exposed. The SLT is a set of criteria to evaluate usefulness and value of each service.

After having chosen a set of services, the implementation strategy has to be defined. Encapsulation of services allows to choose different ways to implement each service. Common strategies to form new service components are (1) implementation from scratch, (2) wrapping of larger legacy components or (3) transforming the required legacy components.

In software migration it is intended to reuse legacy functionality as far as possible. In SOMA, legacy functions usually are *wrapped* and then exposed as services. This has several drawbacks. The legacy system must still be maintained and in addition, the wrapper must be created and later be maintained, too. A different approach is to *transform* legacy functionality into a service implementation.

After having decided on transformation as implementation technique, legacy systems must be analyzed fine-grained. Functionality that is able to implement services has to be identified in the legacy code. In addition, it is important to clearly understand how this functionality is embedded in the legacy, since it has to be separated to build a self-contained service. Finally, the implementation design specifying how to implement the service, is created. In addition, patterns are used to create a framework which is able to integrate the service implementation into the service design.

Extending Service Realization

SOMA does not describe how to implement services by reusing legacy code. In Section 4.4, a model-driven technique is presented to analyze legacy systems fine-grained in order to understand the implementation of legacy functionality. Here, GReQL graph queries are used to retrieve the required information.

2.5 Service Implementation

During the *Service Implementation* phase, services are actually implemented. According to the decisions derived in the Service Realization phase in Section 2.4, services are developed, wrappers are written, or legacy code is transformed. Finally, all services are orchestrated and message flows are established.

Extending Service Implementation

SOMA does not include techniques to transform legacy code into services. In Section 4.5 it is demonstrated how graph transformations are used to transform legacy code into service implementations.

2.6 Service Deployment

The last phase is *Service Deployment*. It deals with exposing the services to the customer's environment. Final user-acceptance tests are performed and the SOA is monitored to verify that it performs as expected.

In general, service deployment is not affected by the service implementation strategies. Further extensions for the final SOMA phase are not required.

This paragraph concludes the description of the SOMA method. Four extension points to the SOMA method have been identified. The next section will introduce the TGraph approach that is used to extend SOMA. In Section 4, the extended SOMA method is applied to the migration of GanttProject.

3 The TGraph Approach

The *TGraph Approach* [12] is a seamless approach for graph-based modeling and implementation. Most, if not all reverse engineering techniques can be based on graph analysis using *graph algorithms* and/or *graph querying* [25]. Additionally, the representation of models as graphs facilitates the use of graph transformation techniques.

The TGraph approach is based on a strong emphasis on metamodeling. Each graph's nodes and edges are typed and the querying and transformation techniques exploit the accessibility of the metamodel information.

In the following sections, the kind of graphs used in the TGraph approach is described, including a short overview on the metamodeling foundation. The model-driven SOMA extensions motivated in Section 2 require reasonable model querying and transformation techniques. Section 3.2 gives an introduction to querying TGraphs with GReQL and Section 3.3 depicts the GReTL-transformation language.

3.1 TGraphs and TGraph Schemas

A *TGraph* is a directed graph where all nodes and edges are typed and may contain attributes. Additionally, edges and nodes are ordered. Edges are first class citizens, so the navigability is always bidirectional and does not depend on the edge's direction. This also enables reasoning on edges directly. In sparse graphs, which usually occur in code and model representations, this also provides more efficient graph traversal, instead of arguing on node tuples.

The graph library *JGraLab (Java Graph Laboratory*[2]) provides a convenient and efficient API for accessing and manipulating TGraphs.

Each TGraph is an instance of a *TGraph schema*. In a model-driven sense, TGraph schemas form metamodels for classes of TGraphs and define edge and node types, including their attribute bindings. Such schemas are specified by using a UML profile called *grUML (Graph UML)*, a tool-ready subset of CMOF slightly more expressive than EMOF [6]. In grUML diagrams, node and edge types and their attributes are specified with UML classes and associations (or association classes). Multiple inheritance between both node and edge types is supported.

Among others, a schema covering the complete abstract syntax of the Java programming language exists. Using a custom parser [4], Java source code, class and jar files can be converted to a TGraph conforming to this schema. These graphs are subject to advanced analysis and transformation using the query and transformation languages described in the next sections.

3.2 GReQL

GReQL (*Graph Repository Query Language*, [5]) is a textual language and its syntax bears some analogies to SQL. One of the most commonly used language elements is the *from-with-report (FWR)* clause. The **from** part is used to declare *variables* and bind them to *domains*. In the **with** part, *constraints* can be imposed on the values of these variables. The **report** part is used to define the structure of the query result.

A sample query for retrieving all super- and subclasses of a class with name HumanResource in a graph conforming to the Java schema is depicted in Listing 1.

In the **from** part, the variable e is bound to all edges of type IsSuperClassOfClass one after the other. The constraint defined in the **with** clause requires that the name attribute of the node acting as source or target of such an edge matches the regular expression "HumanResource". The **report** clause defines the structure of the results as tuples where the entries are pairs of the names of the superclass and subclass. For each IsSuperClassOfClass edge which satisfies the constraint, a tuple is added to the result multiset.

```
from e : E{IsSuperClassOfClass}
with startVertex(e).name = "HumanResource" or
     endVertex(e).name = "HumanResource"
report startVertex(e).name, endVertex(e).name
end
```

Listing 1: A GReQL query to find direct superclasses and subclasses

One of GReQL's especially powerful features are *regular path expressions*, which can be used to formulate queries that utilize the interconnections between nodes and their structure. Therefore, symbols for edges are introduced: −−> for directed edges and <−> if the direction should not be taken

[2] http://jgralab.uni-koblenz.de

into account. Additionally, an edge type written in angle brackets may follow the edge symbol. These symbols can be combined using regular operators: sequence, iteration ($*$ and $+$), alternative ($|$) and so on.

Listing 2 shows a query for finding member classes. Two variables of type ClassDefinition are defined.

```
from o, m : V{ClassDefinition}
with o <--{IsClassBlockOf} <--{IsMemberOf} m
reportSet m end
```

Listing 2: A query using regular path expressions to find member classes

If the ClassDefinition m is a member of o (e.g. a path like the one depicted in the **with** clause exists), the member class m will be reported.

3.3 GReTL

The *GReTL* transformation language (*Graph Repository Transformation Language*) is a Java framework for programming transformations on TGraphs [11]. Instead of creating a new transformation language including its own syntax from scratch, existing technologies were applied, namely JGraLab's Schema API for describing imperative aspects and GReQL for declarative parts. The idea of GReTL is to build a target TGraph schema by writing transformation rules as calls to methods provided by the transformation framework. These methods create new elements in the target schema by delegating to methods in JGraLab's Schema API and GReQL queries given as additional parameters in transformation rules specify declaratively which instances of this new type have to be created in the target graph.

An example rule for creating a node class in the target schema and its appropriate instances is depicted in figure Listing 3.

```
createVertexClass("uml.Class",
      "from t : V{Type}                    "
    + "with t.name =~ \".*[Rr]esource.*\"   "
    + "reportSet t end                     ");
```

Listing 3: GReTL rule for creating a node class and instances thereof

The first parameter uml.Class is the fully qualified name of the new node class to be created in the target schema. The second parameter is a GReQL query given as string, which is evaluated on the source graph and returns the set of Types whose name contain the substring “resource”. These types are used as *archetypes* for the uml.Class nodes that are created in the target graph. For each Type in the result set, a new uml.Class node is created in the target graph. The mapping of archetype to the newly created node is saved and accessible in further rules.

Further methods for creating edge types (including their edge instances), attributes and generalizations between edge and node classes are realized in an analogous manner.

4 Merging SOMA and the TGraph Approach

The previous sections motivated the need of extending SOMA to enable the reuse of legacy software assets in software migration and shortly presented graph-based modeling, analysis and transformation

```
protected void transform() {
  VertexClass umlClass = createVertexClass(
    "uml.Class",
    "           from t : V{Type}     "
        + "with t.name =~ \".*[Rr]esource.*\" "
        + "reportSet t end ");
  createAttribute("name", umlClass, createStringDomain(),
    "           from t : keySet(img_uml$Class) "
        + "reportMap t, t.name end");
  createEdgeClass(
    "uml.Association",
    umlClass,
    umlClass,
    "           from c : keySet(img_uml$Class), c2 : keySet(img_uml$Class) "
        + "with c <--{IsBlockOf} <--{IsMemberOf} "
        + "        <--{^IsBreakTargetOf,^IsContinueTargetOf,^IsTypeDefinitionOf,^
         IsClassBlockOf,^IsInterfaceBlockOf}* "
        + "        [<--{IsTypeDefinitionOf}] c2    "
        + "reportSet c, c2 end",
    "           from t : $                   "
        + "reportMap t, nthElement(t, 0) end",
    "           from t : $$                  "
        + "reportMap t, nthElement(t, 1) end");
  createEdgeClass(
    "uml.IsA",
    umlClass,
    umlClass,
    "           from c : keySet(img_uml$Class), c2 : keySet(img_uml$Class) "
        + "with c (<--{IsSuperClassOf} | <--{IsInterfaceOfClass}) <--{IsTypeDefinitionOf}
         c2 "
        + "reportSet c, c2 end",
    "           from t : $                   "
        + "reportMap t, nthElement(t, 0) end",
    "           from t : $$                  "
        + "reportMap t, nthElement(t, 1) end");
}
```

Listing 4: GReTL transformation from Java to UML

techniques. The migration approach resulting in the extension of SOMA by TGraph-based reengineering techniques is applied to identify services from legacy code, to support specification and realization decisions and to transform legacy code into service implementation.

Following the SOMA phases introduced in Section 2, the integrated approach is applied to the migration of Gantt project into a Service-Oriented Architecture [15]. GanttProject [17] is a project planning tool. It manages project resources and tasks and displays project schedules as Gantt charts. GanttProject is a Java system containing about 1200 classes. The required migration is exemplified by identifying and migrating a *service to manage project resources* by transforming the legacy code.

4.1 Business Modeling and Solution Management

The first phase in SOMA analyzes the current state of the business (Business Modeling). This paper focuses on the analysis and reuse of legacy software. Business modeling is not considered in detail, although it is important to analyze legacy business processes and to define the processes to be supported

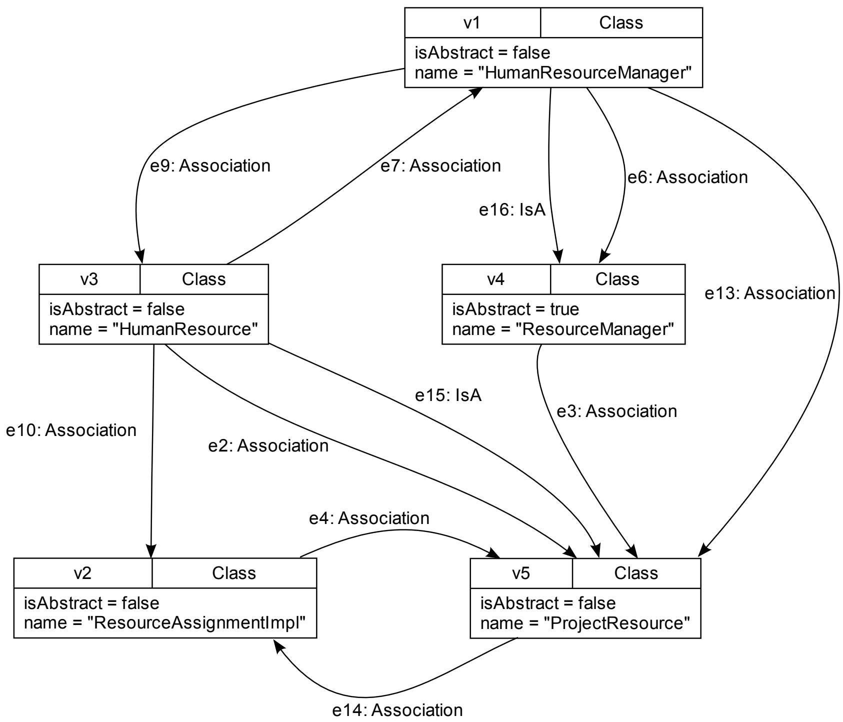

Figure 2: Visualization of classes and interfaces possibly providing functionality to manage resources

by the new SOA. Here, it is assumed that the business process of managing project resources has to be realized by the new SOA and its implementation will rely on GanttProject.

Solution Management adapts the SOMA method to the current project needs. Since GanttProject is a Java system, a TGraph representation for Java systems is required. The TGraph Java 6 metamodel contains about 90 vertex and 160 edge types and covers the complete Java syntax. The GanttProject sources are parsed according to that metamodel, resulting in a graph of 274.959 nodes and 552.634 edges. This graph and the implicit knowledge on resource management provide the foundation for service identification, service specification, service realization and service implementation.

4.2 Service Identification

The identification of services from legacy systems requires a coarse-grained analysis. The graphical user interface of GanttProject is explored first and functionality to manage *project resources* is identified as one main feature of the software. Looking at the legacy code identifies the functionality providing the management of project resources.

Identifying functionality in legacy code is a challenging task and still an open research issue [24]. A GReQL query is used to identify this functionality in the GanttProject-TGraph and a corresponding GReTL transformation visualizes the query result. String search on TGraphs is used to detect possible code areas referring to "resources" and further interconnections of code objects are specified by declarative path expressions. The resulting subgraph is transformed by GReTL into a TGraph conforming to a simple UML schema. Further XMI-based filters (cf. [13]) might be used to render these structures in UML tools.

Listing 4 shows the GReTL transformation supporting coarse-grained legacy code analysis. For each legacy class or interface named "resource" (name =~ \".*[Rr]esource.*\"), this transformation creates

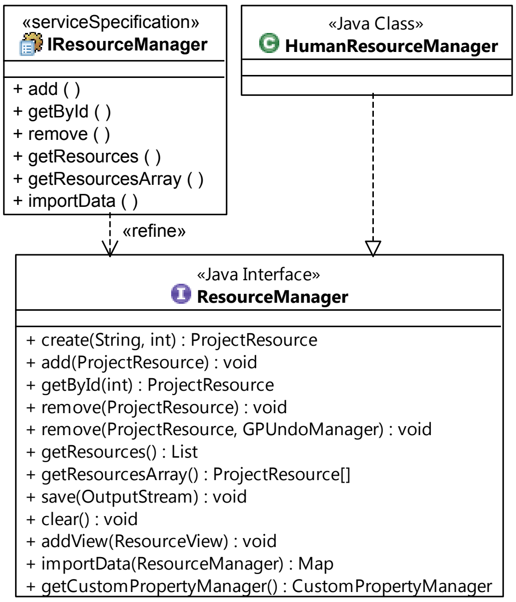

Figure 3: IResourceManager service identified from legacy code

one UML-class-node in the target TGraph. In addition, associations are drawn between those class-nodes whenever one node uses (e. g. by method calls or variable types) another node. Inheritance is visualized by “IsA” edges. For interfaces and abstract Java classes, their UML class counterparts are marked by appropriate attributes. The visualized result of this GReTL transformation is shown in the TGraph in Figure 2.

Looking at the result, the class HumanResourceManager implementing the interface ResourceManager can be identified as functionality to manage project resources. Based on this information, an initial service specification for the service candidate IResourceManager is created and traces to the legacy code are noted (Figure 3). In this phase, no further information about the method signatures of the initial service specification is gathered.

The following SOMA phases specify the IResourceManager service in more detail.

4.3 Service Specification

Service Specification refines the IResourceManager service specification. A *service provider* component is created which will later implement the service specification.

In addition, message flows are created to enable communication with the service. For *method parameters* in the legacy interface, *request messages* are created that are passed to the service. For *return types* in the legacy system, *response messages* are defined that will be returned by the new service. Request and response messages can be derived from legacy code.

Listing 5 shows a GReQL query taking an interface or class name as input and returning method parameters and return types as output. This information is used to derive message parameter types from legacy code.

The result of the specification phase is shown in the class diagram in Figure 4. The service specification now contains information about parameters (They are hidden in the ResourceManagerProvider component since they are already shown in the service specification). In addition, request and response

```
let classname :="HumanResourceManager" in tup(
from hrmClass : V{ClassDefinition},
     usedType : V{Type, BuiltInType}
with  // method parameters of type usedType
 hrmClass.name = classname and hrmClass
 <--{IsClassBlockOf}<--{IsMemberOf}
 <--{IsParameterOfMethod}
 <--{IsTypeOfParameter}
 [<--{IsTypeDefinitionOf}] usedType
reportSet (hasType(usedType, "BuiltInType")) ?
 usedType.type :
 theElement(usedType<--&{Identifier}).name :
 "Error"
end, from hrmClass : V{ClassDefinition},
          usedType : V{Type, BuiltInType}
with // return types of type usedType
 hrmClass.name = classname and hrmClass
 <--{IsClassBlockOf} <--{IsMemberOf}
 <--{IsReturnTypeOf} [<--{IsTypeDefinitionOf}]
 usedType
reportSet (hasType(usedType, "BuiltInType")) ?
 usedType.type :
 theElement(usedType<--&{Identifier}).name :
 "Error"
end)
```

Listing 5: GReQL query retrieving method parameters and return types for message specification

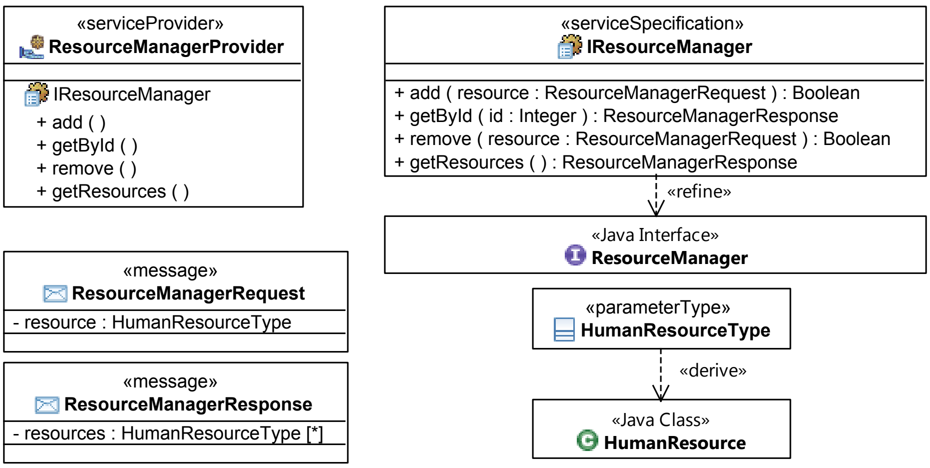

Figure 4: Detailed design of IResourceManager service

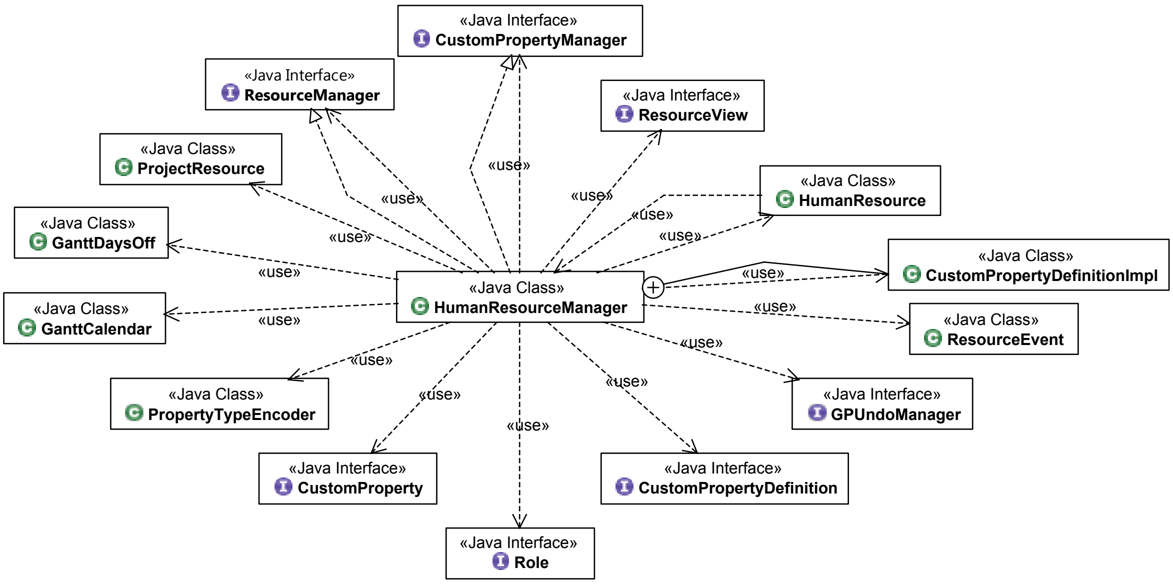

Figure 5: Service Realization: Dependencies of HRM class

messages are defined and one parameter type (HumanResourceType) for these messages has been derived from legacy code.

At the end of this phase, the design of the service itself is mostly completed. The next step is now to decide how the service will be implemented.

4.4 Service Realization

The first decision to make during Service Realization is how to implement the IResourceManager service. Model transformation approaches are also suited for code transformation. Thus, the legacy code here is transformed into a service implementation to provide the business functionality. If service realization by wrapping is decided, wrappers can be generated analogously.

Service Identification already identified one class in the legacy code that may provide functionality to the IResourceManager service: the class HumanResourceManager (short: HRM). The complete but minimal code realizing this functionality has to be determined and transformed into executable code. Slicing these code fragments also requires to consider dependencies of HRM. These dependencies include

- HRM calls methods of other classes (HRM $\rightarrow_{calls}$ method $\rightarrow_{isMemberOf}$ class),
- variables, parameters or return types of HRM (e.g. HRM $\rightarrow_{defines}$ variable $\rightarrow_{hasAsType}$ class),
- inheritance hierarchy (HRM $\rightarrow_{specializes}$ class or HRM $\rightarrow_{implements}$ interface).

Listing 6 describes the GReQL query retrieving these dependencies. It returns a list of all classes and interfaces that HRM depends on. Figure 5 shows a (manually created) visualization of the query result.

Next, the business functionality must be integrated into the overall service design. This is done according to the patterns proposed by Wahli [34].

```
from hrmClass : V{ClassDefinition},
  hrmMethod : V{MethodDefinition},
  usedType  : V{Type}
with
  hrmClass.name = "HumanResourceManager"
  and hrmClass <--{IsClassBlockOf}<--{IsMemberOf} hrmMethod
  and
  (
    hrmMethod ( ( <--{IsBodyOfMethod} <--{IsStatementOfBody}
      (<--{AttributedEdge,^IsBreakTargetOf,^IsContinueTargetOf,^IsTypeDefinitionOf})*
      & {MethodInvocation} <--{IsDeclarationOfInvokedMethod} & {MethodDefinition}
      -->{IsMemberOf} -->{IsClassBlockOf}) |
      (<--{IsParameterOfMethod} <--{IsTypeOf}+ <--{IsTypeDefinitionOf}) |
      (<--{IsBodyOfMethod} <--{IsStatementOfBody}
      (<--{AttributedEdge,^IsBreakTargetOf,^IsContinueTargetOf,^IsTypeDefinitionOf})*
      <--{IsTypeOfVariable} <--{IsTypeDefinitionOf} |
      (<--{IsReturnTypeOf} <--{IsTypeDefinitionOf})) usedType
  or
    hrmClass ( (
      <--{IsClassBlockOf} <--{IsMemberOf} <--{IsFieldCreationOf} <--{IsTypeOfVariable}
      <--{IsTypeDefinitionOf} ) |
      ((<--{IsSuperClassOfClass} | <--{IsInterfaceOfClass})
      <--{IsTypeDefinitionOf}) |
      ((<--{IsClassBlockOf} <--{IsMemberOf})+)) usedType
  )
reportSet theElement(usedType <-- & {Identifier}).name end
```

Listing 6: GReQL query retrieving dependencies

Figure 6 shows the application of these patterns to create a framework to integrate the legacy code which will be transformed in the next phase. The *service component* ResourceManagerSC implements the service specification. A facade pattern is used to implement the service component. The facade class delegates service requests to the appropriate service implementation, in this example the HRM class, and all its dependencies revealed by the GReQL query.

This phases finishes the design of the service. The next step is to implement this design.

4.5 Service Implementation

Service Implementation realizes the IResourceManager service, e. g. as Web Service (as is supposed by SOMA). Migrating identified source code (cf. Section 4.4) to realize the resource management service combines functionality provided by the IBM Rational Software Architect for WebSphere Software V7.5[3] (RSA) and TGraph technology.

First, the code generation capabilities of the RSA are used to create WSDL code (interface description language for Web Services) from the service specification. WSDL is later used to specify the service interface. Next, the design of the service framework (UML diagram in Figure 6 which includes service component, facade pattern and facade interface) is transformed into Java.

So far, the service lacks of business functionality, which will be added by transforming legacy code into a service implementation. The GReQL query described in Listing 6 (Section 4.4) is used to mark the HRM class and all legacy software components it depends on. The Java code-generator of JGraLab

[3]IBM, Rational and WebSphere are trademarks of International Business Machines Corporation.

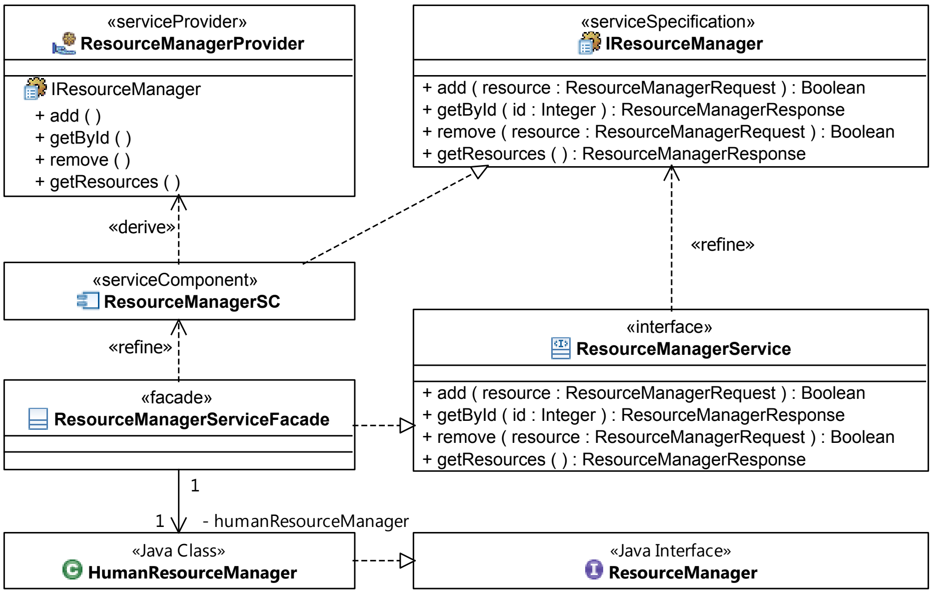

Figure 6: Implementation design of IResourceManager service

is used to generate Java code for all marked classes of the TGraph. This results in Java classes implementing the business functionality of the IResourceManager service. These classes are connected to the service framework. For this purpose, the facade class must be edited manually to delegate service requests to the HRM class. In addition, the facade class translates *message parameters* into *objects* known by the HRM class.

Finally, the *Web Service Wizard* of RSA was used to generate a fully functional Web Service. This wizard takes the WSDL interface description and the Java classes of the service framework and the service implementation and creates the Java EE Web Service implementation.

4.6 Service Deployment

The Web Service created in the last subsection is deployed to the customer. Figure 7 shows a screenshot of RSA's *Web Service Explorer (WSE)*. The WSE provides a simple web interface to test the Web Service. The WSE allows to simulate service requests and to visualize the response of the Web Service. Figure 7 shows the adding of a new project resource. As result, the IResourceManager service returned **true**, signaling that the project resource has successfully been added.

5 Related Work

Model-driven technologies are widely applied in software reengineering activities. Various approaches representing code and higher level system information, including comprehensive analysis techniques already exist. Here, relational structures, object oriented structures and graphs provide most appropriated formal foundations, which in general can be mapped to graph-based structures (cf. e. g. the GXL graph-based interchange format for reengineering data [21]). The presented SOMA extension is strongly based on TGraph modeling, which — in contrast to object oriented approaches — views edges as first class citizens that can be addressed directly and provide more efficient graph traversal means

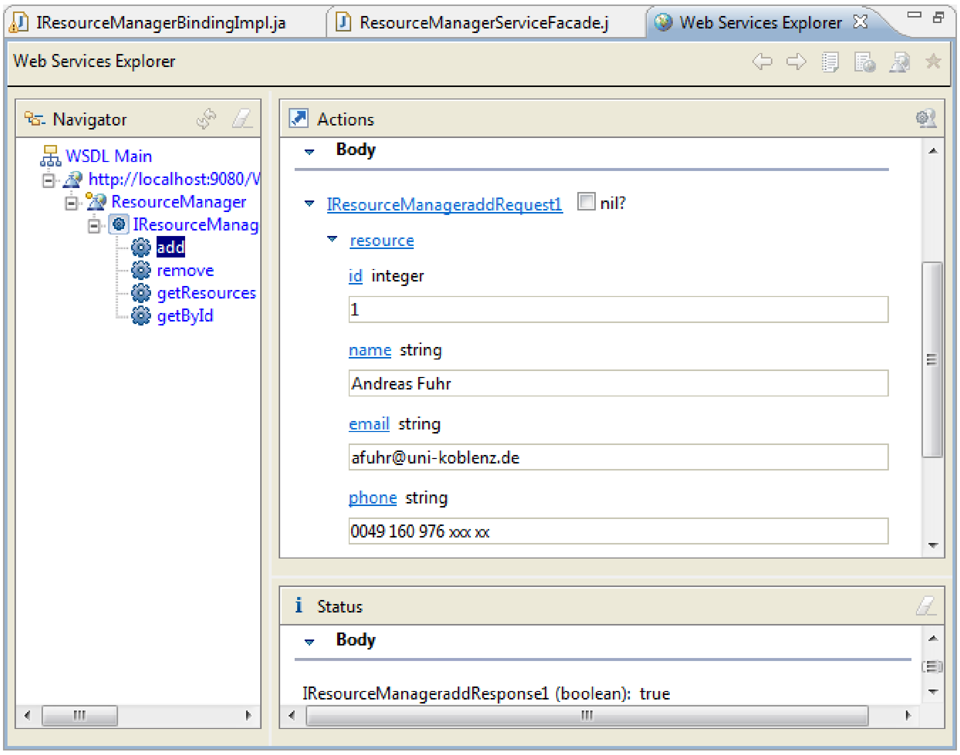

Figure 7: Visualization of functional Web Service

[12].

OMG´s Model-Driven Architecture [28] is one way to realize model-driven approaches. In addition, the Architecture-Driven Modernization (ADM, [23]) initiative applied model-driven approaches for representing programing code in software reengineering. But, an integrated representation for code and more abstract view on software systems is missing so far. The extended SOMA approach presented here, funds on a (currently implicitly) represented model combining business process models and code models. Further efforts in the SOAMIG project will address an explicitly defined (cf. [22]) integrated model.

Furthermore, techniques for analyzing and transforming models are also established. In the model driven vein, e. g. ATLAS Transformation Language (ATL, [3]) and OMG´s Query/View/Transformation Specification (QVT, [29]) reflect the current MDA state-of-the-art. Transformation techniques based on grammars and/or term systems are provided by the TXL [8] or by Stratego [33]. These approaches are most applicable to the transformation of code, whereas application to graphical modeling languages requires extensive mappings to textual languages. Query- and transformation techniques used in this work, are directly based on TGraphs (instead of object networks) which view edges as first class citizens. TGraph technology has shown their applicability to both, visual models [10] and code, in reengineering [25] activities.

Whereas a plethora on publication on the development of Service-Oriented Architectures exists, migrating legacy systems to SOA is only addressed in a few papers. The SMART approach [31] deals with the planning of SOA migration projects, but does not provide concrete migration or migration tool support. Correia et al. [9] and Fleurey et al. [14] describe general approaches of model-driven migration into a new technology not especially focused on SOA. Correia et al. describe a graph-based approach which mentions SOA as possible target architecture [27]. In contrast to SOAMIG, this approach focuses on annotating functionality in legacy code instead of directly identifying services from source code. Marchetto and Ricca [26] propose an approach to migrate legacy systems into a SOA step by step. However, this approach does not focus on model-driven techniques and uses wrapping as general migration strategy. Another approach focusing on wrapping is described in [18].

In contrast to these approaches, the work presented here provides a coherent model-driven approach to software migration by integrating an established SOA forward-engineering approach with graph-based reengineering technologies. In addition, in SOAMIG software systems are viewed at all levels of abstraction including business processes and code.

6 Conclusion and Future Work

In this paper, we described a model-driven approach to migrate legacy systems, extending IBM's SOMA method. The approach was applied to the migration of GanttProject towards a Service-Oriented Architecture. This example demonstrated the identification and specification of services by analyzing legacy code, the identification of responsible functionality in legacy code and the transformation of legacy code into a service implementation. As result, a fully functional Web Service was generated, whose business functionality was implemented by transforming legacy code.

As part of the SOAMIG project, we will continue research on model-driven migration into SOA. The results of this first proof-of-concept will have to be confirmed on larger examples and additional research is needed to enable automation of the process. In collaboration with industrial project partners, their Java and COBOL systems will be migrated into Service-Oriented Architectures.

Acknowledgements

We want to express our thanks to Rainer Gimnich, IBM Software Group, Frankfurt/Main, Germany for his support in understanding SOMA and various fruitful discussions on applying SOMA in software migration.

About the Authors

Andreas Fuhr

Andreas Fuhr studied computer science (B. Sc.) at Johannes-Gutenberg University of Mainz (Germany). In his bachelor thesis, he explored the potential of extending IBM's SOMA method by model-driven migration techniques. Since April 2009, he is studying computer science (M. Sc.) at the University of Koblenz-Landau (Germany). In addition, he is working as research assistant at the Institute for Software Technology on the SOAMIG project. His main research interests are software engineering, model-driven software development and software architectures.

Tassilo Horn

Tassilo Horn studied computer science at the University of Koblenz-Landau. In his diploma thesis, he developed an optimizer for the graph query language GReQL, which performs transformations on the syntax graph of the query. Some transformations, such as "selection as soon as possible" are also well known in other domains like databases, while others are special to the domain of GReQL and TGraphs. Since June 2008 he works as a scientific staff member and PhD student at the Institute for Software

Technology. His main research interests include graph querying and transformation techniques and their application in software maintenance.

Dr. Andreas Winter

Andreas Winter acts as senior researcher and teacher of Software Engineering at the Institute of Computer Science of University of Koblenz-Landau. Current research topics include software engineering, metamodeling and transformation, service-oriented tool interoperability, software migration and software-(re-)engineering processes.

From April 2006 to September 2008 Andreas acted as a substitute for the chair of Software Engineering at the Institute for Computer-Science of Johannes-Gutenberg-University, Mainz. Prior to his appointment in Mainz, he was already affiliated with University of Koblenz-Landau and University of Waterloo, Canada.

Andreas served in committees of various software engineering and reengineering conferences and workshops and co-organized various national and international events. He served as program-chair of the 12th and 13th European Conference on Software Maintenance and Reengineering. He is chairperson of the software-reengineering interest group of the German computer society and member in the Steering Committees of the International Conference on Software Language Engineering and the European Conference on Software Maintenance and Reengineering.

References

[1] E. Ackermann, "Ein Referenz-Prozessmodell zur Software-Migration," University of Koblenz-Landau, Koblenz, Diploma Thesis, 2005.

[2] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley, "SOMA: A method for Developing Service-Oriented Solutions," *IBM Systems Journal*, vol. 47, no. 3, pp. 377–396, 2008.

[3] ATLAS Group. (2009) ATL: User Guide. [Online]. Available: http://wiki.eclipse.org/ATL/User_Guide

[4] A. Baldauf and N. Vika, "Java-Faktenextraktor für Gupro," University of Koblenz-Landau, Koblenz, Studienarbeit, 2009.

[5] D. Bildhauer and J. Ebert, "Querying Software Abstraction Graphs," in *Working Session on Query Technologies and Applications for Program Comprehension (QTAPC), collocated with ICPC*, 2008.

[6] D. Bildhauer, H. Schwarz, S. Strauss, V. Riediger, and T. Horn, "grUML – A UML based modelling language for TGraphs," 2009, Project Report, unpublished.

[7] M. L. Brodie and M. Stonebraker, *Migrating Legacy Systems, Gateways, Interfaces & The Incremental Approach*. San Francisco: Morgan Kaufmann, 1995.

[8] J. R. Cordy, "TXL: A Language for Programming Language Tools and Applications," in *Proc. LDTA, ACM 4th International Workshop on Language Descriptions, Tools and Applications Barcelona*, 2004, pp. 1–27.

[9] R. Correia, C. Matos, R. Heckel, and M. El-Ramly, "Architecture Migration Driven by Code Categorization," in *Software Architecture, First European Conference, ECSA, Aranjuez, Spain, September 24-26, Proceedings*, ser. Lecture Notes in Computer Science, Flávio Oquendo, Ed., vol. 4758. Springer, 2007.

[10] J. Ebert, R. Süttenbach, and I. Uhe, "Meta-CASE in Practice: a Case for KOGGE," in *Advanced Information Systems Engineering, 9th international Conference, CAiSE'97*, ser. LNCS. Springer, 1997, vol. 1250, pp. 203–216.

[11] J. Ebert and T. Horn, "The GReTL Transformation Language," 2009, Project Report, unpublished.

[12] J. Ebert, V. Riediger, and A. Winter, "Graph Technology in Reverse Engineering: The TGraph Approach," in *10th Workshop Software Reengineering: Bad Honnef*, ser. GI-EditionProceedings, vol. 126. Bonn: Ges. f. Informatik, 2008, pp. 67–81.

[13] J. Ebert and A. Winter, "Using Metamodels in Service Interoperability," in *Postproceedings of 13th Annual International Workshop on Software Technology and Engineering Practice (STEP'05)*, Y. Zou and M. DiPenta, Eds. IEEE Computer Society, 2006, pp. 147–156.

[14] F. Fleurey, E. Breton, B. Baudry, A. Nicolas, and J.-M. Jezequel, "Model-driven Engineering for Software Migration in a Large Industrial Context," in *Model Driven Engineering Languages and Systems: 10th international conference, MODELS, Nashville, USA; proceedings*, ser. International Conference on Model Driven Engineering Languages and Systems, vol. 4735. Berlin: Springer, 2007, pp. 482–497.

[15] A. Fuhr, "Model-driven Software Migration into a Service-oriented Architecture," Johannes-Gutenberg University, Mainz, Bachelor Thesis, 2009.

[16] A. Fuhr, T. Horn, A. Winter, and R. Gimnich, "Extending SOMA for Model-Driven Software Migration into SOA," in *11. Workshop Software-Reengineering, Bad Honnef*, 2009.

[17] GanttProject. (2009) The GanttProject. [Online]. Available: http://ganttproject.biz/

[18] R. Gimnich, "SOA Migration: Approaches and Experience," *Softwaretechnik-Trends*, vol. 27, no. 1, 2007.

[19] T. Gipp and A. Winter, "Applying the ReMiP to Web Site Migration," in *Proceedings Ninth IEEE International Symposium on Web Site Evolution, Paris, France (WSE)*. IEEE Computer Society, 2007, pp. 9–13.

[20] N. Gold, C. Knight, A. Mohan, and M. Munro, "Understanding Service-Oriented Software," *IEEE Softw.*, vol. 21, no. 2, pp. 71–77, 2004.

[21] R. C. Holt, A. Schürr, S. Elliott Sim, and A. Winter, "GXL: A graph-based standard exchange format for reengineering," *Science of Computer Programming*, vol. 60, no. 2, pp. 149–170, 2006.

[22] D. Jin, J. R. Cordy, and T. R. Dean, "Where's the Schema? A Taxonomy of Patterns for Software Exchange," in *10th International Workshop on Program comprehension*. Los Alamitos: IEEE Computer Society, 2002, pp. 65–74.

[23] V. Khusidman and W. Ulrich, "Architecture-Driven Modernization: Transforming the Enterprise," 2007.

[24] K. Kontogiannis, G. A. Lewis, D. B. Smith, M. Litoiu, H. Müller, S. Schuster, and E. Stroulia, "The Landscape of Service-Oriented Systems: A Research Perspective," in *Proceedings of the International Workshop on Systems Development in SOA Environments*. IEEE Computer Society, 2007.

[25] B. Kullbach and A. Winter, "Querying as an Enabling Technology in Software Reengineering," in *Proceedings of the 3nd European Conference on Software Maintenance and Reengineering*. Los Alamitos: IEEE Computer Society, 1998, pp. 42–50.

[26] A. Marchetto and F. Ricca, "Transforming a Java application in a equivalent Web-services based application: Toward a Tool Supported Stepwise Approach," in *Proceedings Tenth IEEE International Symposium on Web Site Evolution, Beijing, China (WSE)*. IEEE Computer Society, 2008.

[27] C. Matos, "Service Extraction from Legacy Systems," in *Graph Transformations: 4th International Conference, ICGT, Leicester, United Kingdom; proceedings*, D. Hutchison, H. Ehrig, R. Heckel, T. Kanade, and J. Kittler, Eds., vol. 5214. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 505–507.

[28] OMG, "MDA Guide Version 1.0.1," 2003.

[29] ——, "Meta Object Facility (MOF) 2.0: Query/View/Transformation Specification: Final Adopted Specification," 2007.

[30] V. T. Rajlich and K. H. Bennett, "A Staged Model for the Software Life Cycle," *Computer*, vol. 33, no. 7, pp. 66–71, 2000.

[31] D. B. Smith, "Migration of Legacy Assets to Service-Oriented Architecture Environments," in *Companion to the proceedings of the 29th International Conference on Software Engineering*. IEEE Computer Society, 2007, pp. 174–175. [Online]. Available: http://dx.doi.org/10.1109/ICSECOMPANION.2007.48

[32] H. M. Sneed and S. Opferkuch, "Training and Certifying Software Maintainers," in *12th European Conference on Software Maintenance and Reengineering, CSMR, Athens, Greece*. IEEE Computer Society, 2008, pp. 113–122.

[33] E. Visser, "Stratego: A Language for Program Transformation based on Rewriting Strategies. System Description of Stratego 0.5," in *Rewriting Techniques and Applications (RTA'01), LNCS 2051, Berlin*, A. Middeldorp, Ed., 2001, pp. 357–361.

[34] U. Wahli, *Building SOA Solutions Using the Rational SDP*, ser. IBM Redbooks. IBM International Technical Support Organization, 2007.

[35] A. Winter and J. Ziemann, "Model-based Migration to Service-oriented Architectures: A Project Outline," in *CSMR, 11th European Conference on Software Maintenance and Reengineering, Workshops*, H. M. Sneed, Ed., 2007, pp. 107–110.

Bisher erschienen

Arbeitsberichte aus dem Fachbereich Informatik

(http://www.uni-koblenz.de/fb4/publikationen/arbeitsberichte)

Andreas Fuhr, Tassilo Horn, Andreas Winter, Model-Driven Software Migration Extending SOMA, Arbeitsberichte aus dem Fachbereich Informatik 16/2009

Eckhard Großmann, Sascha Strauß, Tassilo Horn, Volker Riediger, Abbildung von grUML nach XSD soamig, Arbeitsberichte aus dem Fachbereich Informatik 15/2009

Kerstin Falkowski, Jürgen Ebert, The STOR Component System Interim Report, Arbeitsberichte aus dem Fachbereicht Informatik 14/2009

Sebastian Magnus, Markus Maron, An Empirical Study to Evaluate the Location of Advertisement Panels by Using a Mobile Marketing Tool, Arbeitsberichte aus dem Fachbereich Informatik 13/2009

Sebastian Magnus, Markus Maron, Konzept einer Public Key Infrastruktur in iCity, Arbeitsberichte aus dem Fachbereich Informatik 12/2009

Sebastian Magnus, Markus Maron, A Public Key Infrastructure in Ambient Information and Transaction Systems, Arbeitsberichte aus dem Fachbereich Informatik 11/2009

Ammar Mohammed, Ulrich Furbach, Multi-agent systems: Modeling and Virification using Hybrid Automata, Arbeitsberichte aus dem Fachbereich Informatik 10/2009

Andreas Sprotte, Performance Measurement auf der Basis von Kennzahlen aus betrieblichen Anwendungssystemen: Entwurf eines kennzahlengestützten Informationssystems für einen Logistikdienstleister, Arbeitsberichte aus dem Fachbereich Informatik 9/2009

Gwendolin Garbe, Tobias Hausen, Process Commodities: Entwicklung eines Reifegradmodells als Basis für Outsourcingentscheidungen, Arbeitsberichte aus dem Fachbereich Informatik 8/2009

Petra Schubert et. al., Open-Source-Software für das Enterprise Resource Planning, Arbeitsberichte aus dem Fachbereich Informatik 7/2009

Ammar Mohammed, Frieder Stolzenburg, Using Constraint Logic Programming for Modeling and Verifying Hierarchical Hybrid Automata, Arbeitsberichte aus dem Fachbereich Informatik 6/2009

Tobias Kippert, Anastasia Meletiadou, Rüdiger Grimm, Entwurf eines Common Criteria-Schutzprofils für Router zur Abwehr von Online-Überwachung, Arbeitsberichte aus dem Fachbereich Informatik 5/2009

Hannes Schwarz, Jürgen Ebert, Andreas Winter, Graph-based Traceability – A Comprehensive Approach. Arbeitsberichte aus dem Fachbereich Informatik 4/2009

Anastasia Meletiadou, Simone Müller, Rüdiger Grimm, Anforderungsanalyse für Risk-Management-Informationssysteme (RMIS), Arbeitsberichte aus dem Fachbereich Informatik 3/2009

Ansgar Scherp, Thomas Franz, Carsten Saathoff, Steffen Staab, A Model of Events based on a Foundational Ontology, Arbeitsberichte aus dem Fachbereich Informatik 2/2009

Frank Bohdanovicz, Harald Dickel, Christoph Steigner, Avoidance of Routing Loops, Arbeitsberichte aus dem Fachbereich Informatik 1/2009

Stefan Ameling, Stephan Wirth, Dietrich Paulus, Methods for Polyp Detection in Colonoscopy Videos: A Review, Arbeitsberichte aus dem Fachbereich Informatik 14/2008

Tassilo Horn, Jürgen Ebert, Ein Referenzschema für die Sprachen der IEC 61131-3, Arbeitsberichte aus dem Fachbereich Informatik 13/2008

Thomas Franz, Ansgar Scherp, Steffen Staab, Does a Semantic Web Facilitate Your Daily Tasks?, Arbeitsberichte aus dem Fachbereich Informatik 12/2008

Norbert Frick, Künftige Anfordeungen an ERP-Systeme: Deutsche Anbieter im Fokus, Arbeitsberichte aus dem Fachbereicht Informatik 11/2008

Jürgen Ebert, Rüdiger Grimm, Alexander Hug, Lehramtsbezogene Bachelor- und Masterstudiengänge im Fach Informatik an der Universität Koblenz-Landau, Campus Koblenz, Arbeitsberichte aus dem Fachbereich Informatik 10/2008

Mario Schaarschmidt, Harald von Kortzfleisch, Social Networking Platforms as Creativity Fostering Systems: Research Model and Exploratory Study, Arbeitsberichte aus dem Fachbereich Informatik 9/2008

Bernhard Schueler, Sergej Sizov, Steffen Staab, Querying for Meta Knowledge, Arbeitsberichte aus dem Fachbereich Informatik 8/2008

Stefan Stein, Entwicklung einer Architektur für komplexe kontextbezogene Dienste im mobilen Umfeld, Arbeitsberichte aus dem Fachbereich Informatik 7/2008

Matthias Bohnen, Lina Brühl, Sebastian Bzdak, RoboCup 2008 Mixed Reality League Team Description, Arbeitsberichte aus dem Fachbereich Informatik 6/2008

Bernhard Beckert, Reiner Hähnle, Tests and Proofs: Papers Presented at the Second International Conference, TAP 2008, Prato, Italy, April 2008, Arbeitsberichte aus dem Fachbereich Informatik 5/2008

Klaas Dellschaft, Steffen Staab, Unterstützung und Dokumentation kollaborativer Entwurfs- und Entscheidungsprozesse, Arbeitsberichte aus dem Fachbereich Informatik 4/2008

Rüdiger Grimm: IT-Sicherheitsmodelle, Arbeitsberichte aus dem Fachbereich Informatik 3/2008

Rüdiger Grimm, Helge Hundacker, Anastasia Meletiadou: Anwendungsbeispiele für Kryptographie, Arbeitsberichte aus dem Fachbereich Informatik 2/2008

Markus Maron, Kevin Read, Michael Schulze: CAMPUS NEWS – Artificial Intelligence Methods Combined for an Intelligent Information Network, Arbeitsberichte aus dem Fachbereich Informatik 1/2008

Lutz Priese,Frank Schmitt, Patrick Sturm, Haojun Wang: BMBF-Verbundprojekt 3D-RETISEG Abschlussbericht des Labors Bilderkennen der Universität Koblenz-Landau, Arbeitsberichte aus dem Fachbereich Informatik 26/2007

Stephan Philippi, Alexander Pinl: Proceedings 14. Workshop 20.-21. September 2007 Algorithmen und Werkzeuge für Petrinetze, Arbeitsberichte aus dem Fachbereich Informatik 25/2007

Ulrich Furbach, Markus Maron, Kevin Read: CAMPUS NEWS – an Intelligent Bluetooth-based Mobile Information Network, Arbeitsberichte aus dem Fachbereich Informatik 24/2007

Ulrich Furbach, Markus Maron, Kevin Read: CAMPUS NEWS - an Information Network for Pervasive Universities, Arbeitsberichte aus dem Fachbereich Informatik 23/2007

Lutz Priese: Finite Automata on Unranked and Unordered DAGs Extented Version, Arbeitsberichte aus dem Fachbereich Informatik 22/2007

Mario Schaarschmidt, Harald F.O. von Kortzfleisch: Modularität als alternative Technologie- und Innovationsstrategie, Arbeitsberichte aus dem Fachbereich Informatik 21/2007

Kurt Lautenbach, Alexander Pinl: Probability Propagation Nets, Arbeitsberichte aus dem Fachbereich Informatik 20/2007

Rüdiger Grimm, Farid Mehr, Anastasia Meletiadou, Daniel Pähler, Ilka Uerz: SOA-Security, Arbeitsberichte aus dem Fachbereich Informatik 19/2007

Christoph Wernhard: Tableaux Between Proving, Projection and Compilation, Arbeitsberichte aus dem Fachbereich Informatik 18/2007

Ulrich Furbach, Claudia Obermaier: Knowledge Compilation for Description Logics, Arbeitsberichte aus dem Fachbereich Informatik 17/2007

Fernando Silva Parreiras, Steffen Staab, Andreas Winter: TwoUse: Integrating UML Models and OWL Ontologies, Arbeitsberichte aus dem Fachbereich Informatik 16/2007

Rüdiger Grimm, Anastasia Meletiadou: Rollenbasierte Zugriffskontrolle (RBAC) im Gesundheitswesen, Arbeitsberichte aud dem Fachbereich Informatik 15/2007

Ulrich Furbach, Jan Murray, Falk Schmidsberger, Frieder Stolzenburg: Hybrid Multiagent Systems with Timed Synchronization-Specification and Model Checking, Arbeitsberichte aus dem Fachbereich Informatik 14/2007

Björn Pelzer, Christoph Wernhard: System Description:“E-KRHyper“, Arbeitsberichte aus dem Fachbereich Informatik, 13/2007

Ulrich Furbach, Peter Baumgartner, Björn Pelzer: Hyper Tableaux with Equality, Arbeitsberichte aus dem Fachbereich Informatik, 12/2007

Ulrich Furbach, Markus Maron, Kevin Read: Location based Informationsystems, Arbeitsberichte aus dem Fachbereich Informatik, 11/2007

Philipp Schaer, Marco Thum: State-of-the-Art: Interaktion in erweiterten Realitäten, Arbeitsberichte aus dem Fachbereich Informatik, 10/2007

Ulrich Furbach, Claudia Obermaier: Applications of Automated Reasoning, Arbeitsberichte aus dem Fachbereich Informatik, 9/2007

Jürgen Ebert, Kerstin Falkowski: A First Proposal for an Overall Structure of an Enhanced Reality Framework, Arbeitsberichte aus dem Fachbereich Informatik, 8/2007

Lutz Priese, Frank Schmitt, Paul Lemke: Automatische See-Through Kalibrierung, Arbeitsberichte aus dem Fachbereich Informatik, 7/2007

Rüdiger Grimm, Robert Krimmer, Nils Meißner, Kai Reinhard, Melanie Volkamer, Marcel Weinand, Jörg Helbach: Security Requirements for Non-political Internet Voting, Arbeitsberichte aus dem Fachbereich Informatik, 6/2007

Daniel Bildhauer, Volker Riediger, Hannes Schwarz, Sascha Strauß, „grUML – Eine UML-basierte Modellierungssprache für T-Graphen“, Arbeitsberichte aus dem Fachbereich Informatik, 5/2007

Richard Arndt, Steffen Staab, Raphaël Troncy, Lynda Hardman: Adding Formal Semantics to MPEG-7: Designing a Well Founded Multimedia Ontology for the Web, Arbeitsberichte aus dem Fachbereich Informatik, 4/2007

Simon Schenk, Steffen Staab: Networked RDF Graphs, Arbeitsberichte aus dem Fachbereich Informatik, 3/2007

Rüdiger Grimm, Helge Hundacker, Anastasia Meletiadou: Anwendungsbeispiele für Kryptographie, Arbeitsberichte aus dem Fachbereich Informatik, 2/2007

Anastasia Meletiadou, J. Felix Hampe: Begriffsbestimmung und erwartete Trends im IT-Risk-Management, Arbeitsberichte aus dem Fachbereich Informatik, 1/2007

„Gelbe Reihe“
(http://www.uni-koblenz.de/fb4/publikationen/gelbereihe)

Lutz Priese: Some Examples of Semi-rational and Non-semi-rational DAG Languages. Extended Version, Fachberichte Informatik 3-2006

Kurt Lautenbach, Stephan Philippi, and Alexander Pinl: Bayesian Networks and Petri Nets, Fachberichte Informatik 2-2006

Rainer Gimnich and Andreas Winter: Workshop Software-Reengineering und Services, Fachberichte Informatik 1-2006

Kurt Lautenbach and Alexander Pinl: Probability Propagation in Petri Nets, Fachberichte Informatik 16-2005

Rainer Gimnich, Uwe Kaiser, and Andreas Winter: 2. Workshop ''Reengineering Prozesse'' – Software Migration, Fachberichte Informatik 15-2005

Jan Murray, Frieder Stolzenburg, and Toshiaki Arai: Hybrid State Machines with Timed Synchronization for Multi-Robot System Specification, Fachberichte Informatik 14-2005

Reinhold Letz: FTP 2005 – Fifth International Workshop on First-Order Theorem Proving, Fachberichte Informatik 13-2005

Bernhard Beckert: TABLEAUX 2005 – Position Papers and Tutorial Descriptions, Fachberichte Informatik 12-2005

Dietrich Paulus and Detlev Droege: Mixed-reality as a challenge to image understanding and artificial intelligence, Fachberichte Informatik 11-2005

Jürgen Sauer: 19. Workshop Planen, Scheduling und Konfigurieren / Entwerfen, Fachberichte Informatik 10-2005

Pascal Hitzler, Carsten Lutz, and Gerd Stumme: Foundational Aspects of Ontologies, Fachberichte Informatik 9-2005

Joachim Baumeister and Dietmar Seipel: Knowledge Engineering and Software Engineering, Fachberichte Informatik 8-2005

Benno Stein and Sven Meier zu Eißen: Proceedings of the Second International Workshop on Text-Based Information Retrieval, Fachberichte Informatik 7-2005

Andreas Winter and Jürgen Ebert: Metamodel-driven Service Interoperability, Fachberichte Informatik 6-2005

Joschka Boedecker, Norbert Michael Mayer, Masaki Ogino, Rodrigo da Silva Guerra, Masaaki Kikuchi, and Minoru Asada: Getting closer: How Simulation and Humanoid League can benefit from each other, Fachberichte Informatik 5-2005

Torsten Gipp and Jürgen Ebert: Web Engineering does profit from a Functional Approach, Fachberichte Informatik 4-2005

Oliver Obst, Anita Maas, and Joschka Boedecker: HTN Planning for Flexible Coordination Of Multiagent Team Behavior, Fachberichte Informatik 3-2005

Andreas von Hessling, Thomas Kleemann, and Alex Sinner: Semantic User Profiles and their Applications in a Mobile Environment, Fachberichte Informatik 2-2005

Heni Ben Amor and Achim Rettinger: Intelligent Exploration for Genetic Algorithms – Using Self-Organizing Maps in Evolutionary Computation, Fachberichte Informatik 1-2005