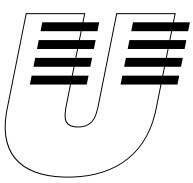


UNIVERSITÄT
KOBLENZ · LANDAU

Workshop Software-Reengineering und Services

Rainer Gimnich, Andreas Winter

1/2006

Fachberichte
INFORMATIK

ISSN 1860-4471

Universität Koblenz-Landau
Institut für Informatik, Universitätsstr. 1, D-56070 Koblenz

E-mail: researchreports@uni-koblenz.de,
WWW: http://www.uni-koblenz.de/FB4/

Vorwort

Der Workshop *Software Reengineering und Services* befasst sich mit der Nutzung von Software-Reengineering-Ansätzen für die Erstellung von Services und die Migration in Service-Orientierte Architekturen. Services sind wiederverwendbare Software-Ressourcen mit einer zugehörigen Beschreibung, über die Leistungen gesucht, gebunden und aufgerufen werden können. Eine Service-Orientierte Architektur (SOA) ist eine IT-Architektur, die die Struktur der Services gestaltet und die Unternehmensressourcen in einer flexiblen Weise (‚on demand') zusammenführt. Services werden vom Fachbereich als exponierbare Elemente eines Geschäftsprozesses verstanden. Aus Sicht der Softwareentwicklung gelten Services als funktionale Einheiten, die entweder neu entwickelt oder aus vorhandenen Software-Komponenten, Anwendungen oder Software-Paketen ‚herausgelöst' und realisiert werden.

Die drei ausgewählten Beiträge des Workshops skizzieren das *Software Reengineering für Services* auf eindrucksvolle Weise: Jens Borchers beschreibt ein industriell erprobtes Vorgehen zur Umgestaltung vorhandener Softwaresysteme und betont hierbei insbesondere Analysen, Bereinigungen und Umgestaltungen der Softwarearchitekturen. Jörg Ziemann, Katrina Leyking, Timo Kahl und Dirk Werth gehen von konzeptuellen Problemen bei der SOA-Migration aus und schlagen einen Ansatz aus dem Bereich der Unternehmungsmodellierung vor, um geschäftliche Anforderungen explizit in den SOA-Entwurf einzubringen. Harry Sneed schließlich baut auf seiner umfangreichen Reengineering-Erfahrung auf und beschreibt Werkzeugfunktionen, die die Transformation existierender Software in SOA unterstützen helfen.

Neben der Betrachtung des Reengineerings als Unterstützer auf dem Weg zu Service-Orientierten Architekturen ist auch die Anwendung der Ideen der Service-Orientierung auf das Software-Reengineering selbst zu betrachten. *Services für das Reengineering* untersuchen Services als Träger für Software-Reengineering-Funktionen. Harry Sneed zeigt in seinem Beitrag hierzu auch diverse „Reengineering-Services" auf, die die Migration in Service-Orientierte Architekturen unterstützen.

Alle Beiträge gehen davon aus, dass der beginnende Übergang zu SOA auf Unternehmensebene in Praxis nicht ohne sorgfältige Analyse und großflächige Umgestaltung existierender Anwendungssysteme realisierbar sein wird. Die Unternehmen wollen ihre Software-Investitionen und die Wettbewerbsvorteile ihrer Entwicklungen schützen und sich gleichzeitig in neue, flexible Architekturen bewegen, die zudem erhebliche Kostenvorteile für Weiterentwicklung, Betrieb und Wartung mit sich bringen.

Wir danken den Autoren für die professionelle Präsentation und Dokumentation ihrer Arbeiten und den deutlichen Praxisbezug und den Gutachtern im Programmkomitee für ihre kritische und konstruktive Bewertung. Unser besonderer Dank gilt Franz Lehner, der einen Reengineering-Workshop auf der Multikonferenz Wirtschaftsinformatik 2006 angeregt und ermöglicht hat. Die Praxis wird zeigen, wie sich das komplexe Gebiet *Software Reengineering und Services* aus seinen Kinderschuhen heraus bewegt und sich konzeptuell und technologisch weiter ausgestaltet.

Koblenz, im Februar 2006

Andreas Winter
Universität Koblenz
Institut für Softwaretechnik

Rainer Gimnich
IBM Software Group
Enterprise Integration

Programm

Jens Borchers
(Steria Mummert Consulting, Hamburg)
Umgestaltung bestehender komplexer Anwendungssystem
- vom „Heute“ zum „Morgen“ mit Services

Jörg Ziemann, Katrina Leyking, Timo Kahl, Dirk Werth
(Institute for Informations Systems (IWi), DFKI Saarbrücken)
Enterprise Model driven Migration from Legacy to SOA

Harry Sneed
(AneCon GmbH, Wien)
Reengineering von Legacy Programmen für die Wiederverwendung als Web Services

Jens Borchers

Umgestaltung bestehender komplexer Anwendungssysteme - vom „Heute“ zum „Morgen“ mit Services

Vortrag bei der MKWI06 in Passau am 21. Februar 2006

steria mummert consulting

Agenda

Das Unternehmen – das Wichtigste

Die Ausgangslage

Die Herausforderungen

Das Vorgehen in der Praxis

Erfahrungen und die weiteren Aussichten

steria mummert consulting

Agenda

Das Unternehmen

Die Ausgangslage

Die Herausforderungen

Das Vorgehen in der Praxis

Erfahrungen und die weiteren Aussichten

steria mummert consulting

Eine starke Allianz hat sich formiert

Das Unternehmen

steria

MUMMERT CONSULTING

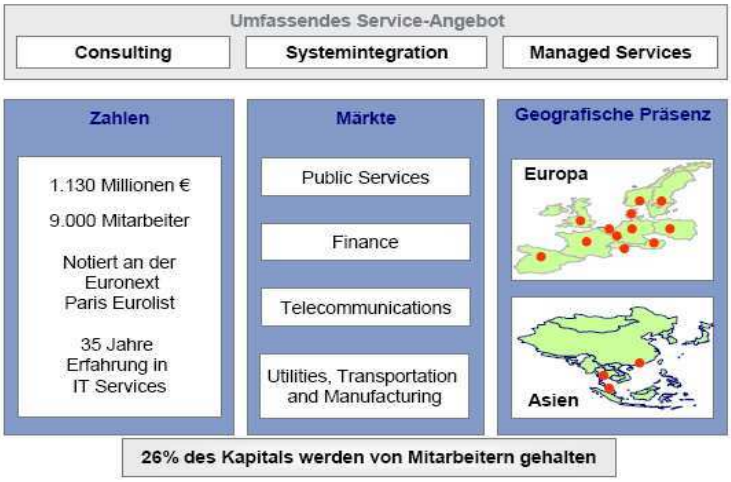
Seit dem 01.01.2005 ist die Mummert Consulting AG ein Unternehmen der Groupe Steria SCA.

Gemeinsam mit der Groupe Steria können wir nun unseren Kunden die komplette Wertschöpfungskette von der Beratung über die Systemintegration bis hin zu Managed Services anbieten.

steria mummert consulting

Dienstleistungsbranchen im Fokus

Groupe Steria gesamt

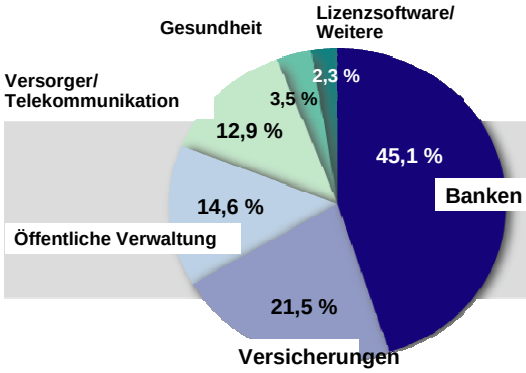


steria mummert consulting

Dienstleistungsbranchen im Fokus

Steria Mummert in D

Unternehmensberatung
für Finanz- und Dienstleistungsbranchen.



Gesamtleistung (IAS/IFRS) 2004: 146,202 Mio. Euro

Mitarbeiter 2004: 1.097

steria mummert consulting

Agenda

Das Unternehmen

Die Ausgangslage

Die Herausforderungen

Das Vorgehen in der Praxis

Erfahrungen und die weiteren Aussichten

steria mummert consulting

Die Ausgangslage

Die ursprünglichen Ziele

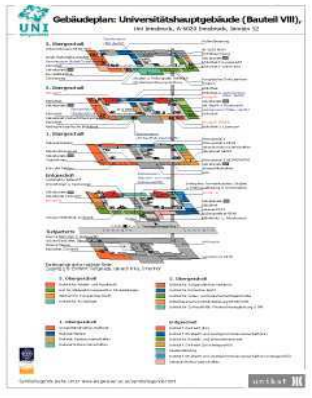
Eindeutige Funktionen.

Klare Strukturen.

Saubere Pläne.

Definierte Bauteile.

Kein Ballast.



(mit freundlicher Genehmigung von BAUHAUS / Uni Bremen-Stuttgart)

steria mummert consulting

Die Ausgangslage

Die bittere Realität

Grundstrukturen gerade noch erkennbar.

Einzelfunktionen schwer lokalisierbar.

Strukturverletzungen an vielen Stellen.

Abriß und Neubau oder Renovierung?

Wer bezahlt das?

Wie wird es nach ein paar Jahren aussehen?

(mit freundlicher Genehmigung von BAUHAUS / Uni Bremen-Stuttgart)

steria mummert consulting

Die Ausgangslage

Die Rahmenbedingungen

Eigentlich ist alles vorhanden.

In den großen Unternehmen sind heute praktisch alle wesentlichen Geschäftsfunktionen durch entsprechende IT-Systeme unterstützt.

Was ändert sich eigentlich über-haupt noch?

- Die äußeren (gesetzlichen) Rahmenbedingungen
- Die Prozesse, innerhalb der die Teilfunktionen zusammenwirken sollen.
- Die Benutzergruppen und ihre unterschiedlichen Bedürfnisse
- Die technischen Plattformen und Implementierungen.
- Die Organisation, die Software erstellt und wartet.

Belady-Lehmann´s Gesetz gilt immer.

Alle (erfolgreichen) Softwaresysteme müssen sich ständig anpassen.

steria mummert consulting

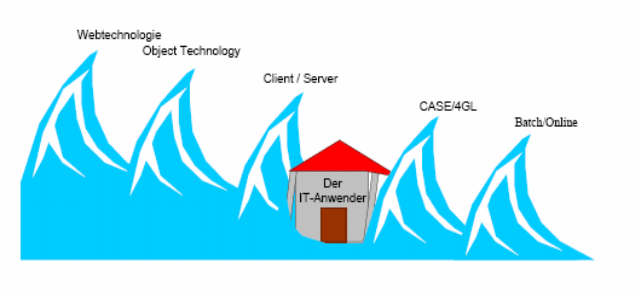
Die Ausgangslage

Der Wandel der Architekturen

Ca. alle 10 Jahre ein gravierender Neuansatz.

Technische Evolutionen sind parallel gelaufen:

- Betriebssysteme
- Programmiersprachen
- Datenbanken
- Transaktionsmanager
- Oberflächenstandards
- Kommunikationskanäle



The Waves of Change

(Harry Sneed: Objektorientierte Systemmigration, Addison-Wesley 1998)

steria mummert consulting

Agenda

Das Unternehmen

Die Ausgangslage

Die Herausforderungen

Das Vorgehen in der Praxis

Erfahrungen und die weiteren Aussichten

steria mummert consulting

Die Herausforderungen

Die Heterogenität der Systeme

Wo ist denn eigentlich „das Problem“?

Die Individualsysteme sind aufgrund ihres Alters (z. T. mehrere Jahrzehnte mittlerweile!) unter technischen Restriktionen entstanden, die heute i. d. R. nicht mehr gelten.

Nur die wenigsten Systeme sind jemals grund-legend von diesen ursprünglichen Restriktionen „befreit“ worden**.**

Es sind viele Teilsysteme in den unterschiedlichen Implementierungsansätzen „zurück“ geblieben.

Viele - auch neue (!) - Individualsysteme sind immer noch zu monolithisch. Viele Standardsoftware-systeme leider auch noch.

Lange bekannte Prinzipien werden noch immer nicht konsequent umgesetzt.

– „The problems of the 50s, 60s, and 70s are still with us.“ (David Parnas, 2001)

steria mummert consulting

Die Herausforderungen

Der Weg ist das Ziel

Was sind die Alter-nativen für die Systeme?

Die klassischen „Vier“ sind jetzt „Fünf“:

Wegwerfen, da ohnehin überflüssig.

Evolutionäre Weiterentwicklung.

Ganz neu entwickeln.

Durchgängiger Einsatz von „Standard“-Software.

Neu: „Komposition“ neuer Systeme aus Geschäfts-Einzelkomponenten:

- Neue individuelle Komponenten
- Funktionen aus Standard-Software
- Einbeziehung bestehender Systeme

steria mummert consulting

Die Herausforderungen

Der Weg ist das Ziel

Was sind die Alternativen für die weitere Systementwicklung?

Die üblichen Risiken:

Wenig beherrschte neue Technologie für strategische Systeme.

Die üblichen Risiken von Neuentwicklungen eingehen.

Die Rahmenbedingungen von Standard-Software akzeptieren.

Im Integrations-“Strudel“ untergehen

Ist/war EAI eine Lösung?

Werden Services die Welt einfacher machen?

Auf welchem Level werden wir Re-Use beherrschen können?

steria mummert consulting

Die Herausforderungen

Architekturwechsel gestern und heute

Die großen Architekturwechsel der Vergangenheit (und auch noch Gegenwart):

Betriebssystemwechsel

Systemnahe Trägersysteme

- Datenbankphilosophie und -system
- Transaktionsmanager / Application Server

Front-Ends und Benutzeroberflächen

Sprachumsetzungen

- 3GL nach 3GL, 3GL nach 4GL, 4GL nach 3GL
- OO nach OO

Hauptmerkmale:
Keine Änderung der fachlichen Funktionalität!
Mittlerweile gut beherrscht, kostengünstig und schnell durchführbar.

steria mummert consulting

Die Herausforderungen

Architekturwechsel heute und morgen

Die nächsten Architek-turwechsel (heute und morgen):

Wesentlich stärker durch Geschäftsanforderungen geprägt.

Technische Architektur nur noch als „Enabler“, nicht mehr als Treiber.

- Basissysteme sind z.T. „Commodity“ geworden.
- Weitere Reduktion der technischen Vielfalt bleibt wichtige Aufgabe.

Standardsoftware nimmt wesentliche Rolle ein.

Keine alleinige IT-interne „Veranstaltung“ mehr.

Hauptmerkmale:

Neuorganisation/-verteilung der fachlichen Funktionen!

Viele Beteiligte aus unterschiedlichen Bereichen.

Outsourcing und Kostenreduktion nur für Teilbereiche möglich.

(Noch) nicht voll beherrscht.

steria mummert consulting

Agenda

Unser Unternehmen

Die Ausgangslage

Die Herausforderungen

Das Vorgehen in der Praxis

Erfahrungen und die weiteren Aussichten

steria mummert consulting

Das Vorgehen

Die wesentlichen Phasen

Motivations- und Finanzierungsphase

- Business Case
- Sponsoren

Vorbereitungsphase

- Die grundsätzliche Umsetzungs-strategie.
- Fachliche und technische Architektur-konzepte.
- Inventarisierung und Bewertung der betroffenen Systeme.
- Implementierung der Zielsysteme.
- Schaffung von Testumgebungen für Quell- und Zielumgebung.
- Implementierung der Werkzeug-umgebung.
- Implementierungen von Spezial-mechanismen.

steria mummert consulting

Das Vorgehen

Die wesentlichen Phasen

Umsetzungsphase

Mögliche alternative Vorgehen:

- „Big Bang“ – alles fertigstellen, dann gemein-sam einsetzen
- „Schleichend“ – Anpassungen im normalen Wartungsbetrieb um- und einsetzen
- „paketorientiert“ („Chicken little“-Approach) – in geordneten Teilpaketen, die dann auch direkt in Produktion übernommen werden

Empfohlen: Rigide und forcierte Umsetzung der Konzepte in Projektform

Orthogonale Umsetzungsalternativen:

- Anwendungskomplexe als Leitfaden (vertikaler Ansatz)
- Softwareschichten als Leitfaden (horizontaler Ansatz)

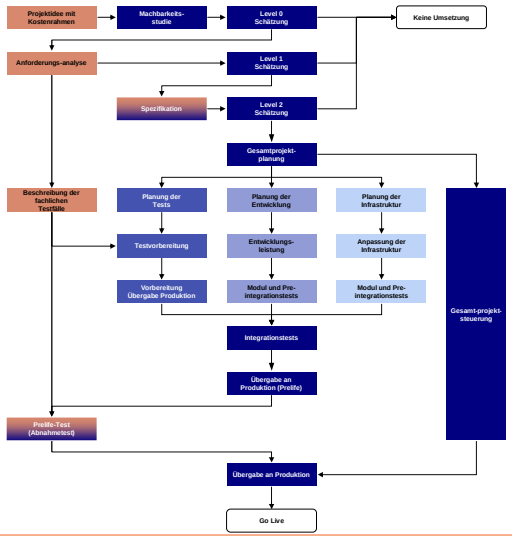
steria mummert consulting

Das Vorgehen

Eine übliche Projektorganisation

Die wesentlichen IT-Rollen in Architektur- Projekten sind:

- Anforderungsmanagement
- Architekturmanagement
- Software-Entwicklung und -Anpassung
- Daten-Konsolidierung und -Migration
- Konfigurationsmanagement
- Qualitäts- und Testmanagement
- Systemmanagement
- Projektmanagement



steria mummert consulting

Das Vorgehen

Die üblichen Hauptaufgaben

Unterschiede zu rein technischen Architek-turveränderungen

Aufwendigere Analysen notwendig
- Architektur-Explorer (BAUHAUS, GUPRO u.a.)
- Reverse Engineering (?)
- Cross Referencing, Clone Detection

Fachliche Bewertungen und Entscheidungen erforderlich
- Service-Identifikation
- Service-Zuschnitte
- Service-Verteilung auf Subsysteme

Nachweis der unveränderten Funktionalität - Regressionstests
- Nur noch bedingt auf rein technischer Basis (Capture/Replay)
- Aber mindestens so kritisch wie bei technischem Reengineering

steria mummert consulting

Das Vorgehen

Die üblichen Hauptaufgaben

Mittel- und langfristige Aufgaben in der IT-Organisation

- Architektur muß aktiv gemanagt werden.
  - Pflege und Konfigurationsmanagement der Architekturmodelle
  - Qualitätssicherung auf fachlicher und technischer Ebene
  - Rolle in Projekten
  - Standardsoftware muß einbezogen werden
- Auswirkungen auf die Entwicklungsorganisation
  - Pflege Service-Directory (UDDI?)
  - Herstellung neuer Services – Verantwortung und Kosten
  - Wartungsmanagement für Services – „unchartered waters“
- Und immer: Nachweis der Funktionalität
  - Testmanagement wird noch kritischer!

steria mummert consulting

Agenda

Unser Unternehmen

Die Ausgangslage

Die Herausforderungen

Das Vorgehen in der Praxis

Erfahrungen und die weiteren Aussichten

steria mummert consulting

Erfahrungen und Aussichten

Lessons learned (so far)

Die wesentlichen Erkenntnisse:

Die grundlegenden Konzepte für „ideale“ Architekturen sind altbekannt, aber lange nicht konsequent umgesetzt worden – passiert es jetzt?

Konsequente Architekturbereinigungen kosten Geld.

- Wer kann nachweisen, wie der ROI aussieht?
- Wer denkt heute noch langfristig genug?

Selbst bestimmen, wo es hingehen soll!

- Eigene Kontrolle der wichtigsten Architektur-blöcke.
- Weitgehende Unabhängigkeit muss Ziel Nr. 1 sein.
- Und: Nicht jedem Trend nachrennen, viele sind schon wieder „verglüht“!

Fazit: Architektur lässt sich bewegen!

- Wenn die Ziele und ihre Umsetzung klar sind.

steria mummert consulting

**Vielen Dank für Ihre Aufmerksamkeit,
Ihre Fragen sind willkommen.**

steria mummert consulting

Enterprise Model driven Migration from Legacy to SOA

Jörg Ziemann, Katrina Leyking, Timo Kahl, Dirk Werth

Institute for Informations Systems (IWi)
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, Geb. D3 2
66123 Saarbruecken
ziemann@iwi.uni-sb.de
leyking@ iwi.uni-sb.de
kahl@iwi.uni-sb.de
werth@iwi.uni-sb.de

Abstract: Still today many enterprises rely on inflexible, monolithic legacy systems inadequate for fast changing global markets. Service-oriented Architecture (SOA) represents the most recent approach aiming at flexible integration of heterogeneous application systems along business processes. But analyzing existing approaches to develop SOA based systems reveals a rather scattered state-of-the-art modus operandi full of conceptual gaps, most of them lacking business understanding. This paper proposes a business driven legacy-to-SOA migration approach based on enterprise modelling to leverage the potential of SOA while considering current enterprise requirements as well as existing (legacy) systems.

1 Motivation

Enterprises are faced with the frequent upcoming of new technologies and applications. However, they often do not follow the upgrade trends, but instead rely on their existing applications. In consequence, still today a lot of enterprises use inflexible, monolithic legacy systems for their business operations. These are applications using outdated technological concepts and programming languages. In most cases, detailed knowledge of the systems is lost and the substitution by a modern application is technologically not possible (due to missing knowledge) or not economically reasonable.
Service-oriented Architecture seems to represent the greatest paradigm shift since several decades. By definition SOA differ from previous IT architectures in their lose coupling of distributed components and their integration over open internet-based standards. Thus, SOA connects software through a pool of abstract services that provide a well defined, self-contained functionality of a software module. It enables the on-demand composition of enterprise software [Wo03] and therefore provides enterprises with a high degree of flexibility combined with an optimal support for their business processes [LRS02]. Thus, SOA has grown from a theoretical concept to one used in practice, cp. for example [Ac05];[Zi05]. SOA offers enterprises a wide range of advantages, however,

they can only profit if their applications are SOA-ready. This supposes that they expose Web Services. Legacy applications more than ten years old certainly don't.
Consequently, the migration of legacy systems into SOA-environments is a major research challenge with a significant economical impact. Despite major research efforts in the field of Legacy Reengineering and Service-orientation, current approaches only focus the technical migration itself and do not reflect business requirements. However, they incorporate the core in the migration since business-interests are the motivation of migration.
We propose the use of enterprise models to provide a solid base for migration. Given such a fundamental perception of enterprise modelling as umbrella discipline, this paper proposes a holistic approach that closes the gaps between Legacy Reengineering, Business Process Management and SOA Implementation. In order to enable a stepwise, business-driven construction of SOA, an overall methodology to migrate legacy systems to SOA is proposed to enable the migration. For this purpose, we analyze current migration approaches and reveal a set of gaps to the business requirements. Afterwards, we describe the steps to take into account for business requirements manifested in enterprise models. So, we align the technical migration procedure to enterprise needs.

2 Legacy to SOA migration – State of the art approaches

The concept of SOA, including the possibility to wrap existing programs and to publish them via a functional and less technical description suggests an easy way on how to turn legacy systems into a SOA obeying business requirements. But existing approaches are either vendor dependent (e.g. [Wo03]), focus solely on the business side (e.g. [SF02]) or rely too much on technique, disregarding business requirements (e.g. [ZKG04]). Experience from first SOA implementations suggest that existing development processes and notations, e.g. Object Oriented Analysis (OOAD) and Design, Enterprise Architecture (EA) Frameworks and Business Process Modelling (BPM) only cover part of a holistic SOA development [ZKG04]. Thus, enterprises still lack a vendor independent interdisciplinary approach for SOA development (cp. also [KM04]).

2.1 Current Approaches

Modifying a software product after delivery in order to improve its performance and adapt it to a changed environment, Software Reengineering keeps Software Systems effective and efficient and thus maintains them [IEEE83]. Legacy Reengineering specifically aims to resolve the issue of an insufficient information base for evolving the system due to lacking documentation. Therefore the legacy system must be analyzed using Reverse Engineering techniques, addressing two goals: (1) identifying the system's components and their interrelationships, and (2) creating representations of the system at a higher level of abstraction.
The existing Reengineering methods provide a good foundation to deliver the information necessary for building software architecture on top of the legacy system, which follows the process-oriented and Service-oriented paradigms. For some cases, e.g. identification of components to be published as Web Services, and to a lesser extent Web-

Service Orchestration and Choreography, existing methods can be extended for SOA-Reengineering. For other SOA Reengineering aspects, e.g. extraction of information necessary for Service Discovery or process-oriented aspects like service based transactions, no methods exist. Nevertheless, no reengineering methods exist that focuses on extraction of relevant information for SOA – especially not process-oriented SOA. Rather than being of a purely technical nature, the challenges in this area are related to business engineering: How can a sub-functionality be identified as a potential service, or how can business process models be derived for a legacy system?
To build flexible, reusable IT systems it is not sufficient to deploy Web Services, but a methodology is required to ensure that all principles of a SOA are met and the advantages it offers are realized. Although vendors already offer complex proprietary software systems to enact a SOA, there is a lack of holistic methods that systematically prepare a SOA implementation, taking into account requirements of existing (legacy) system and of current business needs as well. Almost all existing approaches implicitly use a bottom up approach, neglecting business needs. Focusing on the IT perspective yields brittle Web Services and case-by-case, experimental systems that fall short of implementing SOA principles [ZKG04]. This yields an information system that is made up of a myriad of heterogeneous data, applications and middleware that are neither flexible nor business focused.

2.2. Gap Analysis

Combining existing techniques and practices results in a three-step approach bringing legacy systems to service-orientation. As Fig. 1 illustrates major fractions remain causing three open questions.

Gap between legacy systems and Web Services: How to decompose?

The first question is how to decompose legacy systems into functional units which can be represented by Web Services. This problem is commonly tackled by reverse engineering methods. Hereby program analysis as well as tracing operational system behaviour may provide a starting point. But especially aspects of functional granularity, security, reliability, transactionality, etc. are not taken into account sufficiently. Likewise for the characteristics of business needs, methods have to be created to match characteristics of legacy systems to SOA systems. It has to be investigated how special characteristics of legacy systems (reliability, scalability, policies) can be matched to SOA systems.

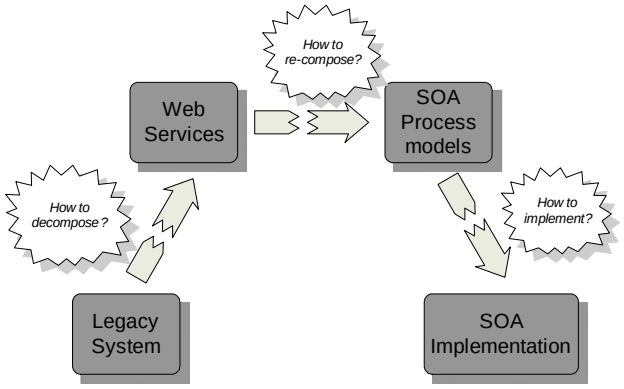


Figure 1: Conventional procedure for Legacy-to-SOA-migration

Gap between Web Services and SOA process standards: How to re-compose?

Once Web Services are defined, they are primarily stateless functions, able to execute single business activities [LR04]. In order to provide added value within an enterprise infrastructure, they must be embedded in an overall flow of control, a SOA based model. It specifies timely and logical conditions a service is subject to. A number of initiatives have been undertaken to define languages to describe service processes. Whereas consolidation of composition standards is looming with more and more vendors adapting BPEL4WS (also called WS-BPEL or BPEL, [An03]) as their composition language of choice, there are more serious gaps to overcome. The major gap persists in the clash between service composition and business needs, since WS-standards so far do not display practical enterprise needs [Ka03].

Gap between SOA standards and SOA executing infrastructure: How to implement?

SOA process models developed by simply translating business process logic are far from being executable. They rather offer raw frames to which various technical details must be added. For the development of SOA the realization of a complex set of supporting application infrastructure is required, including composition engines, service buses, application servers, and transformation engines. There is a heterogeneous range of products available on the market which addresses certain aspects of SOA but a methodology of how to compose different modules of a SOA infrastructure to obtain certain business needs is still missing. Altogether, none of the existing steps takes into account pending business requirements. None of the gaps described above are limited to missing technological techniques but most of them are mainly caused by a lack of business understanding. The field of enterprise modelling, discussed in the following section, promises to provide a bridge between soft business requirements and technical Legacy-2-SOA construction.

3. Migration based on enterprise models

Information constituting an enterprise range from its processes and behaviour, object and material flow to organizational units, personnel resources and system infrastructure. Through enterprise modelling (EM) that information is gathered, processed and displayed in a well-defined, explicit format in order to overcome the clash between business and IT perspective causing IT infrastructures that do not map to business requirements. Thus, EM especially contributes to migration difficulties by documenting business activities and related information to be supported by the target system. An enterprise model is defined as an abstract description of the enterprise system or a specific section of it. Various EM methodologies have brought about an impressive number of different languages (currently more than 300) addressing divers perspectives on the needs of enterprises.

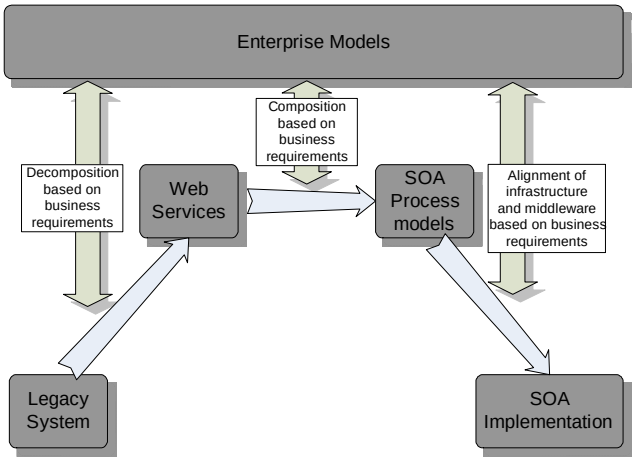


Figure 2: Business-oriented Legacy-to-SOA-migration

They all divide complex enterprise reality into a set of dimensions: functions, data, organizational structure and process control. The latest mostly represents the centre of focus as (business) processes integrate dynamically all other dimensions. For decades, Enterprise modelling has proven to be a viable approach to analyze organizations and to restructure their IT systems. As fig. 2 illustrates and the following paragraphs delineate, business requirements captured in enterprise models can complement a technical Legacy-to-SOA migration and thus provide for a more holistic, business-driven approach.

3.1. Enterprise models and Web Service engineering

Web Service engineering incorporates the question of how to slice an application system into functions (Web Services) that can be exposed externally. Thus, the problem is how to decompose the legacy application into Web Services in a way that

- they are coarse enough to encapsulate implementation details. If one chooses the service granularity too fine, there is no added value to a SOA. Instead, the SOA is used to 'reprogram' the application logic.
- they are fine enough to enable a flexible re-composition of services. If one defines the services too wide (in the worst case, one wraps the application by only one service), one cannot incorporate the intended business behaviour into the SOA because the modules that can be composed, are spanning too large parts of your business process.

The right granularity is determined by the technical requirements of the legacy system as well as business requirements. However, within this article, we assume the technical questions as secondarily. Thus, a major contribution of enterprise models for Web Service engineering is the determination of a functional abstraction degree suitable for a flexible business process composition. We propose a procedure to define an adequate granularity of Web Services of an existing legacy system based on a function tree model of the legacy systems (cf. fig. 2). This model represents a hierarchical structure of all functions used within the system. Each node represents a function and its sub-nodes represent sub-functions. From this tree, the task is to select the right functions in the right granularity that has to be converted (i.e. wrapped) into Web Services.

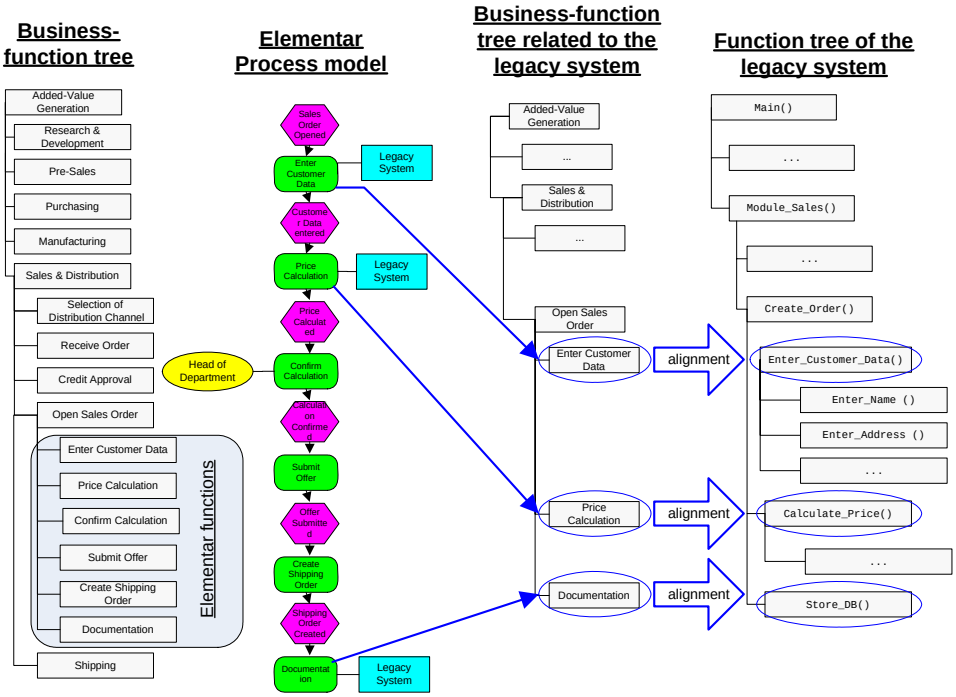


Figure 3: Enterprise modelling based Web Service definition

To do so, enterprise models seem to offer valuable possibilities. Most of the commonly used Enterprise Architectures (such as ARIS [Sc99], Zachman [Za87], etc.) include a functional description of the enterprise. This can also be expressed within a hierarchical tree, resulting in a business-oriented function tree model. This model is similar to the

legacy function tree, but comprehends business functions (such as ‘invoice material’) instead of system ones (e.g. ‘create bill entry in database’). Thus, the challenge is to detect those functions within the business function tree which are on the one hand related to the legacy system and on the other hand of relevance for the business process management task.

This approach suggests the selection of the relevant functions on the basis of process models. In an enterprise, processes are modelled in different granularities. In order to define the Web Service in the right granularity the most elementary process models are relevant. These models are detailed enough to define significant services. In a next step it is necessary to identify these functions which are associated to the legacy system.

The functions which are performed or supported by the targeted legacy system will be marked by links to an application system or type of application system which represents the legacy system in the process models. Only these functions will be decisive for the service definition whereas all other functions are obsolete. On the basis of the process models the overall function tree will be reduced to a diagram which exclusively contains functions which are related to the targeted legacy system. Such a hierarchical diagram is the basis for the determination of the optimal Web Service granularity from a business point of view.

Having identified these business-tree nodes, the function tree of the legacy system needs an investigation to reveal functions that incorporate these business functionalities. The functions that map the marked business functions are those that should be transformed into Web Services, because

- they are abstract (coarse) enough to hide sufficiently implementation details. In fact, in choosing finer services (i.e. select sub-functions of this function F), the application logic of F within the service orchestration language would have to reconstructed, but you would only use F as a whole therein.
- they are detailed (fine) enough to allow the definition of the intended business process. Rather, having more abstract service definitions (i.e. a super-node of function F), F can be used exclusively in the way, that it is used in the implementation of the super-node. This may compromise the business process definition and result in orchestration models that are not compatible to the intended business processes.

The final step is the technical wrapping of the functions identified by Web Services. This comprises the definition of WSDL data that specifies the service interfaces and access data as well as the creation of the SOAP-port itself.

3.2. Enterprise models and service process composition

The previous section described procedures to create single Web Services based on current business requirements and existing IT systems. To integrate the Web Services in process models that describe logical and timely dependencies between them, we presume the existence of conceptual business process models (e.g. EPCs) that incorporate also process knowledge extracted from legacy systems. Future research has to investigate how existing Reengineering methods have to be enhanced to extract all SOA relevant information. Two different types of SOA based process descriptions can be distinguished: Orchestration, used to display executable processes, and Choreography, used to display abstract processes [An03]. Within a company, usually executable business processes are used while service choreography models display peer-to-peer interactions and thus are more likely to be used in cross organizational processes [Ka04].
Orchestration of Web Services must leverage all essential information of this business process model into the SOA Process model to ensure business concordance. For this purpose only the business process models of highest level of detail within a process hierarchy are regarded. Transforming business process models into SOA process models requires a change of language from conceptual notations to executable – mostly XML-based – notations as illustrated by Fig. 3. E.g. an EPC process model could be transformed to BPEL orchestration (cp. also [ZM05]).

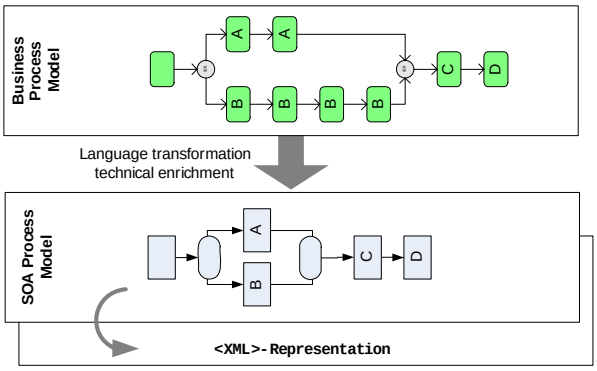


Figure 3: Model-based process transformation

To create the SOA process model, the first step is to map business functions of the underlying business process on to Web Services. The relationship between functions and services established previously as described in 3.2 provides the necessary information which services cover which business functions. Based on these mapping relations functions or subsets of them (e.g. the two *functions* A in the upper part of fig. 3) can be substituted by a service call (e.g. the *service* A). Having performed this substitution for the whole process model the second step is to consistently translate the process's logics from the business process notation to the SOA process notation, i.e. mapping logical structures of both concepts. First research approaches have already been undertaken e.g. in the field of transforming EPC to BPEL processes.

Choreography descriptions can be seen as interface definitions representing not only static but dynamic information: they describe the sequence in which functions of a service have to be executed and thus increase the reusability of business components. In order to derive Choreography descriptions from conceptual process models two ways are feasible: First, the model of an executable business process can be transformed to an executable Service Orchestration model, e.g. a common EPC could be transformed to an executable BPEL process. Consecutively, from this orchestration model a choreography model could be derived displaying the interactions between the composed process and its partners. The second way starts from a conceptual model of the choreography. This could be by conceptual view processes (e.g. [LG05], [SKK04]) which only show that part of an executable process that describes interactions with a specified partner. This view process then can be better transformed, the more technical Web Service Choreography descriptions there are.

4. Summary and future research

In this paper we motivated and described a business driven legacy-to-SOA migration approach based on enterprise modelling considering enterprise requirements as well as existing systems. We described how knowledge stored in enterprise models can be used to first develop single Web Services and second to integrate these services in SOA process models.
We focused on gaps situated on the conceptional level; a future framework for business driven SOA development also must tackle the gap between requirements of conceptual models and their technical implementation. Though investigated by various research efforts - most prominently the Model Driven Architecture (MDA) - challenges in closing this gap still exist for SOA implementation. For example, practitioners complain that after migration of legacy to SOA systems the latter lack certain characteristics of the old systems, e.g. performance and reliability.
Usually the Enterprise Service Bus (ESB) is used for SOA implementation, thus below conceptual business processes and SOA processes two further layers exist: the ESB layer and the IT infrastructure layer [Pa05]. Future research has to establish links between characteristics of enterprise models, SOA process models and their realization in the ESB. This comprises the direction of development, e.g. to ensure that transactionality and security requirements specified in enterprise models are realized in the ESB, as well as the direction of monitoring and controlling [CK05]. Apart from business process monitoring, a holistic methodology to determine the grade of synchronization between enterprise models and their underlying SOA implementation is necessary, also.

References

[Ac05] M. Acharya, A. Kulkarni, R. Kuppili, R. Mani, N. More, S. Narayanan, P. Patel, K. W. Schuelke, and S. N. Subramanian, "*SOA in the Real World - Experiences*" *ICSOC 2005*, pp. 437-449, 2005.

[An03] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana, *"Business Process Execution Language for Web Services- Version 1.1*", 2003. Http://dev2dev.bea.com/techtrack/BPEL4WS.jsp.

[CK05] D. E. Cox and H. Kreger, "*Management of the service-oriented-architecture life cycle*", In: IBM Systems Journal, vol. 44, no. 4, pp. 709-726, 2005.

[IEEE83] ANSI/IEEE Standard 729-1983, *IEEE Standard Glossary of Software Engineering Terminology*, IEEE, 1983.

[Ka03] D. Kaye, *"Loosely Coupled. The Missing Piece of Web Services"*, RDS Press, Marin County, 2003.

[Ka04] N. Kavantzas, D. Burdett, and G. Ritzinger, "*Web Services Choreography Description Language Version 1.0* - W3C Working Draft 27 April 2004", 2004. Http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427.

[KM04] I. H. Kruger and R. Mathew, *Systematic development and exploration of service-oriented software architectures. In Fourth Working IEEE/IFIP Conference on Software Architecture*, Oslo, 2004.

[LG05] S. Lippe and U. Greiner, *"A Survey on State of the Art to Facilitate Modelling of Cross-Organisational Business Processes"*, 2005.

[LRS02] F. Leymann, D. Roller, and M.-T. Schmidt, "*Web Services and business process management*", In: IBM Systems Journal, vol. 41, no. 2, pp. 198-211, 2002.

[LR04] F. Leymann and D. Roller. *"Modeling Business Processes with BPEL4WS*", 7th GI Conference "Modellierung 2004", 1. GI-Workshop XML4BPM 2004, XML Interchange Formats for Business Process Management. P-45, pp.7-24, Köllen Verlag, Bonn, 2004. Lecture Notes in Informatics (LNI).

[Pa05] J. Pasley, "*How BPEL and SOA are changing Web Services development*", In: Ieee Internet Computing, vol. 9, no. 3, pp. 60-67, May2005.

[Sc99] A.-W. Scheer, *"ARIS - business process modelling"*, 2., completely rev. and enl. ed., Springer, Berlin [et al], 1999.

[SF02] H. Smith and P. Fingar, *"Business Process Management: The Third Wave"*, Meghan-Kiffer, Florida, 2002.

[SKK04] A.-W. Scheer, R. Klein, and F. Kupsch, *"Modellierung inter-organisationaler Prozesse mit Ereignisgesteuerten Prozessketten"*, 178, Saarbrücken, 2004.

[Wo03] D. Woods, "*Enterprise Services Architecture"*, O'Reilly & Associates, Sebastopol, 2003.

[Za87] J. A. Zachman, "*A Framework for Information-Systems Architecture*", In: Ibm Systems Journal, vol. 26, no. 3, pp. 276-292, 1987.

[Zi05] O. Zimmermann, V. Doubrovski, J. Grundler, and K. Hogg, "*Service-oriented architecture and business process choreography in an order management scenario: rationale, concepts, lessons learned*" Companion To the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (San Diego, CA, USA, October 16 - 20, 2005), pp. 301-312, ACM Press, New York, 2005.

[ZKG04] O. Zimmermann, O. P. Krogdahl, and C. Gee, "*Elements of Service-Oriented analysis and Design: an interdisciplinary modeling approach for SOA projects"*. 2004, IBM - whitepaper", 2006. http://www-128.ibm.com/developerworks/library/ws-soad1/.

[ZM05] J. Ziemann and J. Mendling, "*Transformation of EPCs to BPEL – A pragmatic approach*" 7th International Conference on the Modern Information Technology in the Innovation Processes of the industrial enterprises, Genoa, 2005.

Reengineering von Legacy Programmen für die Wiederverwendung als Web Services

Harry M. Sneed
AneCon GmbH, Wien

Abstrakt: Das Ziel einer Service-orientierten Architektur ist die Bereitstellung vorgefertigter Funktionalität, die Anwender nach Bedarf abrufen kann. Ein Teil dieser Funktionalität wird von kommerziellen Softwareherstellern geliefert, ein Teil wird aus dem Open-Source Bereich bezogen, ein Teil wird vom Anwender zu diesem Zweck neu entwickelt und ein Teil wird aus der bestehenden Softwaremasse entnommen. Es ist noch schwer zu prognostizieren wie diese Anteile aufgeteilt werden, aber es steht bereits fest, daß ein großer Teil der angebotenen Web Services aus den vorhandenen Systemen stammen wird. Dieser Beitrag beschreibt wie Legacy Programme für diese Verwendung durch mehrere Reengineering Schritte, einschließlich Reinigung, Restrukturierung, Remodularisierung und Refaktorierung aufzubereiten sind. Ein Tool – SoftRedo – wird eingesetzt, um diese Schritte weitestgehend zu automatisieren.

Keywords: Web Services, SOA, Legacy Programme, Reengineering, Reuse, Recycling.

Web Services sind verpackte Software Funktionalitäten, auf die über das Internet zugegriffen werden kann. Der Zugriff erfolgt über eine Web Service Schnittstelle, die mit der Web Service Definition Language – WSDL – spezifiziert ist [1]. Darin werden die auszuführenden Funktionen zusammen mit ihren Argumenten, bzw. Eingangsparameter, ihren Ergebnissen, bzw. Ausgangsparameter, und ihren Ausnahmebedingungen beschrieben. Web Services reichen von trivialen Berechnungen wie Kalenderfunktionen bis hin zu umfassenden betriebswirtschaftlichen Algorithmen wie Preiskalkulationen. Die Granularität der Web Services ist eine wichtige Frage beim Entwurf einer Service-orientierten Architektur. [2]
Die Antwort auf jene Frage ist ausschlaggebend dafür, woher die Software zur Implementierung der Web Services bezogen wird. Einfache, wohl definierte Einzelfunktionen mit wenigen Parameter und einem singulären Ergebnis werden leichter zu kaufen, bzw. zu beschaffen sein. Komplexe, betriebsspezifische Funktionen mit zahlreichen Parameter und multiplen Ergebnissen müssen hingegen entweder neu entwickelt oder aus bestehenden Programmen übernommen werden.

In Anbetracht der Zeit und Kosten, die mit der Entwicklung und, vor allem, mit dem Test neuer Algorithmen verbunden sind, ist es eher wahrscheinlich, daß die komplexen, betriebsspezifischen Web Services aus dem vorhandenen Programmvorrat entnommen werden. Denn auch wenn der Code alt ist, sind die darin enthaltenen Algorithmen ausgereift. Meistens sind diese das Resultat einer langjährigen Evolution in der vieles versucht und oft nachgebessert wurde bis endlich eine akzeptable Lösung zustande kam. Es wäre daher töricht und unwirtschaftlich diesen langen Lernprozeß nur wegen der Umstellung der Softwarearchitektur zu wiederholen. Es liege nahe, die großen Investitionen der Vergangenheit in die neue Architektur hinüberzuretten. Jede Kosten/Nutzen Analyse belegt, daß dieses Alternative günstiger ist. [3]

1. Alternative Ansätze zur Wiederverwendung der alten Programme

Die Implementierung neuer Web Services durch die Wiederverwendung des alten Codes ist zwar wesentlich billiger als Web Services von Grund auf zu entwickeln, aber auch dieses Alternative verlangt seinen Preis. Alte Programme sind in der Regel nicht so konstruiert, daß sie auf Anhieb wiederverwendet werden können. Sie sind eng mit ihren Daten und ihrer Umgebung verflochten. Um sie wiederverwenden zu können, müssen diese engen Verflechtungen gekappt werden und der alte Code aus der Abhängigkeit zur alten Umgebung befreit werden. [4] Solche Reengineering Maßnahmen und die Kapselung des Codes sind die Hauptkostentreiber der Wiederverwendung. Sie sind zugleich die zwei wesentlichen Schritte in der Migration geschlossener umgebungsabhängiger Programme in offene allgemein verwendbare Web Services. Der erste Schritt ist das Reengineering, der zweite Schritt die Kapselung und der dritte Schritt die Integration gekapselter Komponenten in eine Webarchitektur.
In dem vorliegenden Beitrag geht es um den ersten Schritt – die Lösung der potentiellen Web Services aus ihrer alten Umgebung. Dazu bieten sich drei alternative Ansätze an:

1.) das Reverse Engineering,
2.) die Emulation der alten Schnittstelle, und
3.) die Transformation des alten Sourcecodes.

1.1 Reverse Engineering alter Programme

Das *Reverse Engineering* sieht vor, daß der Algorithmus von dem Code abgetrennt wird, in dem er implementiert ist. Durch eine Analyse des Codes wird die darin enthaltene Logik modelliert und in Form von Diagrammen oder in einer übergeordneten Entwurfssprache dargestellt. Die Diagramme, z.B. Aktivitätsdiagramme, Sequenzdiagramme oder Zustandsdiagramme, beschreiben auf einer höheren Abstraktionsebene was in dem Code geschieht und können als Grundlage für die Reimplementierung des Codes in einer anderen Sprache dienen.[5] Der Vorteil

dieses Ansatzes ist die Wiederverwendung der alten Lösung ohne den alten Code zu nutzen. Der Nachteil ist die Notwendigkeit die alte Lösung neu zu kodieren und zu testen. Es kann lange dauern bis der neue Code die gleiche Reife erlangt wie der Alte.

1.2 Emulation der alten Schnittstellen

Die Runtime Schnittstelle steht als Alternative dem Reverse Engineering gegenüber. Hier bleibt der Code unangetastet. Der Source Code wird nicht einmal betrachtet. Es sind die Datenschnittstellen der alten Programme, die Eingangs- und Ausgangsdatenflüsse, die analysiert und aufgebohrt werden, um so den Anschluß zur Laufzeit zu finden. Die Integration findet auf der binären Ebene statt. Die Web Services werden als Proxys, bzw. als Stellvertreterkomponenten entwickelt, die Web Service Requests in Datenströme umsetzen und an das Ziel-Programm umleitet. Nachdem das Programm ein Ergebnis produziert hat wird dieses abgefangen und in einzelne Daten zerlegt. Hieraus wird ein Web Service Response generiert.
Normalerweise dreht es sich bei den alten Schnittstellen um einen Datenstrom zu einem TP-Monitor wie CICS, IMS oder Tuxedo. In der alten Umgebung werden die Dialogprogramme mit Daten vom Bildschirm über eine binäre Schnittstelle gespeist. Die Batchprogramme holen ihre Daten über eine Datei- oder Datenbankschnittstelle. Beide Schnittstellen werden in der neuen Umgebung emuliert. Der Vorteil dieses Ansatzes ist, dass die alten Programme in ihrer alten Umgebung bleiben dürfen. Man spart die Kosten und Risiken einer Migration. Der Nachteil ist, daß Web Services, die auf dieser Objektcode-Kapselung basieren ineffizient und unflexibel sind. Der Anwender wird gezwungen immer mehr Code um sie herumzubauen. Die Kluft zwischen den Anforderungen der Web Services und der Leistung des alten Codes wird immer breiter ohne die Möglichkeit den alten Code verändern zu können. [6]

1.3 Transformation des alten Sourcecodes

Diese Reengineering Alternative liegt zwischen den beiden anderen Alternativen. Hierbei wird mit dem alten Sourcecode weitergearbeitet. Er wird in eine Form versetzt, die es erlaubt den Code in die neue Umgebung zu übertragen und ihn dort neu zu compilieren und als Web Service einzubinden. Vorher muß der Code allerdings transformiert werde, damit er wiederverwendbar wird. Das kann auch Kosten verursachen, vor allem wenn dieses manuell erfolgen muß. Der große Vorteil dieses Ansatzes ist, dass die Integration auf Sourceebene vollzogen wird. Damit hat der Anwender mehr Möglichkeiten den Code zu beeinflussen. Der Hauptnachteil ist die Notwendigkeit sich mit dem alten Code zu befassen. Wer nicht dazu in der Lage ist, soll es lieber lassen. [7]

2. Voraussetzungen für die Wiederverwendung von Legacy Code

Ursprünglich war das Ziel des Reengineering die Verbesserung der Codequalität im Hinblick auf die Wartung und Weiterentwicklung. Die dazu erforderlichen Maßnahmen zielten auf eine Steigerung der Modularität, Lesbarkeit und Änderbarkeit des Codes. Im Falle von Web Services steht die *Wiederverwendbarkeit* im Vordergrund. Es geht in erster Linie um die Befreiung des Codes aus seinen Abhängigkeiten. Davon gibt es mindestens vier Formen:

1.) **Datenabhängigkeit:** Der Code ist von den Daten abhängig, die im Code fest verdrahtet sind.
2.) **Umgebungsabhängigkeit:** Der Code ist von der technischen Umgebung (insb. von der Ein- und Ausgabe von Daten und dem Zugriff auf Dateien und Datenbanken) abhängig.
3.) **Runtime-Abhängigkeit***:* Der Code ist oft von einem Runtime-System abhängig, das den Arbeitsspeicher verwaltet, die Verbindungen zwischen getrennt compilierten Modulen herstellt und eingreift wenn etwas schief läuft, e.g. exception handling.
4.) **Aufrufabhängigkeit:** Code-Komponenten sind von anderen Code-Komponenten im gleichen Anwendungssystem abhängig. Sie werden aufgerufen, um Dienste zu leisten oder Daten zu liefern, auf die diese Komponente angewiesen ist.

Damit ein Codebaustein – ein Modul, eine Prozedur, oder eine Klasse – als Web Service wiederverwendet werden kann, gilt es ihn aus jenen Verflechtungen zu lösen und in einen unabhängigen, mehrfach verwendbaren Baustein zu verwandeln.

Hierzu muss der Sourcecode durch mehrere Transformationen angepasst werden. Zum einen, müssen die fest verdrahteten Daten aus den prozeduralen Anweisungen entfernt und als Parameter neu definiert werden. Zum zweiten, müssen die IO-Operationen und Datenbankzugriffe aus der Verarbeitungslogik ausgelöst und in eine separate Zugriffsschale versetzt werden. Zum dritten, müssen die Verbindungen zum Runtime-System gekappt und in eigene, eingebaute Funktionen innerhalb der Komponente verwandelt werden. Zum vierten, müssen die aufgerufenen Funktionen und Methoden in fremden Komponenten, von den Komponenten übernommen und in den aufrufenden Baustein eingebaut werden. In der OO-Testtechnologie wird dies als Class Flattening bezeichnet. [8]

Alle vier Maßnahmen dienen dazu, einen Codebaustein, der in eine andere Umgebung verpflanzt werden sollte, aus seiner Umgebung herauszulösen und so zu konfigurieren, dass er in seiner Zielumgebung, einer Service-orientierten Architektur, weiter funktionsfähig ist. Diese Aufgabe ist keineswegs trivial, aber mit Hilfe automatisierter Werkzeuge zu bewältigen. Wichtig ist, wie bei der menschlichen Organ Transplantationen, daß der Codebaustein die Transplantation heil übersteht, damit er seine Funktion in der Zielumgebung aufnehmen kann.

3. Transformationen durch das Werkzeug SoftRedo

Das Werkzeug SoftRedo wurde konzipiert und implementiert, um die oben genannten Voraussetzungen zu erfüllen. Darüber hinaus erfüllt es zwei weitere Ziele:

- Restrukturierung und
- Refaktorierung

Diese beiden Funktionen dienen der erleichterten Pflege und Fortschreibung von Softwarekomponenten. Insofern nutzen sie der Wiederverwendung als Web Service nur indirekt.

SoftRedo unterstützt die Transformation von Assembler, PL/I COBOL OO-COBOL, C und C++ Programme durch sieben Reengineeringschritte:

1. Reformat,
2. Reassign,
3. Relocate,
4. Redirect,
5. Remodularize,
6. Restructure,
7. Refactor.

3.1 Reformating

Im ersten Transformationsschritt wird der Code entsprechend seiner Verschachtelungsstufe eingerückt und neu formatiert. Außerdem werden multiple Anweisungen in einer Zeile auseinander genommen und auf mehrere Zeilen verteilt, so daß es nur eine Anweisung pro Zeile gibt. Dies erleichtert nicht nur die Lesbarkeit, sie erleichtert auch die weitere Verarbeitung des Codes.

3.2 Reassignment

Im zweiten Transformationsschritt werden alle Literale und numerischen Konstanten, die in prozeduralen Anweisungen vorkommen, durch symbolische Konstanten ersetzen. An Stelle des Texts, bzw. der Zahl, wird ein Variabelname eingesetzt. Je nach Sprache wird die Variabel als EQU, define, 88 Feld oder Konstante in einer Include oder Copy Member vereinbart, wo sie von außen verändert werden kann, ohne in den prozeduralen Code eingreifen zu müssen.

3.3 Relocation

Im dritten Transformationsschritt werden alle IO-Operationen und Datenbankzugriffe erkannt und durch Funktionsaufrufe ersetzt. Die IO-Operationen kommen in eine IO-Schale, die Datenbankzugriffe in eine Datenzugriffsschicht. Damit sind sie aus dem Mainline Code entfernt und können, je nachdem in welcher Umgebung der Code läuft, angepaßt werden ohne den Code zu beeinflussen. Falls der Code als Web Service wieder verwendet werden sollte, wird die IO-Schale durch eine Kap-

selungsschicht abgelöst. Dies wäre nicht möglich, solange die IO-Operationen in der Verarbeitungslogik eingebettet sind. Ohne die Umsetzung der vielen Read, Write, Get, Put, Send, Receive-Anweisungen wird es nie gelingen, ein bestehendes Programm als Web Service zu verwenden. Dann blieben nur die zwei Alternativen – Reimplementierung oder Emulierung.

3.4 Redirection

Im vierten Transformationsschritt geht es um das Abfangen der Runtime-System Aufrufe. Falls der Anwender den Code in eine andere Umgebung verpflanzen will, eine Umgebung in der diese Ausnahmedienste eventuell gar nicht existieren oder wenn dann mit einer völlig anderen Schnittstelle, müssen die Verbindungen zum Runtime System umgeleitet werden. Typische Beispiele sind SVC-Operationen in Assembler, ON Bedingungen in PL/I, EOF Behandlung in COBOL und die Ausnahmebehandlung in C++. In CICS Programme sind es die Handle Aide Bedingungen. [9]
Solche Interaktionen mit dem darunterliegenden Betriebs- oder Runtime-System, die eine Unterbrechung in der Programmausführung verursachen, müssen von lokalen Funktionen im Code der betroffenen Komponente behandelt werden. Im Rahmen der Web Services sollten derartigen Unterbrechungen zu einer Fehlermeldung an den Clienten führen. Dort müssen sie von Wrappern abgefangen und in einen abnormalen Response umgewandelt werden. Deshalb kommt es darauf an, sie schon vorher umzuleiten. Sonst besteht die Gefahr, daß auf einen Web Service Request allenfalls eine Timeout-Fehlermeldung folgt.

3.5 Remodularization

Im fünften Transformationsschritt werden die gegenseitigen Abhängigkeiten abgebaut. In jeder Sprache sind die einzelnen Module aufeinander angewiesen. In OO-Sprachen ist diese Verteilung der Funktionalität auf viele Funktionsträger besonders ausgeprägt. Ein Modul ruft das andere auf, um ihm einen Dienst zu leisten oder ihm eine Information zukommen zu lassen. Das aufgerufene Modul ruft wiederum ein anderes Modul auf usw. So entsteht eine Kette von Abhängigkeiten. Sollte ein Knoten aus dieser Kette gegenseitiger Abhängigkeiten herausgelöst werden, um als Web Service wieder verwendet zu werden, muß die Kette durchbrochen werden und zwar am besten dort wo sie beginnt. Jeder Aufruf einer fremden Funktion muß markiert und untersucht werden. Insofern als die fremde Funktion, bzw. Methode, nicht so groß ist, kann sie im aufrufenden Modul dupliziert werden. Am Ende des Zielmoduls wird es demnach eine Reihe zusätzlicher interner Funktionen samt ihrer Daten geben, denn die fremden Daten müssen auch übernommen werden. Somit wird hier die Verteilung der Funktionalität und Daten wieder rückgängig gemacht. Die Komponente, die für einen Web Service vorgesehen ist, sollte möglichst all inklusive sein. In C++ heißt bedeutet dies, dass auch die geerbten Attribute und Methoden eingebaut werden.

Sollten es sich jedoch herausstellen, daß die fremden Funktionen zu groß oder zu komplex sind, steht man vor der Entscheidung sie entweder als untergeordnete Web Services einzusetzen oder auf die Dienstleistung zu verzichten und die Verarbeitungslogik des Zielmoduls dementsprechend umzuschreiben. Falls die Abhängigkeiten zu zahlreich sind, was an der Fan-Out Metrik erkennbar wird, ist die Komponente als Web Service nicht geeignet. [10]
Aus diesem Grunde sind COBOL und PL/I Programme besser geeignet für Web Services als Assembler oder C++ Programme. So paradox es klingt, je modularer ein System ist, desto schwieriger ist es, einzelne Bausteine zur Wiederverwendung in einer anderen Umgebung herauszulösen.

3.6 Restructuring

Im sechsten Transformationsschritt werden die externen GO TO Verzweigungen entfernt. Dies ist in PL/I und C++ weniger ein Problem, wohl aber in COBOL und Assembler. Spagetti-Code, der voller GO TO Verzweigungen steckt, ist bekanntlich schwer zu verstehen und zu verändern. Dies hat aber wenig Auswirkung auf die eigentliche Wiederverwendbarkeit in einer anderen Umgebung, solange die GO TOs sich innerhalb der zu übernehmenden Komponente bewegen. Kritisch wird es wenn sie über CSECT, Section, Procedure oder Funktion Grenzen hinausgehen. Solche „bösen“ GOTOs müssen auf jeden Fall gekappt werden. *SoftRedo* erkennt solche Verzweigungen und fängt sie mit einem Platzhalter ab. Das Tool kann auch lokale GOTO Anweisungen entfernen, aber nur wenn der Anwender es wünscht, denn dies führt zu einer starken Veränderung der Ablaufstruktur. [11]

3.7 Refactoring

In dem siebten und letzten Schritt kann der ausgewählte Codebaustein noch durch *SoftRedo* refaktoriert werden. Es soll aber hier betont werden, daß diese Maßnahme keinen Einfluß auf die Wiederverwendung eines Codebausteins als Web Service hat. Aus der Sicht des Web Service Benutzers ist es unwesentlich, wie groß und wie komplex die einzelnen Prozeduren oder Methoden der Web Services sind. Es hat auch keine Auswirkung auf die Verpflanzung des Codes. Hauptsache ist, es ist möglich das Modul oder die Klasse aus seiner Umgebung herauszulösen und in eine neue zu versetzen. Dennoch, kann *SoftRedo* auch Methoden und Prozeduren, die zu komplex geraten sind, zerlegen und die innere Verschachtelung in neue untergeordnete Methoden. Bzw. Prozeduren, auslagern. Damit sind die als Web Service wiederaufbereiteten Codebausteine weniger komplex.

4. Ergebnisse der Transformationen

Nach Vollendung der sieben Transformationsschritte wird die Wiederverwendbarkeit der ausgewählten Altprogramme als Web Services um ein vielfaches gestiegen sein. Alles was Ihrer Nutzung als einzelne Komponente in einer neuen Umgebung

beeinträchtigen würde, ist beseitigt. Sie enthalten keine fest verdrahteten Daten mehr. Sollte es welche gegeben haben, sind sie jetzt als einstellbare Parameter neu definiert. Die Verarbeitungslogik ist frei von eigener IO-Operationen und Datenbankzugriffen. Diese sind jetzt in speziellen IO-Routinen ausgelagert. Sie sind auch aus der Abhängigkeit zur technischen Runtime Umgebung befreit. Alle Aufrufe von Systemdienste sind in eingebaute Funktionen umgeleitet. Von besonderer Wichtigkeit ist der Zustand, dass sie von übergeordneten Klassen, untergeordneten Modulen und nebengeordneten Methoden und Funktionen abgekoppelt sind. Diese externen Dienstleistungen sind jetzt durch interne Methoden, bzw. Prozeduren, wahrgenommen. Schließlich sind externe GO TO Verzweigungen gekappt und zu komplex geratene Prozeduren oder Methoden zerlegt.
Erst jetzt sind die Legacy Sourcen in einem Zustand, in dem sie überhaupt in eine neue Umgebung versetzt werden können. Jetzt lassen sie sich, z.B. durch ein Werkzeug wie *SoftWrap* kapseln und als Web Service Komponente in einer Service-orientierten Architektur wiederverwenden. Dort müssen nur noch ihre Zugangsschnittstellen, die APIs, TP-Datenströme und Eingabedateien, in WSDL Schnittstellen umgewandelt und veröffentlicht werden, damit jeder auf sie zugreifen kann.

5. Andere Arbeiten auf diesem Gebiet

Es darf nicht unerwähnt bleiben, daß auch Andere auf diesem vielversprechenden Gebiet tätig sind. Das liegt daran, daß die Nachfrage nach fertigen Web Services immer größer wird. Da die wenigsten in der Lage sind, ihre Web Services selbst neu zu entwickeln, werden solche Software Komponenten auf dem Markt gesucht. Hier stoßen die Anwender jedoch schnell auf die große Kluft zwischen dem was fremde Web Services leisten und dem was ihre eigene Geschäftsprozesse brauchen. Sie stehen dann vor einer schwierigen Entscheidung. Entweder stellen sie ihre Geschäftsprozesse um, damit sie zu den angebotenen Web Services passen oder sie stellen ihre Programme um damit sie als Web Services dienen können. Natürlich können sie auch auf Web Services und Service-orientierte Architekturen verzichten und abwarten bis es was anders gibt. Dennoch sind eine größere Anzahl von Anwendern bereit den Sprung auf den neuen Zug zu wagen. Demzufolge sind auch eben soviele Dienstleister bereit ihnen dabei zu helfen.
Unter dem Stichwort „Code Mining“ ist die Firma *MicroFocus* schon lange bemüht aus alten COBOL Programmen wiederverwendbare Bausteine zu gewinnen. Sie bietet eine Reihe kommerzieller Werkzeuge speziell zu diesem Zweck an. [12] Die *Delta GmbH* mit Sitz in Deutschland bietet ebenso Werkzeuge an, mit denen Codebausteine aus alten COBOL und PL/I Programmen entnommen, gekapselt und als Web Service angeboten werden. [13] In der Schweiz befaßt sich die Firma *Zühlke Engineering* mit dieser Problematik und in Italien hat das *RCOST* Zentrum in Benevento bereits solche Migrationsprojekte für die kommunale Verwaltungen im E-Government Bereich durchgeführt. [14] Nicht zuletzt ist auch die *SAP* zu erwähnen, die bemüht ist, R3 ABAB Programme als Web Services unter *MySap* wiederzuverwenden. [15]

Im Bereich der Wissenschaft sind etliche Forscher mit dem Thema „Web Services“ beschäftigt aber nur wenige mit der Wiedergewinnung von Web Services aus Legacy Software. Führend hier sind Forscher an den Universitäten Durham in G.B. [16], Benevento in Italien [17] sowie Victoria und Alberta in Canada [18]. Alle betreiben Forschung mit dem Ziel Legacy Code als Web Service wiederzuverwenden. Bis jetzt haben sie lediglich Pilotprojekte vorzuweisen, aber es ist nur eine Frage der Zeit bis die Ergebnisse dieser Forschung in der Praxis Früchte tragen.

6. Zusammenfassung

In dem Maße wie sich Web Services durchsetzen wird die Nachfrage nach der Wiederverwendung bestehender Software als Web Services wachsen. Sowohl Anbieter als auch Anwender von Web Services werden gezwungen auf das große Reservoir an vorhandene Altsoftware zurückzugreifen. In dem Moment werden sie vor der Frage stehen, ob sie nur Algorithmen kopieren, den Objekt Code in der Urumgebung weiter verwenden oder ob sie den Source Code transformieren. Dieser Beitrag zeigte einen Weg auf, den Source Code zu transformieren. Das hier vorgestellte Transformationsverfahren ist für die Sprachen Assembler, PL/I, COBOL, C und C++ schon automatisiert und erprobt. Die Zukunft wird erst zeigen, ob die Anwender bereit sind diesen Weg zu gehen. Bis dahin bleibt diese Arbeit nur ein Forschungsprojekt.

Literaturhinweise:

[01] Conrad,S./Hasselbring, W./Koschel, A./Tritsch, R.. Enterprise Application Integration, Elsevier Spektrum Akademischer Verlag, München, 2006, S. 191

[02] Krafzig, D./Blanke, K./Slama, D.: Enterprise SOA, Prentice-Hall Pub., Upper Saddle River, N.J., 2004, S. 6

[03] Frakes, W./Terry, C.: „Software Reuse – Metrics and Models“, ACM Computing Surveys, Vol.28, 1996, S. 415-435.

[04] Krueger, C.: „Eliminating the Adaption Barrier“, IEEE Software, Aug. 2002, S. 29

[05] Aversano, L./Canfora, G./Cimitile, A./De Lucia, A.: „Migrating Legacy Systems to the Web - An Experience Report“, in Proc. of 5th IEEE CSMR, Lissabon, March 2001, S. 148

[06] Litoiu, M.: „Migrating to Web Services – a performanace engineering approach“, Journal of Software Maintenance and Evolution“, Vol. 16, Nr. 1, 2004, S. 51

[07] Sneed, H./Sneed, S.: Web-basierte Systemintegration, Vieweg Verlag, Wiesbaden, S. 257

[08] Binder,R.: Testing Object-oriented Systems, Addison-Wesley Longman, Reading Mass., 1999, S. 810

[09] Sellink, A./Sneed, H. M./Verhoef, C.: „Restructuring of COBOL/CICS Legacy Systems“, in Proc. of 3rd European CSMR, IEEE Press, Amsterdam, March 1999, S. 72

[10] Sneed,H.: “Measuring Reusability of Legacy Software Systems“, Journal of Software Process, Vol. 4, No. 1, March 1998, p. 43-48

[11] Sneed, H.: „Encapsulating Legacy Software for Reuse in Client/Server Systems”, Proc. of 3rd WCRE, IEEE Press, Monterey, CA., Nov. 1996, S.104

[12] Redaktion: “Mining für Web Services in Legacy Software” in Computerzeitung, Nr. 50, Dez. 2005, S.12

[13] Vasoros, A.: “Migrating Legacy Components to the Internet”, Proc. of Industrial Session, IEEE-CSMR 2002, Budapest, Sept. 2002, s.32

[14] Distante, D./Perrone, V.: „Migrating to the Web of a Legacy Application – The Sinfor Project“ in Proc. of 4th IEEE Workshop on Website Evolution, Montreal, Canada, Oct. 2002, S. 85.

[15] von Hahn:, E.: Werterhaltung von Software – Planung und Bewertung von Reengineering Maßnahmen am Beispiel von Standard Software, Deutscher Universitätsverlag, Wiesbaden, 2005, S. 173

[16] Lavery, J./Bodyreff, C./Ling, B./Allison, C.: „Modelling the Evolution of Legacy systems to web-based Systems“, Journal of Software Maintenance and Evolution, Vol. 16., Nr. 1, 2004, S. 5

[17] Bodhuin, T./Tortorella, M.: „Migrating COBOL Systems to the Web by using a MVC Design Pattern“, in Proc. of 9th IEEE WCRE, Richmond, Nov. 2002, S. 329

[18] Tilley, S./Huang, S./Müller, H./Wong, K.: „On the Business Value and technical Challenges of adapting Web Services“, Journal of Software Maintenance and evolution, Vol. 16, Nr. 1, 2004, .31

Available Research Reports (since 2000):

2006

1/2006 *Rainer Gimnich, Andreas Winter.* Workshop Software-Reengineering und Services.

2005

16/2005 *Kurt Lautenbach, Alexander Pinl.* Probability Propagation in Petri Nets.

15/2005 *Rainer Gimnich, Uwe Kaiser, Andreas Winter.* 2. Workshop "Reengineering Prozesse" — Software Migration —.

14/2005 *Jan Murray, Frieder Stolzenburg, Toshiaki Arai.* Hybrid State Machines with Timed Synchronization for Multi-Robot System Specification.

13/2005 *Reinhold Letz.* FTP 2005 — Fifth International Workshop on First-Order Theorem Proving.

12/2005 *Bernhard Beckert.* TABLEAUX 2005 — Position Papers and Tutorial Descriptions.

11/2005 *Dietrich Paulus, Detlev Droege.* Mixed-reality as a challenge to image understanding and artificial intelligence.

10/2005 *Jürgen Sauer.* 19. Workshop Planen, Scheduling und Konfigurieren / Entwerfen.

9/2005 *Pascal Hitzler, Carsten Lutz, Gerd Stumme.* Foundational Aspects of Ontologies.

8/2005 *Joachim Baumeister, Dietmar Seipel.* Knowledge Engineering and Software Engineering.

7/2005 *Benno Stein, Sven Meier zu Eißen.* Proceedings of the Second International Workshop on Text-Based Information Retrieval.

6/2005 *Andreas Winter, Jürgen Ebert.* Metamodel-driven Service Interoperability.

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

4/2005 *Torsten Gipp, Jürgen Ebert.* Web Engineering does profit from a Functional Approach.

3/2005 *Oliver Obst, Anita Maas, Joschka Boedecker.* HTN Planning for Flexible Coordination Of Multiagent Team Behavior.

2/2005 *Andreas von Hessling, Thomas Kleemann, Alex Sinner.* Semantic User Profiles and their Applications in a Mobile Environment.

1/2005 *Heni Ben Amor, Achim Rettinger.* Intelligent Exploration for Genetic Algorithms – Using Self-Organizing Maps in Evolutionary Computation.

2004

12/2004 *Manfred Rosendahl.* Objektorientierte Implementierung einer Constraint basierten geometrischen Modellierung.

11/2004 *Urs Kuhlmann, Harry Sneed, Andreas Winter.* Workshop Reengineering Prozesse (RePro 2004) — Fallstudien, Methoden, Vorgehen, Werkzeuge.

10/2004 *Bernhard Beckert, Gerd Beuster.* Formal Specification of Security-relevant Properties of User-Interfaces.

9/2004 *Bernhard Beckert, Martin Giese, Elmar Habermalz, Reiner Hähnle, Andreas Roth, Philipp Rümmer, Steffen Schlager.* Taclets: A New Paradigm for Constructing Interactive Theorem Provers.

8/2004 *Achim Rettinger.* Learning from Recorded Games: A Scoring Policy for Simulated Soccer Agents.

7/2004 *Oliver Obst, Markus Rollmann.* Spark — A Generic Simulator for Physical Multi-agent Simulations.

6/2004 *Frank Dylla, Alexander Ferrein, Gerhard Lakemeyer, Jan Murray, Oliver Obst, Thomas Röfer, Frieder Stolzenburg, Ubbo Visser, Thomas Wagner.* Towards a League-Independent Qualitative Soccer Theory for RoboCup.

5/2004 *Peter Baumgartner, Ulrich Furbach, Margret Groß-Hardt, Thomas Kleemann.* Model Based Deduction for Database Schema Reasoning.

4/2004 *Lutz Priese.* A Note on Recognizable Sets of Unranked and Unordered Trees.

3/2004 *Lutz Priese.* Petri Net DAG Languages and Regular Tree Languages with Synchronization.

2/2004 *Ulrich Furbach, Margret Groß-Hardt, Bernd Thomas, Tobias Weller, Alexander Wolf.* Issues Management: Erkennen und Beherrschen von kommunikativen Risiken und Chancen.

1/2004 *Andreas Winter, Carlo Simon.* Exchanging Business Process Models with GXL.

2003

18/2003 *Kurt Lautenbach.* Duality of Marked Place/Transition Nets.

17/2003 *Frieder Stolzenburg, Jan Murray, Karsten Sturm.* Multiagent Matching Algorithms With and Without Coach.

16/2003 *Peter Baumgartner, Paul A. Cairns, Michael Kohlhase, Erica Melis (Eds.).* Knowledge Representation and Automated Reasoning for E-Learning Systems.

15/2003 *Peter Baumgartner, Ulrich Furbach, Margret Gross-Hardt, Thomas Kleemann, Christoph Wernhard.* KRHyper Inside — Model Based Deduction in Applications.

14/2003 *Christoph Wernhard.* System Description: KRHyper.

13/2003 *Peter Baumgartner, Ulrich Furbach, Margret Gross-Hardt, Alex Sinner.* 'Living Book' :- 'Deduction', 'Slicing', 'Interaction'..

12/2003 *Heni Ben Amor, Oliver Obst, Jan Murray.* Fast, Neat and Under Control: Inverse Steering Behaviors for Physical Autonomous Agents.

11/2003 *Gerd Beuster, Thomas Kleemann, Bernd Thomas.* MIA - A Multi-Agent Location Based Information Systems for Mobile Users in 3G Networks.

10/2003 *Gerd Beuster, Ulrich Furbach, Margret Groß-Hardt, Bernd Thomas.* Automatic Classification for the Identification of Relationships in a Metadata Repository.

9/2003 *Nicholas Kushmerick, Bernd Thomas.* Adaptive information extraction: Core technologies for information agents.

8/2003 *Bernd Thomas.* Bottom-Up Learning of Logic Programs for Information Extraction from Hypertext Documents.

7/2003 *Ulrich Furbach.* AI - A Multiple Book Review.

6/2003 *Peter Baumgartner, Ulrich Furbach, Margret Groß-Hardt.* Living Books.

5/2003 *Oliver Obst.* Using Model-Based Diagnosis to Build Hypotheses about Spatial Environments.

4/2003 *Daniel Lohmann, Jürgen Ebert.* A Generalization of the Hyperspace Approach Using Meta-Models.

3/2003 *Marco Kögler, Oliver Obst.* Simulation League: The Next Generation.

2/2003 *Peter Baumgartner, Margret Groß-Hardt, Alex Sinner.* Living Book – Deduction, Slicing and Interaction.

1/2003 *Peter Baumgartner, Cesare Tinelli.* The Model Evolution Calculus.

2002

12/2002 *Kurt Lautenbach.* Logical Reasoning and Petri Nets.

11/2002 *Margret Groß-Hardt.* Processing of Concept Based Queries for XML Data.

10/2002 *Hanno Binder, Jérôme Diebold, Tobias Feldmann, Andreas Kern, David Polock, Dennis Reif, Stephan Schmidt, Frank Schmitt, Dieter Zöbel.* Fahrassistenzsystem zur Unterstützung beim Rückwärtsfahren mit einachsigen Gespannen.

9/2002 *Jürgen Ebert, Bernt Kullbach, Franz Lehner.* 4. Workshop Software Reengineering (Bad Honnef, 29./30. April 2002).

8/2002 *Richard C. Holt, Andreas Winter, Jingwei Wu.* Towards a Common Query Language for Reverse Engineering.

7/2002 *Jürgen Ebert, Bernt Kullbach, Volker Riediger, Andreas Winter.* GUPRO – Generic Understanding of Programs, An Overview.

6/2002 *Margret Groß-Hardt.* Concept based querying of semistructured data.

5/2002 *Anna Simon, Marianne Valerius.* User Requirements – Lessons Learned from a Computer Science Course.

4/2002 *Frieder Stolzenburg, Oliver Obst, Jan Murray.* Qualitative Velocity and Ball Interception.

3/2002 *Peter Baumgartner.* A First-Order Logic Davis-Putnam-Logemann-Loveland Procedure.

2/2002 *Peter Baumgartner, Ulrich Furbach.* Automated Deduction Techniques for the Management of Personalized Documents.

1/2002 *Jürgen Ebert, Bernt Kullbach, Franz Lehner.* 3. Workshop Software Reengineering (Bad Honnef, 10./11. Mai 2001).

2001

13/2001 *Annette Pook.* Schlussbericht "FUN - Funkunterrichtsnetzwerk".

12/2001 *Toshiaki Arai, Frieder Stolzenburg.* Multiagent Systems Specification by UML Statecharts Aiming at Intelligent Manufacturing.

11/2001 *Kurt Lautenbach.* Reproducibility of the Empty Marking.

10/2001 *Jan Murray.* Specifying Agents with UML in Robotic Soccer.

9/2001 *Andreas Winter.* Exchanging Graphs with GXL.

8/2001 *Marianne Valerius, Anna Simon.* Slicing Book Technology — eine neue Technik für eine neue Lehre?.

7/2001 *Bernt Kullbach, Volker Riediger.* Folding: An Approach to Enable Program Understanding of Preprocessed Languages.

6/2001 *Frieder Stolzenburg.* From the Specification of Multiagent Systems by Statecharts to their Formal Analysis by Model Checking.

5/2001 *Oliver Obst.* Specifying Rational Agents with Statecharts and Utility Functions.

4/2001 *Torsten Gipp, Jürgen Ebert.* Conceptual Modelling and Web Site Generation using Graph Technology.

3/2001 *Carlos I. Chesñevar, Jürgen Dix, Frieder Stolzenburg, Guillermo R. Simari.* Relating Defeasible and Normal Logic Programming through Transformation Properties.

2/2001 *Carola Lange, Harry M. Sneed, Andreas Winter.* Applying GUPRO to GEOS – A Case Study.

1/2001 *Pascal von Hutten, Stephan Philippi.* Modelling a concurrent ray-tracing algorithm using object-oriented Petri-Nets.

8/2000 *Jürgen Ebert, Bernt Kullbach, Franz Lehner (Hrsg.).* 2. Workshop Software Reengineering (Bad Honnef, 11./12. Mai 2000).

7/2000 *Stephan Philippi.* AWPN 2000 - 7. Workshop Algorithmen und Werkzeuge für Petrinetze, Koblenz, 02.-03. Oktober 2000 .

6/2000 *Jan Murray, Oliver Obst, Frieder Stolzenburg.* Towards a Logical Approach for Soccer Agents Engineering.

5/2000 *Peter Baumgartner, Hantao Zhang (Eds.).* FTP 2000 – Third International Workshop on First-Order Theorem Proving, St Andrews, Scotland, July 2000.

4/2000 *Frieder Stolzenburg, Alejandro J. García, Carlos I. Chesñevar, Guillermo R. Simari.* Introducing Generalized Specificity in Logic Programming.

3/2000 *Ingar Uhe, Manfred Rosendahl.* Specification of Symbols and Implementation of Their Constraints in JKogge.

2/2000 *Peter Baumgartner, Fabio Massacci.* The Taming of the (X)OR.

1/2000 *Richard C. Holt, Andreas Winter, Andy Schürr.* GXL: Towards a Standard Exchange Format.