Continuous Engineering for Industrial Scale Software Systems

Organizers: Herbert Weber, Hausi Müller

Dagstuhl Seminar 98092

Contents

Introduction

by Herbert Weber, Technical University of Berlin
and Fraunhofer ISST Berlin/Dortmund, Germany

Information and Communication Systems are growing together to long-living Information and Communication Infrastructures that are mission critical to organizations, businesses and the whole society. As any other kinds of artefacts Information and Communication Infrastructures age over time getting out of synchrony with business demands and advanced technology. As a consequence they need their continuous modification to live up to the changing requirements.

A solid scientific foundation for a continuous engineering of those infrastructures is on its early stage: most of the practical approaches are rather ad-hoc approaches and lack systematics and engineering rigor. Scientific approaches often lack the association with reality.

The seminar is intended to bring people with industrial experience and solid scientific background together to lay the ground for a scientific underpinning of practical continuous engineering.

Continuous Engineering of Information and Communication Infrastructures — A Position Statement

by Herbert Weber, Technical University of Berlin
and Fraunhofer ISST Berlin/Dortmund, Germany

New Challenges

Radical changes are taking place throughout the economy. Efforts aimed at gaining a competitive edge in terms of price and quality are seldom limited to regional markets. Today's entrepreneurs face competition on a global scale, and this global change has put them under increasing pressure to review traditional products and organizational and work processes – including the underlying information technology.

Information has long since become a production factor. A smooth and steady flow of information, both within and between organisations, is now a strategic imperative. A competitive advantage will thus only be enjoyed by those who exploit to the full the potential benefits of modern information and communication technology.

Information highways, broadband communication, multimedia, video conferences, teleworking and telecooperation – these are just a few of the

catch-words used to denote the possibilities seen in the years ahead. This development heralds the end of an era in which centralized company departments were made responsible to provide for the information technology requirements of all users.

The 1980s were characterized by dramatic advances in hardware development. Large mainframes were complemented by midrange computers which were in turn complemented by powerful personal computers and workstations.

Networked workstations have already brought about a decentralization of computer services in many corporations. These networks encourage the integration of independent data processing capabilities and isolated subsystems into larger configurations.

A further change in the structure of information technology has been taken place since the beginning of the 1990's. Individual information systems were being inter-connected to form coherent structures.

Such integrations lead to new configurations which – analogous to other basic structural forms – are referred to as information and communication infrastructures.

Transforming independent information systems into integrated structures also changes their basic characteristics. Whereas previously such systems were developed to satisfy immediate needs, the principle aim pursued in the development of Information and Communication Infrastructures is to ensure that the systems' useful life can be prolonged indefinitely, in much the same way as this principle applies to comparable infrastructural entities, such as road networks and telephone systems.

New Applications

Business Applications

Changes in the management structures implemented in industry have intensified the demand for an effective exchange of information and improved efficiency in communication between collaborative work groups. The transitions from closed DP "islands" to integrated distributed system solutions as well as the division of corporations into cooperative and flexible profit centers, the flattening of administrative hierarchies and the integration of outsourced external services have become an essential factor in responding to global markets.

Various developments in information technology are currently paving the way for the inter-connection of independent applications:

- Client/server systems based on PCs and workstations complement centralized mainframe architectures.
- Computers are being transformed into working environments capable of supporting complex work processes.
- Substantial progress has been made in establishing international standards for protocols, platforms, and interfaces, such as TCP/IP, http, HTML, X.400, X.500, SQL, EuroISDN.
- Manufacturer alliances are being formed to enforce industrial standards, for example OMG with CORBA, OSF with MOTIF and DCE.
- Simple alphanumeric terminals are being replaced by others equipped with graphic interface systems, window technology and multimedia capabilities.
- High-speed broadband networks are advancing rapidly to replace slower conventional telecommunications lines.
- Groupware and workflow systems enable the coordination of teamwork in offices and even the coordination of work in large organisations.
- Multimedia systems allow for the visualization of complex information.
- Virtual reality systems allow the simulation of activities in 3D spaces etc.

All these new concepts and technologies contribute to the "re-vitalization" or even "re-invention" of many applications like

- in the banking business
- in the insurance business
- in commerce
- in the publication business
- in the logistics business
- in tourism
- in the service and maintenance business

etc.

Technical application

Information and communication technologies are paving the way towards intelligent products of many different kinds. Therefore they are developed as an integral part of many engineering systems like

- automobiles
- aircraft and space vehicles, and
- last but not least, many houshold goods.

The infrastructures in demand here are expected to conform to especially high quality standards which in turn lead to demands for "safty-critical" or even "zero-defect" software in information and communication infrastructures.

Many new products like telephone sets, consumer electronic products, micro-electronic devices etc. are subject to permanent improvements and refinements and this demands for their continuous engineering. These improvements and refinements are whereever possible left to changes to software of the embedded information and communication infrastructures.

Manufacturing technologies undergo again dramatic changes. Early attempts to fully integrate different stages in the production of goods are reconsidered now and brought to a consolidation based on new integration technologies for information and communication infrastructures. As a consequence manufactured goods are brought to the market faster and global competition is largely based on "time-to-market" performance of corporations. By the same token the continuous improvement and refinement of goods demands for continuous adaptations of their manufacturing technologies. Once again many of these adaptions are meant to be achieved with changes in the supporting information and communication infrastructure.

Most visibly are at the moment dramatic changes in production technologies for the car manufacturing. These publicly debated revolutionary developments represent however just the tip of the iceberg. Both the production and disposal of almost all technical goods is being reconsidered, new production technologies based on new divisions of labor and on new information and communication technologies are implemented in almost all sectors of the industry.

Last but not least, information and communication technologies are replacing traditional technologies. The best known example is the replacement of hydrolic steering and control systems in aircrafts by "fly-by-wire" technologies, which are again based on highly complex information and communication infrastructures.

New Structures

Substantial productivity gains can be achieved in many domains with the amalgamation of previously autonomous computer applications. In this way, programs and software systems do not remain separate but become linked via different kinds of integration and communication platforms into compounds that become critically important to the application. As a consequence, many of the programs and software systems installed in recent years have evolved into assets of an unexpected durability. We refer to such durable and evolving systems as Information and Communication Infrastructures.

Like other kinds of infrastructures, Information and Communication Infrastructures are developed for continuous use and to enable adaptations and extensions through partial renewal, partial extension and partial replacement, without changes to the underlying basic concept. The establishment of Information and Communication Infrastructures thus demands a wide range of skills:

- the skill of the business consultant for the analysis of business structures and practices which the infrastructures are meant to support,
- the skill of the engineer for the analysis of the engineering problem and for the specification of the requirements for the needed infrastructure,
- the skill of the technology advisor for the choice of the right concepts and systems for the infrastructure,
- the skill of the technology developer for the adaptation of existing products to their use in an infrastructure,
- the skill of the technology and systems integrator for the amalgamation of individual technologies and systems into a coherent overall infrastructure.

Information and communication infrastructures are meant to provide numerous decisive advantages when compared with traditional systems:

- IT infrastructures are built to be improved in an evolutionary manner based on a fundamental concept that has been carefully planned and established.
- IT infrastructures are built to be altered and extended at any time, allowing entrepreneurs essential flexibility in responding to changing requirements.

- IT infrastructures make it possible to utilize new and improved components in conjunction with older existing systems.
- This capacity to develop infrastructures gradually over an extended period of time safeguards investments.

IT infrastructures present a substantial change to software and systems engineering. The logical relationships between the components in software and systems are highly complex, and this complexity increases as these systems themselves are combined into a larger whole. That demands for modern architectures that are

- **modular:**
  Components are constructed as independent, reusable entities.
- **open:**
  Systems can be used and adapted over long times
- **compatible:**
  Structures have to be established that are compatible to acceptanced principles, and the use of standardized interfaces, components and invariant platforms.

New Characteristics

Information and Communication Infrastructures are in most cases product, mission or even business critical. They must guarantee maximum safety and reliability. This is particularly true for systems used in high-risk applications, such as nuclear power stations and aircraft piloting, traffic control, banking and telephone networks.

The infrastructures currently used often do not fulfil this prerequisite. In many cases there are substantial shortcomings in their quality:

- Systems are often faulty and unreliable. This can often be attributed to such factors as unsystematic engineering techniques, a lack of theory and formalization, inadequate quality assurance, and bad systems management, among others.
- The architecture and functions of legacy systems are often not known in detail. In the majority of cases this is attributable to inadequate maintenance techniques, and to incomplete or non-existent documentation.

If these factors are present simultaneously, systems can become an incalculable risk. This presents corporate users with new challenges, particularly in view of their exposure to competition in international markets.

New Technologies

Network Technologies

Network technologies facilitate the realization of integrated Information and Communication Infrastructures, offering several substantial benefits:

- They support geographically distributed organizations.
- They protect investments made in independent systems and user-specific software.
- They reduce the high cost of new developments through the integration of existing systems.
- They improve and accelerate work practices by providing reliable collaboration support to users.

Network technologies allow now the deployment of:

- heterogeneous communication systems
- heterogeneous information management systems, comprising file systems, relational and object-oriented databases
- different hardware and software within the same Information and Communication Infrastructure

Process Management

Private enterprises and public organizations continuously address themselves to the task of reorganization. Business concepts – such as lean management, lean production or business process reengineering – can only be put into practice successfully if supported by information technology. Companies with flat, customer-oriented management structures must be able to call upon high-performance communication systems that are readily adaptable to changing needs.

The organization of activities into work groups plays a key role in the decentralized management structures being adopted today. The effectiveness of such groups has a decisive influence on critical business performance factors such as costs, processing time, productivity, time ot market and the quality of products and services.

Industry and public organizations are increasingly aware of the need to enhance the effectiveness of such work groups. They are now taking a critical look at their organizational structures and procedures, as well as at the corresponding business processes.

Computer supported process management provides organizations with far-reaching advantages:

- Explicit models enable cooperative processes to be more readily understood and critically examined.
- Teamwork is improved, and cooperation between team members is guided and comprehensively supported.
- Problems in communication, attributable to misunderstandings, misdirection etc., can be alleviated.

Information Management

Development of Information and Communication Infrastructures increasingly relies upon the implementation of comprehensive information models.

These information models:

- offer a uniform, global view on information maintained in an enterprise, and
- enable the documentation of relationships between the data compiled in different parts of an organization

They facilitate:

- a better understanding of complex organizational structures
- the application integration into Information and Communication Infrastructures
- the systematic design of software

New modelling techniques are needed to

- document the semantic content of data at various levels of abstraction
- check on their syntactic and semantic consistency
- enable the development of domain and customer specific view on models
- enable their implementation by means of different kind of data management systems like data bases, object bases etc.
- enable the coexistence of different models in heterogeneous Information and Communication Infrastructures
- enable the re-engineering and migration of models

Continuous Engineering

Software and systems that make up Information and Communication Infrastructures are different in nature from those that serve as products of a limited lifetime. Their development and evolution require practises other than those of classical software and systems engineering.

1. Infrastructures must be continuously enhanced, so that they can cope with changing user or business requirements with ease.

2. Existing systems must be easy to amalgamate with new systems in an infrastructure.

3. The replacement of components in existing infrastructures by new, improved components must be straightforward and the migration from an old infrastructure to an updated one must take place in a controlled way.

4. The porting of systems in an infrastructure from one standard platform to another standard platform must be possible with minimum effort.

5. The reuse of existing systems and components in the same and in other infrastructures must be possible with minimum effort.

Software represents the predominant investment factor in Information and Communication Infrastructures. Over many years and at great expense companies and public organizations have created infrastructures which satisfy their specific requirements. As a consequence many organizations continue to use software systems that are far too rigid to respond to the rapidly changing conditions.

Substantial efforts must therefore be invested to modernise those systems and to acquire detailed knowledge of how they were designed and how they function in their reverse engineering.

This is an essential step prior to altering or extending that software in its reengineering. The reengineering of legacy software systems opens up a wide variety of possibilities:

- Protection of existing investments

- Structural improvement of outdated or substantially altered software systems

- Increased comprehensibility of programs

- Adaptation of existing software to new tasks and operating environments
- Restructuring, for example from mainframe applications to client/ server architectures
- Modernization of software, for example, with sophisticated graphical user interfaces
- Improvement in the operation and maintenance properties of software

The possibilities outlined above must, however, be offset of the costs involved.

Reverse Engineering Strategies for Software Migration

by Hausi A. Müller, University of Victoria, Canada

The need for maintaining and improving software and information systems has risen dramatically over the past decade. Dealing with old software systems, which constitute billion-dollar assets to corporations and governments, has been recognized as a critical problem by industry, academia, and entrepreneurs. Migrating and reengineering involves capturing, preserving, and extending knowledge about software, analyzing and understanding software, and finally changing, improving, and evolving software. Reverse engineering approaches have been particularly useful in the reengineering arena. Reverse engineering is the process of generating new information about software such as synthesizing abstractions and generating different views. This presentation concentrates on the reverse engineering strategies for software migrationand reports on our experience migrating PL/I to C++ code in collaboration with IBM Toronto Laboratory.

About the author

Hausi A. Müller received a Diploma Degree in Electrical Engineering from the Swiss Federal Institute (ETH) in Zurich. From 1979 to 1982 he worked as a software engineer for Brown Boveri & Cie in Baden, Switzerland (now called ASEA Brown Boveri). In 1984 and 1986, he received M. Sc. and Ph. D. degrees in computer science from Rice University, Houston, Texas, respectively. Since 1986 he has been at the University of Victoria, British Columbia where he is an Associate Professor of Computer Science and served as Acting Chair for 1995/96.

In 1992/93 Dr. Müller was on sabbatical at the Centre for Advanced Studies in the IBM Toronto Laboratory working with the program understanding group. While at IBM, Müller analyzed the source code of SQL/DS, a multi-million-line database management system, using his Rigi reverse engineering environment. He currently is a principal investigator of CSER (Consortium for Software Engineering Research), a Canadian, industry-lead consortium sponsored by NSERC and NRC.

His research interests include software engineering, software evolution, reverse engineering, software reengineering, software migration, program understanding, software architecture, and software maintenance. He was a Program Co-Chair for the IEEE International Conference on Software Maintenance-ICSM '94 in Victoria, September 19–23, 1994, for the 7th IEEE International Workshop on Computer-Aided Software Engineering- CASE '95 in Toronto, July 10–14, 1995, and the IEEE 4th International Workshop on Program Comprehension-WPC '96 in Berlin, March 29–31, 1996. He is on the Editorial Board of IEEE Transactions on Software Engineering.

Dr. Müller is President of Hypersystems Technologies Inc., a company specializing in tools and consulting for reengineering and migrating legacy software systems such as for the Year 2000 Problem.

Transformations of Software Architectural Structures

by Ric Holt, University of Waterloo, Canada

There is a method of software architecture extraction in which the person doing the extraction takes a hybrid approach, deriving facts from the code and determining heriarchic structure by interviewing persons familiar with the software. These two sources of informations are merged to derive the higher level structure (the concrete architecture) of the software system.

This work is illustrated using Software Bookshelf tools operating on a 250,000 line industrial program. An algebraic approach is used to transform the low level structure from source code to higher level structures, at the architectural level.

The low level facts are represented as edges (tuples) in a typed graph. Using Tarski's theory that defines an algebra of operations on such graphs, these facts are manipulated to high level facts. These operations include relation union, intersection, composition, transitive closure, etc., all incorporated into a script language called Grok. Using this language, relations are aggregated, sub-trees are consolidated to single nodes, and sub-trees (subsystems) are separated for analysis. All of these manipulations, or structure

transformations, can be thought of as data base queries, and are written as Tarski graph operators.

Evolving Legacy Systems: New Technologies Supporting Architectural Views of Software

by Steven Woods, Carnegie Mellon University, USA

The SEI's Product Line Systems (PLS) Program aims to enable widespread Product Line Practice (PLP) through architecture-based development. PLP focuses on engineering and reengineering software-intensive systems from the perspective that that an organization's evolving software asset base is best viewed as a core "product" with a corresponding reference architecture. THis "core" represents an attempt to manage and constrain possible system variability according to carefully determined enterprise-wide business goals. While each organization's differences entail different paths towards sound PLP, the SEI is refining a framework accomodating various organizational starting points. One key aspect of moving towards product lines is the transitioning between legacy systems (and legacy architecture) and a carefully designed (new) target architecture. The process of acquiring a high-level system understanding can be assisted through the application of emerging technologies in computing — in particular, the SEI is currently investigating application of architectural recovery, distributed object technologies and net-centric computing to reengineering. Reengineering tools and methods help leverage software legacy assets by supporting expert architectural analysis and extraction through software visualization and flexible, intelligent componentization. While many similar and complimentary tools exist, the interoperability of these reengineering tools has long been a source of difficulty for researchers and practictioners alike. An attempt at defining and reaching consensus on an open middleware schema (CORUM) for reengineering tools has recently been commenced in cooperation with no fewer than 6 concerned institutions, and this work is continuing as anevolving middleware for SEI's architectural analysis toolset (DALI).

Component and Connector Abstractions in Software Design for Integrated Object Systems

by Stefan Tai, Technical University of Berlin, Germany

In our research, we focus on two major directions supporting continuous software engineering: *software architecture*, and *object integration technologies*.

In software architecture, a system is viewed and modeled as a set of components and connectors. Components are abstractions of system level computational parts, connectors are abstractions of component interdependencies. The components, and the mechanisms guiding their interoperation (like communication styles and interaction protocols), are recorded as separate, distinct architectural abstractions. This separation of design concerns leads to system representations that make information about component connections explicit, thus facilitates component (ex)change and partial modifications.

Object integration technologies like the OMG's CORBA, on the other hand, address implementation support for the integration of diverse software components in distributed, heterogeneous environments. They define standard component interconnection and interoperation models to encapsulate incompatible component implementations: All integrated software entities are considered objects that export object-like interfaces and interact by means of method invocation.

In order to effectively abstract, structure, and express the software architecture of systems that are continuously build using object integration technology, dedicated software architectural design support is needed. Furthermore, impacts on software system design as introduced by the chosen object technology (for example, regarding general purpose object services like the CORBAServices) must be well understood and recorded to support system understanding and continuous development.

In this talk, we propose a software component and connector model for architectural representation of integrated object systems. We argue to represent the software architecture of ORB-based systems on two different levels of abstraction: using connectors as pattern-like, distinct abstractions of component interactions in an abstract architecture, and using component abstractions only that relate to ORB object computational models in a concrete architecture. We exemplify our concepts for modeling CORBA systems and discuss related design issues of continuous engineering on an example of a large, long-lived system.

The Contribution of Process Management to a Continuous Engineering of Office Applications

by Wolfgang Deiters, Fraunhofer ISST Dortmund, Germany

A systematic management of business processes is considered to be one approach that contributes to a continuous engineering of software systems. Managment of business process means on the one hand side the proper defi-

nition, analysis and improvement of business processes, thus yielding in improved process oriented business organizations. On the other hand side it means the process oriented design and implementation of the IT-systems supporting the organization's business. One enabling technology supporting a process oriented IT-development is the workflow technology. The basic principle of that technology is to define the IT relevant parts of the business processes in so called workflow models. The main parts of a workflow model are the definition of activities to be performed in the processes, the definition of the possible activity schedules that are valid and the definitions of responsibilities (i. e. the definition of who is allowed to perform which activity). In a workflow system the workflow models are being processed by a workflow engine thus driving the process along. For supporting the various activities tool services are being called during activity execution.

There are a couple of workflow management systems available on the market that support the workflow management paradigm. However, in many cases there are several problems that hinder the proper development of workflow based applications. Among these problems are the legacy problem, i. e. the problem to adaequately integrate existent applications into a workflow application, the data problem, i. e. the necessity to partially store redundant data that could lead to inconsistencies, and the problem of how to appropriately separate legacy systems' functionality from workflow system's functionality.

Furthermore workflow management system that are available today still lack some concepts for exception handling and ad hoc modifications. With respect to this issue concepts for a late modelling of processes and concepts for a flexible process model configuration are sketched. Furthermore, a classification scheme for distinguishing between different classes of structured and semi-structured processes is shown. For supporting processes belonging to different classes of that classification scheme different techniques (beside workflow management systems) have been developed or are being developed. The problem to integrate these different kinds of CSCW systems is addressed and the need to develop a service oriented architectural framework for the integration of workflow systems, groupware systems and other kinds of CSCW systems is discussed.

A Workflow-Based Software Architecture for Flexible Systems

by Sebastian Erdmann, Technical University of Berlin, Germany

Many organisations are faced with the need to reorganise their business from functional to process oriented approaches. Such reorganisation can be supported by workflow technology.

By using process models, the global flow of control and flow of business-relevant data is modelled explicitly. Formerly, it was hidden in custom (or off-the-shelf) software systems. Separating the process model, the computational aspects, and the data management enhances the system flexibility and configurability.

In current workflow management systems, however, the integration of external software components is problematic. In particular, the data exchange between the workflow management system and external software relies on ad-hoc solutions. Two data models of the business-relevant data exist: one in the workflow schema, and one in the application model.

We propose a solution based on the FUNSOFT process modelling language. We extend the FUNSOFT language by an object-oriented type system and provide a common meta model which integrates process modelling and the object-oriented modelling. Thereby, consistency between both kinds of models can be achieved.

We use the viewpoint concept of RM-ODP as a general framework for our models. Process models are used in the *Enterprise Viewpoint* to specify general relations between actors, activities, and data. Object-oriented modelling is used in the *Information Viewpoint* and the *Computational Viewpoint* to specify the structural and behavioural properties of the system, respectively.

On Inserting Program Understanding Technology into the Software Change Process

by Kenny Wong, University of Victoria, Canada

Program understanding technologies can be applied effectively in the analysis phase of a software change process. The analysis phase naturally follows a goal-driven metaprocess. Described are issues involved with inserting program understanding technology into existing practice and into such a metaprocess. The implied processes of program understanding and reverse engineering tools play an important role. These issues pose major programs for the acceptance of redocumentation tools such as Rigi, an evolvable re-

verse engineering tool. An example using Rigi and its analysis methodology for change-impact analysis is considered.

Program understanding tools and techniques can play an important role in the analysis part of the software change process. In particular, tools such as Rigi can use exact interfaces to help answer the question of whether certain changes should proceed. It is important to recognize the issues limiting the adoption of program understanding technology for continuous engineering. Tools have their own implied methodologies, processes, and strategies of use. These implied processes need to coexist within an analysis process that is naturally goal-directed. However, some tools have substantial preparation and forward-feeding "eager" activities. These aspects and other overhead may account partly for the difficulty of introducing certain program understanding tools into the development environment, where the tool users have high expectations of quick, guaranteed results, with no learning curve.

Open, lightweight tools that are specialized to do only a few things very well may be needed for easier technology insertion. Adaptable tools where it is simple to work around or recover after their problems are needed. Toward this need, Rigi supports tool evolution where the tool becomes more domain-specific. Strategies such as conceptual modeling and script writing are used to help capture analysis experience and knowledge about the business. The tradeoff is that although these activities may be reusable, they are often time consuming.

Native Patterns

by Alex Sellink and Chris Verhoef, University of Amsterdam, The Netherlands

We generate a native pattern language from a context-free grammar. So if we have the underlying grammar of code that needs to be analyzed, or renovated the pattern language comes for free. We use native patterns for recognition and renovation of code. The pattern language is global in the sense that patterns can match entire programs. We illustrate native patterns by discussing a tool that remediates a notoriously difficult Year 2000 problem using native patterns.

Categories and Subject Description:
D.2.6 [Software Engineering]: Programming Environments–Interactive;
D.2.7 [Software Engineering]: Distribution and Maintenance–Restructuring;
D.3.4 [Processors]: Parsing.

Additional Key Words and Phrases:
Reengineering, System renovation, Patterns, Plans, Clichis, Native pattern language, Year 2000 problem, Leap year problem.

Development, Assessment, and Reengineering of Language Descriptions

by Alex Sellink and Chris Verhoef, University of Amsterdam, The Netherlands

We discuss tools that aid in the developent, the assessment and the reengineering of language descriptions. The assessment tools give an indication what is wrong with an existing language description, and give hints towards correction. From a correct and complete language description, it is possible to generate a parser, a manual, and on-line documentation. The parser is geared towards reengineering purposes, but is also used to parse the examples that are contained in the documentation. The reengineered language description is a basic ingredient for a reengineering factory that can manipulate this language. We demonstrate our approach with a proprietary language for real time embedded software systems that is used in telecom industry. The described tool support can also be used to develop a language standard without syntax errors in the language description and the code examples.

Categories and Subject Description:
D.2.6 [Software Engineering]: Programming Environments–Interactive;
D.2.7 [Software Engineering]: Distribution and Maintenance–Restructuring;
D.3.4 [Processors]: Parsing.

Additional Key Words and Phrases:
Reengineering, System renovation, Language description development, language engineering, grammar reengineering, generation of manuals, generation of on-line documentation, computer aided language engineering, CALE, Message Sequence Charts.

Data is the Core of Continuous Engineering

by Ralf Kramer and Ulrike Kölsch, FZI Karlsruhe, Germany

Introduction and Motivation

Most re-engineering approaches for dealing with legacy systems focus on the re-engineering of programs. From our perspective and experience, however, these approaches are neither sufficient for implementing the changes in these

huge information and application systems that run the business of the enterprises where they are employed, nor are they sufficient to effect enduring change in the business processes of an enterprise.

The core of every legacy (or heritage) system and application is data. In order to re-engineer existing applications, it is mandatory that the semantics of the data be understood, because the semantics of the data determine the semantics of the application program. The approach developed in our previous projects, the extraction of an object-oriented model of a system's data and its functionality obtained by viewing the given system and extracting information from the data structure description, application programs, and documentation provides the background and the methodology we use in this paper. As a result of this reverse engineering approach, we achieve a data-centered object-oriented model of the legacy application.

The reverse engineered, object-oriented model as a representative of the legacy system's data and functionality is the appropriate basis for re-engineering and, in the long run, for continuously engineering information and application systems.

In the remainder of this position paper, we want to provide evidence for this claim based on our projects in two fields: namely, environmental information systems for public authorities and financial systems.

Experience from real-world projects

In both application areas, technological advances were the starting point:

- For environmental information systems (EIS) used by public authorities, Web technology offered the opportunity to serve both in-house users and the public (as required by EU law) cost efficiently for the first time.

- In the finance sector, middleware for open client/server architectures (OMG's CORBA) was the starting point, because it promised to allow for more user-friendly, distributed and loosely-coupled applications.

Distributed Environmental Information Systems

In a fairly large number of EIS projects, we started with the almost classical approach of using HTML, HTTP, and CGI to access (relational) databases, geographic information systems (GIS) and other systems that were already in place. In the second generation, this approach was complemented by Java-based retrieval modules that allowed the implementation of advanced retrieval capabilities and the avoidance of some of the deficiencies of the CGI

approach. Due to rapidly evolving technologies in this field, we currently are dealing with the third generation of Web-based systems, which take advantage of Java for data updates throughout the Web, as well.

Parallel to the evolution of technologies, the underlying databases were only changed modestly. This holds both for databases that have already been in place for some time (like the EIS Baden-Württemberg) and for databases that are built at the same time as Web-based software is being developed. (like WebCDS for the European Environment Agency). This supports our claim that data is the most valuable asset each organization has and that data is the core of every approach to develop new application software.

Finance Sector

In the finance sector, mainframes play an important role, because they ensure reliability in the presence of mass data processing, secure and safe 24 hours a day, 7 days a week, 52 weeks a year operation, scalability up to terabytes of data stored in databases (not necessarily in database management systems), and a level of security which can be considered to be much higher than that of client/server systems. Furthermore, those mainframe systems guarantee that all data is handled in a legally correct fashion by a supervisiory banking agency which makes sure that mainframe systems fulfill all legal requirements. Obviously, these host-based application and information systems are highly-sensitive and crucial for the banking industry.

On the other hand, these systems often suffer from severe drawbacks in their development and maintenance history. They are often written in 3GL or even Assembler, they use proprietary and degenerated interfaces for their data, they often use file system-based databases and not DBMS, and they use clumsy and user-unfriendly character-oriented interfaces.

Since users become increasingly well-acquainted to GUIs, terminal approaches are clearly inadequate. Given the promising nature of open middleware approaches, that support heterogeneous and distributed environments, such as CORBA; it comes as no surprise that a distributed architecture in which data is managed locally is an approach worth investigating. Another appealling concept is that of the object-oriented modelling of business objects and their processes combined with the promise of fast and easy changes and the adoption of such a model for changing business requirements.

However, when starting with a purely object-oriented data model, there is no clue about the model inherent to the already existing application. It is like the design and development of a stand-alone application with no environment to react to and no significant considerations to take into account about how to co-exist with existing application systems or third vendor systems (like

e.g. SAP R3). Obviously, this is not an adequate or sensible approach in an enterprise with a long history of using IT as there will always be existing systems to co-operate and to coordinate with. There are always official standards to be met and so-called standards which are non-offical, but nevertheless compelling, as well as regulations to obey.

Thus, as a consequence of the situation described, we need to adapt the object-oriented modelling and software engineering technique, in order to turn it into an evolutionary and adaptive approach applicable in real world situations.

Proposed Approach

Based on our experience, we do not propose developing a new business object model from scratch. Instead, it makes more sense to reverse engineer existing data, physical data structures, and applications, in order to obtain an object-oriented model of the given information system. Since the already existing information system and its data describe the same are as the intended new application, in most cases a proper object-oriented model can be extracted by stripping away the technical constraints of the older systems used. The reverse object-oriented model is not designed as perfectly as a new one, but it is the result of years of use and fine tuning to an operational application system. So this model takes into account the real needs of the business and focuses on details often overlooked in the first run in modeling approaches that start from scratch, because they lack the accumulated knowledge.

This object-oriented model provides the basis for the new development of an application system. It sets the limits for the new application, in such a way that there are no changes allowed to the model, in order not to risk the co-operation of old and new applications. At the same time, it also clearly illustrates the deficiencies of the existing applications and data and provides the possibility of enhancing and enlarging the model, in order to support the new business features of the application to be developed.

Conclusions

Given a situation in which companies want to react to changes in the market in a more timely fashion, existing and newly developed applications are aging at an accelerated speed. The faster the business changes, the faster the applications crucial for the success of a business become out-dated. Thus, application systems have not only to be re-engineered, but they have to be enhanced and adapted in order to preserve and enhance their functionality as well. These enhancements in their functionality require additional data,

as well. Based on our project experience, especially in the finance sector, we argue that in certain cases a semantic understanding of the host data and modular additions to this data are an adequate approach to provide the newly required functionalities. Neither a complete migration to a distributed client/server architecture, nor the co-existence of the host data and application and the new client/server application are adequate.

Because of the characteristics of contemporary system development and system evolution, it is absolutely necessary to come up with an approach which allows the permanent and evolutionary engineering of software for new and already-existing information and application systems. The future success of software engineering is dependent upon adaption to the ever-changing needs of its customers. Hence, an approach which enables the real and foreseeable continuous engineering of systems is required. The requirement is to plan and construct with room for changes, adoptions, and enhancements, since this will happen in the lifetime of every large information system. As illustrated by Web-based EIS, especially in fields using rapidly evolving techniques, this is even likely to happen in the short run. Furthermore, it is a fact that in real-world software processes, there is always a legacy to work with, to co-operate with, and to integrate the new systems into; software development does not start from scratch, although this is the impression often given. Given a flexible concept of semantic and technical integration, the co-existence and a fruitful co-operation between the systems of different origin and generation can be easily achieved without friction and without the problem of embedding a newly developed application system in a legacy information technology structure and culture as is often encountered.

Acknowledgments

We would like to thank all members of the Database Research Group at FZI (Wassili Kazakos, Arne Koschel, Ralf Nikolai, Claudia Rolker, Rainer Schmidt, Mechtild Wallrath, and Sonja Zwißler) and Prof. Dr. P. C. Lockemann for their ideas, efforts, and contributions to the projects that are the background for this position paper.

Evolving Objects: Conceptual Description of Adaptive Information Systems

by Gunter Saake, University of Magdeburg, Germany

A common metaphor is to see program construction as similar to building a house: an architect is planning the structure, building a house is a sequence of

concrete steps, and the building process terminates after finishing the house. For information systems, the 'information-system-as-city' metaphor is more appropriate: an information system consists of many buildings (= programs) using a shared infrastructure; building a city is a vivid and sometimes chaotic process; there are rather restrictions than concrete prescriptions for building houses; and the construction process will never be finished. Old and new buildings have to co-exist, and old buildings are used for purposes the never been planned for.

The talk discusses the need for adaptive database applications as components of such an information system. Database objects, for example in a bank application or production documents, may have a very long life-span. Some information has to be stored and manipulated for centuries. Together with this object we have a fixed part of manipulation functions, which is ideally stable for the life-span of the object (basic routines for manipulating attribute values, withdraw / deposit for bank accounts). We call this part the *rigid* part of the object behaviour. The rigid part is typically 'hard-coded' in database applications and realized by optimized code.

Other parts of a database application are subject to frequent change: constraints or business rules, exception activation and notification triggers. Rules for computing interests in a bank application or billing processes are examples of such changing parts. We call these parts *evolving*. These changes may result from changes in business processes and policies, but may even modify the behaviour of single instances of object classes. In current applications, the evolving part may be realized by interpreted SQL triggers and stored procedures, which can be activated, modified or deactivated at runtime.

This situation leads to the notion of 'evolving objects' as building blocks of an information system which are designed for being adaptable to new requirements. Our work starts with the object specification language TROLL, which is based on a temporal logic framework for describing objects as observable processes. As an extension to TROLL, a language is presented which allows to separate the rigid (= stable) part of an object specification from an evolving part. This evolving part is stored in specification attributes which can be manipulated in the same fashion like base attributes storing data values. The evolving part corresponds to run-time interpreted SQL triggers and stored procedures in current database applications. The separation of these aspects of object functionality is a key principle for preparing objects for evolution during the design phase. As basis for maintenance and re-engineering, this separation has to be made explicit in implementations, too.

Heterogeneity and Autonomy in Federated Information Systems

by Stefan Conrad, University of Magdeburg, Germany

In this talk we discuss aspects of heterogeneity and autonomy in federated information systems and sketch basic solutions developed for building federated database systems. Heterogeneity and autonomy cause a lot of problems, in particular in building a federated system from (pre-) existing information systems and in adapting a federated system to changing or new requirements.

Starting from a basic architecture of federated database systems, we first consider the (logical) integration of data managed by already existing information systems. Due to autonomy the data is often described by using different data models, applying different modeling constructs, and thus resulting in heterogeneous (database) schemata.

We give a brief overview of possible integration conflicts occurring during schema integration: semantic conflicts, description conflicts, heterogeneity conflicts, and structural conflicts. Furthermore, we present the basic ideas of the Generic Integration Model (GIM), an approach developed in our group in Magdeburg. GIM provides an intermediate representation for schemata. GIM schemata describe the relationships between disjoint extensions (sets of objects) and attributes as properties of these objects. In order to obtain a GIM schema from an existing database schema an extensional analysis is needed to determine the disjoint extensions. For that, subset relationships (due to specialization) and general overlapping of class extensions must be resolved. Given a GIM schema for the database schemata which are to be integrated we can easily derive an integrated schema. We demonstrate this by applying applying the method of concept analysis based on the mathematical theory of concept lattices.

Furthermore, we emphasize the requirements for dynamic or iterative integration methods. Dynamic integration is needed to adapt the federated system to changing requirements (for instance, schema evolution in component systems; addition of further component systems; unavailability of component systems).

Finally, we briefly discuss the problem of behavior integration. There are some concepts of how to provide a global transaction management for transactions using the integrated schema on the federation layer while local transactions within the component system are preserved. However, heterogeneity and autonomy of the component systems can cause severe conflicts between global and local transactions as well. Semantic integration of be-

havior (e.g. on a design level) is still an open problem which we demonstrate by means of a simple example.

On the Role of Object-Oriented Modeling in Continuous Engineering of Distributed, Heterogeneous Information Systems

by Ralf-Detlef Kutsche, Technical University of Berlin
and Fraunhofer ISST, Berlin, Germany

Building and evolving large-scale information infrastrucutures, the integration and/or federation of distributed heterogeneous information resources (often given by databases, local information systems and a bunch of rather local applications around them) has to be combined with a general view of software evolution, combining forward, reverse and re-engineering steps consistently into a continuous engineering process.

One of the major issues in the development and evolution of information systems typically is the requirement of maintaining legacy components in their old shape and organizational embedding as autonomous systems, although becoming well-integrated members of the whole infrastructure.

These premises given, we suggest a general methodology of continuous information systems engineering based on object-oriented models and their consistent evolution. This approach is the outcome of a number of projects in different application domains, such as information systems for tele-medicine services, and environmental information systems. Technically, we make use of the manifold of techniques in the Unified Modeling Language UML within the object-oriented paradigm for the information systems development, methodically emphasizing on the following tasks, which appear in a multi-cycle development model:

- modeling use cases for the new system to be built;
- modeling information structure of the new system;
- modeling information structure and application contexts of the underlying legacy systems – only as far as required;
- relating the underlying models with the (possibly virtual) new structures;
- modeling functional and behavioural aspects of the legacy systems;

- modeling and relating new functionality with existing functionality by functional modeling and specification;
- modeling and relating new behaviour with existing behaviour;

and, in addition, having two specific essentials in our approach towards a continuous evolution on information infrastructures:

- modeling information and metainformation simultaneously, and relating information/metainformation models with each other;
- developing analysis and design models simultaneously, using the metainformation structures for better coherence between analysis and design models.

Practical experiences show that this OO diagrammatic approach can be successfully applied on different levels of abstraction and in all phases of continuous information systems engineering. However, there still remain a number of open questions w. r. t. semantic coherence of the respective models.

List of Participants

Stefan Conrad
University of Magdeburg, Germany
conrad@iti.cs.uni-magdeburg.de

Wolfgang Deiters
Fraunhofer Institute for Software Engineering and Systems Engineering, Dortmund, Germany
deiters@do.isst.fhg.de

Sebastian Erdmann
Technical University of Berlin, Germany
serdmann@cs.tu-berlin.de
http://cis.cs.tu-berlin.de/~serdmann/

Ric Holt
University of Waterloo, Canada
holt@uwaterloo.ca

Ulrike Kölsch
Forschungszentrum Informatik (FZI), Karlsruhe, Germany
koelsch@fzi.de

Ralf Kramer
Forschungszentrum Informatik (FZI), Karlsruhe, Germany
kramer@fzi.de

Ralf-Detlef Kutsche
Technical University of Berlin
and Fraunhofer Institute for Software Engineering and Systems Engineering (ISST) Berlin, Germany
rkutsche@cs.tu-berlin.de
http://cis.cs.tu-berlin.de
http://www.isst.fhg.de

Hausi A. Müller
University of Victoria, Canada
hausi@csr.uvic.ca
http://www.rigi.csc.uvic.ca

Gunter Saake
University of Magdeburg, Germany
saake@iti.cs.uni-magdeburg.de

Alex Sellink
University of Amsterdam,
The Netherlands
alex@wins.uva.nl

Stefan Tai
Technical University of Berlin, Germany
stai@cs.tu-berlin.de
http://cis.cs.tu-berlin.de/~stai/

Chris Verhoef
University of Amsterdam,
The Netherlands
x@wins.uva.nl

Herbert Weber
Technical University of Berlin
and Fraunhofer Institute for Software Engineering and Systems Engineering (ISST) Berlin/Dortmund, Germany
Herbert.Weber@isst.fhg.de

Kenny Wong
University of Victoria, Canada
kenw@csr.csc.uvic.ca

Steven Woods
Carnegie Mellon University, USA
sgw@sei.cmu.edu