### Contents

**Research Articles**

<table>
<thead>
<tr>
<th>Title</th>
<th>Author(s)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Learning: Issues and Challenges for Information Systems and Technology</td>
<td>RD Quilling, GJ Erwin and O Petkova</td>
<td>5</td>
</tr>
<tr>
<td>A Generic Modelling Framework for Interactive Authoring Support Environments</td>
<td>Paula Kotze</td>
<td>15</td>
</tr>
<tr>
<td>An Information-Theoretic Semantics for Belief Change</td>
<td>T Meyer</td>
<td>33</td>
</tr>
<tr>
<td>A Complexity Metrics Model for Software Correction</td>
<td>A Törn, T Andersson and K Enholm</td>
<td>40</td>
</tr>
<tr>
<td>A Conceptual Design for High-Volume Data Processing of Warehouse Database into Multidimensional Database</td>
<td>Paisarn Trakulsuk and Vichit Avatchanakorn</td>
<td>49</td>
</tr>
<tr>
<td>A Pragmatic Approach to Bitemporal Databases: Conceptualization, Representation and Visualisation</td>
<td>Chiyaba Njovu and WA Gray</td>
<td>58</td>
</tr>
<tr>
<td>A Building Recognition System</td>
<td>SP Levitt and B Dwolatzky</td>
<td>68</td>
</tr>
<tr>
<td>Computer Programming and Learning to Write</td>
<td>John Barrow</td>
<td>77</td>
</tr>
<tr>
<td>Co-operating to Learn using JAD Technologies</td>
<td>TA Thomas</td>
<td>87</td>
</tr>
<tr>
<td>Critical Success Factors for the Implementation of DSS at a Selection of Organisations in Kwazulu/Natal</td>
<td>URF Averweg and GJ Erwin</td>
<td>95</td>
</tr>
<tr>
<td>Enhancing the Predictability of Two Popular Software Reliability Growth Models</td>
<td>Peter A Keiller and Thomas A Muzzuchi</td>
<td>105</td>
</tr>
</tbody>
</table>
Generalised Unification of Finite Temporal Logic Formulas
Scott Hazelhurst ........................................ 110

Harmonizing Global Internet Tax: A Collaborative Extranet Model
E Lawrence and B Garner ................................. 119

Improving Object Oriented Analysis by Explicit Change Analysis
Lui Yu, Siew Chee Kong, Yi Xun and Miao Yuan .................. 128

Reconciling the Needs of New Information Systems Graduates and Their Employers in Small, Developed Countries
Rodney Turner and Glenn Lowry ......................... 136

Shortest Delay Scheduling Algorithm for Lossless Quality Transmission of Stored VBR Video under Limited Bandwidth
Fei Li, Yan Liu, Jack Yiu-Bun Lee and Ishfaq Ahmad ................. 146

Software Croma Keying in an Immersive Virtual Environment
Frans van der Berg and Vali Lalioti ........................ 155

Some Automata-Theoretic Properties of \( \cap \)-NFA
Lynette van Zijl and Andries PJ van der Walt ................. 163

Statistical Analysis of an Automated Computer Architecture Learning Environment
JT Waldron, J Horgan and G Keogh ....................... 168

The CILT Multi-Agent Learning System
Herna L Viktor ............................................. 176

The Development of a Generic Framework for the Implementation of Cheap, Component-Based Virtual Video-Conferencing System
Soteri Panagou and Shaun Bangay .......................... 185

The Role of Experience in User Perceptions of Information Technology: An Empirical Examination
Meliha Handzic and Graham Low ............................ 194

What are Web Sites Used for: Cost Savings, Revenue Generating or Value Creating?
Man-Ying Lee ............................................. 201

New Ideas Papers
Approaches to Video Transmission over GSM Networks
Bing Du and Anthony Maeder ............................... 210

From Information Security Baselines to Information Security Profiles
Rossouw von Solms and Helen van der Haar ................. 215

Grounded Theory Methodology in IS Research: Glaser versus Strauss
J Smit ....................................................... 219

Introducing a Continuum of Abstraction-Led Hierarchical Search Techniques
Robert Zimmer and Robert Holte .......................... 223
<table>
<thead>
<tr>
<th>Title</th>
<th>Authors</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multimedia as a Positive Force to Leverage Web Marketing, with Particular Reference to the Commercial Sector</td>
<td>Stan Shear</td>
<td>229</td>
</tr>
<tr>
<td>Understanding HCI Methodologies</td>
<td>Peter Warren</td>
<td>234</td>
</tr>
<tr>
<td>Electronically Published Papers</td>
<td><a href="http://www.cs.wits.ac.za/~philip/SAICSIT/SAICSIT-99/papers.ideas.html">http://www.cs.wits.ac.za/~philip/SAICSIT/SAICSIT-99/papers.ideas.html</a></td>
<td></td>
</tr>
<tr>
<td>Experience Papers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A Java Client/Server System for Accessing Arbitrary CANopen Fieldbus Devices via the Internet</td>
<td>Dieter Bühler, Gerd Nusser, Gerhard Gruhler and Wolfgang Küchlin</td>
<td>239</td>
</tr>
<tr>
<td>An Object-Oriented Framework for Rapid Client-side Integration of Information Management Systems</td>
<td>Ralf-Dieter Schimkat, Wolfgang Küchlin and Rainer Krautter</td>
<td>244</td>
</tr>
<tr>
<td>Distributed Operating Systems: A Study in Applicability</td>
<td>Jürgen Prange and Judith Bishop</td>
<td>249</td>
</tr>
<tr>
<td>Formal Verification with Natural Language Specifications: Guidelines, Experiments and Lessons so far</td>
<td>Alexander Holt</td>
<td>253</td>
</tr>
<tr>
<td>Introducing Research Methods to Computer Science Honours Students</td>
<td>Vashti Galpin, Scott Hazelhurst, Conrad Mueller and Ian Sanders</td>
<td>258</td>
</tr>
<tr>
<td>Visualising Eventuality Structure</td>
<td>ST Rock</td>
<td>264</td>
</tr>
<tr>
<td>Electronically Published Papers</td>
<td><a href="http://www.cs.wits.ac.za/~philip/SAICSIT/SAICSIT-99/papers.exp.html">http://www.cs.wits.ac.za/~philip/SAICSIT/SAICSIT-99/papers.exp.html</a></td>
<td></td>
</tr>
</tbody>
</table>
SAICSIT'99

South African Institute of Computer Scientists and Information Technologists

Annual Research Conference 17-19 November 1999

Prepare for the New Millennium

Is there life after y2k?

Mount Amanzi Lodge, Hartebeespoort, South Africa

Sponsors

http://www.cs.wits.ac.za/

PHDS research/PHDS.html

Think different.
Preface

Philip Machanick, Overall Chair: SAICSIT’99

Running SAICSIT’99, the annual research conference of the South African Institute for Computer Scientists and Information Technologists, has been quite an experience.

SAICSIT represents Computer Science and Information Systems academics and professionals, mainly those with an interest in research. When I took over as SAICSIT president at the end of 1998, the conference had not previously been run as an international event. I decided that South African academics had enough international contacts to put together an international programme committee, and a South African conference would be of interest to the rest of the world.

I felt that we could make this transition at relatively low cost, given that we could advertise via mailing lists, and encourage electronic submission of papers (to reduce costs of redistributing papers for review).

The first prediction turned out to be correct, and we were able to put together a strong programme committee.

As a result, we had an unprecedented flood of papers: 100 submitted from 21 countries. As papers started to come in, it became apparent that we needed more reviewers. It was then that the value of the combination of old-fashioned networking (people who know people) and new-fashioned networking (the Internet) became apparent. While the Internet made it possible to convert SAICSIT into an international event at relatively low cost, the unexpected number of papers made it essential to find many additional reviewers on short notice. Without the speed of e-mail to track people down and to distribute papers for review, the review process would have taken weeks longer, and it would have been much more difficult to track down as many new reviewers in so little time.

Even so, the number of referees who were willing to help on short notice was a pleasant surprise.

The accepted papers cover an interesting range of subjects, from management-interest Information Systems, to theoretical Computer Science, with subjects including database, Java, temporal logic and implications of e-commerce for tax.

In addition, we were very fortunate in being able invite the president of the ACM, Barbara Simons as a keynote speaker. Consequently, the programme for SAICSIT’99 should be very interesting to a wide range of participants.

We were only able to find place in the proceedings for 36 papers out of the 100 submitted, of which only 24 are full research papers. While this number of papers is in line with our expectation of how many papers would be accepted in each category, we did not have a hard cut-off on the number of papers, but accepted all papers which were good enough, based on the reviews. Final selection was made by myself as Programme Chair, and Derrick Kourie, as editor of the South African Computer Journal. Additional papers are published via the conference web site.

We believe that we have put together a quality programme, and hope you will agree.

Acknowledgments

I would like to thank the South African Computer Journal production team, Andries Engelbrecht and Herna Viktor, respectively from the Department of Computer Science and Informatics, University of Pretoria, for their work on producing the proceedings.

The reviewers listed overleaf did an excellent job: many wrote very detailed reports, sometimes after being called in on very short notice. Inevitably, there were some glitches resulting from the unexpected workload, but the buck stops with the programme chair: I promise to do better next time.

I would also like to thank my own department for putting up with the extra work and expense that running a conference entails. I tried not to burden them with too much extra work, but our secretaries, Zain Gowar and Leanne Reddy, inevitably had to take on some extra work. John Ostrowick provided valuable assistance with design of our web pages and call for papers poster. Carol Kernick, who handles our finances and membership records, did a fine job of keeping up with the demands of the conference.

Finally, I would like to thank our sponsors, whose contribution made this conference been possible:

- PricewaterhouseCoopers – sponsored generous prizes and the conference banquet
- National Research Foundation (NRF) – provided financial support
- University of the Witwatersrand – provided financial support
- Programme for Highly Dependable Systems, University of the Witwatersrand – provided financial support
- Standard Bank – provided financial support
Editorial

• Apple Computer - provided equipment for the conference
• Qualica - provided technical support including helping with the conference web site

Web Site

For more information about SAICSIT, including a pointer to the conference site, see <http://www.saicsit.org.za>.

Referees

• Department of Computer Science, University of Pretoria
  – Derrick Kourie
  – Bruce Watson
  – Vali Lalioti
  – Andries Engelbrecht
  – Ivan Mphahlele
• German National Research Center for Information Technology - GMD
  – Gernot Goebbels
• School of Math Stat and IT, Natal University Pietermaritzburg
  – Peter Warren
• Graduate School of Business, University of Cape Town
  – Kurt April
• CSIR
  – James Jardine
• Department of Computer Science, Rhodes University
  – Shaun Bangay
  – Peter Clayton
  – John Ebden
  – Richard Foss
  – George Wells
  – Peter Wentworth
• Information Technology Division, Rhodes University
  – Caro Watkins
• Department of Information Systems, Rhodes University
  – Brenda Mallinson
  – Dave Sewry
• Department of Informatics, University of Pretoria
  – Carina de Villiers
  – Herna Viktor
  – Niek du Plooy
  – Elsie van Rooyen
  – Machdel Matthee
  – Alan Abrahams
  – Jackie Phahlamohlaka
• School of Information Technology & Engineering, University of Ottawa
  – Stan Szpakowicz
  – Dwight Makaroff
• Department of Information Systems, Victoria University
  – Glenn Lowry
  – Peter Shackleton
  – Tas Adam
  – Alastair Wallace
  – Stephen Burgess
  – Dave Burgess
  – Julie Fisher
  – Jerzy Lepa
  – Geoff Sandy
  – Rod Turner
  – Alastair Wallace
  – Andrew Wenn
• Peninsula School of Computing and Information Technology, Monash University
  – Ainslie Ellis
• School of Information Management and Systems, Monash University
  – Angela Carbone
• School of Computer Science & Software Engineering, Monash University
  – Dianne Hagan
  – Judy Sheard
• School of Management Information Systems, Deakin University
  – John Lamp
  – David Mackay
  – Philip Joyce
- Marie van der Klooster
- Bill Hewett
- Jennie Carroll
- Rodney Carr
- Computer and Automation Research Institute, Hungarian Academy of Sciences
  - Ferenc Vajda
- Department of Computer Science, University of the Witwatersrand
  - Ian Sanders
  - David Lubinsky
  - Conrad Mueller
  - Yinong Chen
  - Bob Baber
  - Vashti Galpin
  - András Salamon
  - Scott Hazehurst
  - Philip Machanick
  - Zoltán Fazekas
- Department of Electrical Engineering, University of the Witwatersrand
  - Barry Dwolatzky
  - Farzin Aghdasi
- Department of Mathematics, University of the Witwatersrand
  - Jonathan Burgess
- School of Law, University of the Witwatersrand
  - Victoria Bronstein
- Edward Nathan & Friedland Attorneys, Johannesburg
  - Justine White
- Department of Information Systems & Technology, University of Durban-Westville
  - Geoff Erwin
- Department of Computer Science, Clemson University, Clemson, SC
  - Karen Hay
-Logical SA
  - Philip Green
- Department of Computer Science, University of Bristol
  - Alan Chalmers
- Department of Systems and Computer Science, Howard University, Washington, D.C
  - Todd Shurn
  - John Trimble
- Department of Computer Science, University of Cape Town
  - Sonia Berman
- King’s College, London
  - Peter Wood
- Department of Information Systems, University of Cape Town
  - Paul Licker
  - Steve Erlank
  - Alemayehu Molla
- Department of Department of Mathematics and Applied Mathematics, University of Cape Town
  - Ingrid Rewitzky
  - Peter Jipsen
  - Renkuan Guo
- Department of Information Systems, University of the Western Cape
  - Andy Bytheway
- Department of Electrical Engineering and Computer Science, The George Washington University, Washington, DC
  - Rachelle Heller
- School of Biophysical Sciences and Electrical Engineering, Swinburne University of Technology
  - Ian Macdonald
- Electrical and Electronic Systems Engineering, Queensland University of Technology
  - Anthony Maeder
- Computer Science Department, Åbo Akademi
  - Aimo Törn
  - Kaisa Sere
  - Mats Aspnas
- Division of Informatics, University of Edinburgh
  - Graeme Ritchie
  - Arturo Espinosa-Romero
  - Jon Oberlander
Editorial

- Chris Brew
- Alexander Holt

• Department of Computer Science and Engineering, Helsinki University of Technology
  - Tommi Junttila
  - Nisse Husberg

• Technical University of Budapest, Department of Measurement and Information Systems
  - Andras Pataricza

• NASA-Goddard Space Flight Center
  - Nigel Ziyad

• Department of Computer Science & Information Systems, University of Port Elizabeth
  - Janet Wesson
  - Leon Nicholls

• Computer Technology Department, Indiana University Purdue University Indianapolis
  - Tim Price

• Department of Electrical Engineering and Computer Science, The University of Michigan
  - Trevor Mudge

• Department of Electrical Engineering and Computer Sciences, University of California, Berkeley
  - David Forsyth

• School of Computer Science, University of Birmingham
  - Mark Ryan

• Faculty of Mathematics, Computer Science, Physics & Astronomy, University of Amsterdam
  - Carlos Areces

• Department of Computer Science, Universidade do Vale do Rio dos Sinos - UNISINOS Rio Grande do Sul
  - Marcelo Walter

• School of Information Technology & Mathematical Sciences, University of Ballarat
  - Binh Pham
An Object-Oriented Framework for Rapid Client-side Integration of Information Management Systems

Ralf-Dieter Schimkat\textsuperscript{a}, Wolfgang Kühlin\textsuperscript{a} and Rainer Krautter\textsuperscript{b}

\textsuperscript{a}Wilhelm-Schickard-Institute for Computer Science, University of Tübingen, Sand 13, D-72076 Tübingen, Germany; \{schimkat, kuechlin\}@informatik.uni-tuebingen.de

\textsuperscript{b}debis Systemhaus Competence Center - Document and Workflowmanagement, Erich-Herion-Straße 13, D-70702 Fellbach, Germany, Rainer.Krautter@debis.com

Abstract

Reuse of software components offers the potential for significant savings in application development costs and time. This paper provides a characterization of reuse at different levels of client-based software in distributed document management systems (DMS). An object-oriented framework called JIMA is established as the basis for clients of a wide range of client-server based DMS. The design of the framework is characterized by multiple architectural layers as well as by a set of standardized interfaces among them. The strict separation of conceptional and implementation level successfully offers the potential of reusing software components within the application domain of DMS. In order to rapidly integrate legacy DMS into JIMA, each component can be adapted and extended separately without interfering with other framework components. Because each user needs a special form of DMS due to individual documents and usage patterns the concept of software bundling at the component level in JIMA is introduced. The JIMA framework can be used via the Internet and as a locally installed standalone application transparently. Several examples of existing commercial DMS at debis Systemhaus Competence Center Document and Workflowmanagement are examined with respect to the framework. The programming language Java is used to implement the object-oriented model.

Keywords: frameworks, software reuse, components, design patterns, java, internet, document management systems


1 Introduction

Document management systems (DMS) are designed to manage documents stored in electronic form. Generally, the term document refers to papers, faxes, electronic mails, photos and video clips. DMS combine a number of different functional components for storing, transparently locating, easily retrieving, and exercising control over documents. Traditionally the software architecture of most legacy DMS is based on the two-tier client-server model where the entire application logic is kept on the client and the server functionality is reduced to execute predefined queries to the document database. However, in this paper we are only concerned with DMS which make use of the three-tier client-server model which separates most of the application functionality in an additional middle layer called application server. In this article we focus on an object-oriented framework for three-tier client-server based DMS. A framework is a reusable, semi-complete application that can be specialized to produce custom applications [2]. It is characterized by an abstract set of classes and interfaces which, taken together, establish a generic software architecture for a family of related applications [7, 8]. Our framework is called JIMA as an abbreviation for Java-ImageMaster-Client where ImageMaster is a DMS developed by debis Systemhaus Competence Center Document and Workflowmanagement (EDMD)\textsuperscript{1}.

We identified the key properties of DMS and set up a general, object-oriented model which consists of several independent architectural layers. Within each layer facilities for reusing software on an object-oriented level are offered. The conglomeration of the layered architecture expands the degree of software reuse to the component level. By applying the concept of design patterns [3] the potential of successful software reuse is extended even further. The rest of the paper focuses on client related issues and uses DMS components synonymously for client-based components in DMS.

The remainder of the paper is organized as follows: Section 2 describes the main problems in the software development process of DMS. Section 3 presents the software architecture of JIMA. In section 4 all necessary steps are described in order to integrate a DMS into JIMA. Section 5 gives an overview of legacy DMS which have been integrated successfully into the framework. The paper concludes with a summary and an outlook.

\textsuperscript{1}EDMD is the document and workflowmanagement group in the Engineering division of debis Systemhaus. debis Systemhaus is the IT subsidiary of debis which in turn is the service company of DaimlerChrysler.
2 Problems to Be Solved

In this section we concentrate on three key problems that need to be solved when constructing and maintaining DMS: The tremendous costs for maintenance, the lack of adequate reuse of software architecture and programming libraries, and the overall software quality.

2.1 Maintenance of legacy DMS

The costs for the entire software development process are primarily dominated by maintenance because application requirements change frequently with respect to the users needs. This often results in a poor documentation of design decisions. Maintaining a set of similar but still different legacy systems separately within a certain application domain such as the DMS, multiplies the costs and the effort of software engineers unnecessarily. Thus establishing a single and more generalized framework for DMS reduces the maintenance costs dramatically. Once such a framework exists, every DMS can make extensive use of already set up and implemented functionalities and services.

2.2 Reusability of existing architecture

Any reusable software component has a context for its use. Assumptions made about the environment and architecture design decisions affect the behavior of the component and can become a complex dependent on the constraints of the original design problem. Simple reuse practice occurs where a software component is lifted from its original context for use and placed in a new context. The component may also require adaption in order to be integrated into the new environment. This adaption may result in incorrect and unpredictable behaviour if the process of reuse is not integrated into a software reuse strategy. A better solution for cost effective reuse is the practice of a software architecture which is explicitly designed for reuse. By controlling the way in which the reusable component can be adapted, the uncertainties of the software component behaviour is reduced. Greater emphasis is placed on the documentation of all assumptions and design decisions.

Many of the existing DMS examined in this paper share a solid number of software components. Customizing each reused component separately would have resulted in a not properly manageable number of different versions of the same component.

2.3 Software Quality

The quality of software systems depends directly on the effectiveness of the software development process. Correctness, reliability, and robustness are considered to be the key issues in software quality [4]. The primary requirement for a high degree of quality is a proper understanding of the entire software system. Thus the adequate documentation of the programming code and all design decisions of the architecture becomes crucial to the overall software quality. Documenting architecture-related decisions becomes particularly important when working in large development teams. By centralizing the documentation process and by establishing a common and easily understandable set of documentation rules the overall software quality of each DMS can be improved.

3 JIMA Framework Architecture

In this section we discuss the main components of the JIMA architecture. In JIMA all client-related issues regarding a three-tier client-server DMS are captured in one single framework. The application logic itself still resides on each server. We have adopted an object-oriented model which is implemented in the programming language Java [6]. By using the concept of design patterns we followed the principle of documenting by designing as proposed in [9].

Figure 1: a) The decoupled architecture of JIMA b) Collaboration in JIMA

The introduction of a layered architecture decouples many domain-specific dependencies and offers flexibility needed in order to extend the framework components independently and to integrate additional DMS into the framework. Each layer can be reused within the application domain. In addition there are reuse facilities of framework components which go beyond the domain of DMS. The concept of configuration policies makes it possible to create a DMS of customized software components.

3.1 Decoupled architecture

The decoupled architecture of the framework has three basic components as depicted in figure 1a: The generic DMS-Model component, a GUI component, and a communication component. The introduction of different framework layers built upon well-defined interfaces facilitates the independent adaptability, extensibility, and maintenance of each component. A component is characterized by its internal functionalities and by its external interfaces. Whereas the internal functionalities of each component are extended and modified rather frequently, their external view is quite stable over time. The generic design of the components interfaces contains no implementation-related issues.
Generic DMS-Model The generic DMS-Model represents a general component for DMS. It captures domain-related services in an abstract manner at the conceptional level as described in [1]. According to the Model-View-Controller paradigm [5] it is a generalized model for each client using the framework. All interaction and collaboration between the framework and all DMS are handled transparently via the DMS-Model. Thus application-centered details are completely hidden from the framework. By using so-called hotspots [10] provided by JIMA, a concrete DMS can be integrated into the framework. Those predefined entry points of the framework have to be subclassed and filled out with application-specific behaviour such as the determination of the communication protocol which is to be used by the framework or the decoding of application-specific entities. Generally each property can be modified at runtime.

Communication component The communication component of JIMA is designed to provide different strategies for executing requests and managing several kinds of services. Each concrete DMS has a so-called communication manager which has its own thread of control. Thus each request to an application server can be sent synchronously or asynchronously depending on the chosen application policy. In JIMA a request is characterized by a high-level and a low-level protocol. Whereas the former determines the type of handshake between client and server, the latter is responsible for delivering the data. Thus several kinds of communication middleware such as low-level sockets or Java Remote Method Invocation (RMI) can be used transparently within JIMA as illustrated in figure 1a. A connector is basically a unified interface for transferring different types of objects. Hence the middleware-related implementation code is masked from the framework. The low-level part of the communication component exposes three subcomponents which are divided into layers according to their functionalities: The top layer keeps an abstract communication handle, whereas the middle one manages the buffering regarding this abstract handle. The concrete communication middleware is located in the third layer which uses the services of the layers above. Placing each functionality in a separate layer provides a potential for reuse of each subcomponent. The connector in figure 1a couples the domain-independent low-level communication component with the domain-specific framework part of JIMA.

GUI component The GUI component manages presentation and interaction with the end-user. According to the Model-View-Controller paradigm, it consists of a set of standard views related to the generic DMS-Model. Each view in the GUI component is bound to the model in order to reflect the current state of the active application graphically. It is characterized by the dynamic setup of its visual components. Reusing the GUI component across different applications provides a consistent graphical layout within the application domain of DMS.

3.2 Collaboration between components

Encapsulating primary functionalities of the domain in different components is only a first step towards a successful framework. In order to accomplish an adequate, domain-related framework, the composition and interoperability of components is necessary without sacrificing each component's internal services.

In JIMA the entire collaboration between the GUI and the communication component is directed by the generic DMS-Model as illustrated in figure 1b. Usually the GUI component calls the generic DMS-Model to initiate application-specific requests which are routed to the communication component. There are two so-called callback strategies for forwarding the results from the communication to the GUI component. Firstly, by passing a predefined callback handle to the communication component, the results can be forwarded directly to the receiver. The implementation is based on the Command-pattern [3] which encapsulates a request as an object without knowing anything about the operation being requested or about the receiver of the request. Secondly, by propagating the results through well-defined notification messages, the number of potential receivers can be even higher. The design of propagated callbacks is based on the Observer-pattern which defines an one-to-many dependency between objects, so that, whenever one object changes state, all its receivers are notified and updated automatically. Whereas the loosely coupling of components increases the reusability, the Observer-pattern maintains consistency between the cooperating components.

3.3 Software bundling and configuration policies

The generic design of the basic framework components offers the flexibility to build customized software bundles rather easily. Application configuration and customization at the component level offers several advantages.

Each configuration policy can be modified at runtime. Therefore, user-related and system-related concerns can be considered by the framework appropriately. Thus the look and feel of the visual layout can be adapted by each user independently. In JIMA each user-defined setting (user profile) is stored in the database.

The framework can act as a standalone application installed locally as well as an internet-based application loaded from the server. The dynamic setup of visual components supports a lower initial downloading time when using the application via the internet because only the logic of the components is delivered but not the actual, graphical layout. The concept of multiple threads of control on the client-side supports the user in using several DMS in parallel as depicted in figure 2. This functionality is quite important for large companies which want to integrate several of their DMS in one single framework.
4 Integrating Document Management Systems into JIMA

Generally, the integration of DMS into JIMA can be accomplished in two ways. Firstly, generate new classes with the required functionalities and add them to JIMA. These classes can be reused by further DMS which might be integrated into JIMA in the future. Secondly, reuse existing classes which were originally created for other DMS. This mechanism speeds up the overall process of integration tremendously since existing classes have already been tested and proven to be valid. Each integrated DMS has to provide its own implementation of the class `InformationSystemProperties` which serves as an uniform interface for accessing OMS-related attributes, as shown in figure 4. The description of each attribute is encapsulated in the class `RechercheEntry`, as shown in figure 3. This class serves as a container for describing an attribute in an uniform way. Each time JIMA accesses an attribute, it asks `RechercheEntry` for the specified meta information regarding the attribute. Basically, a symbolic name, an unique identifier, some default values for retrieval and visualization and the kind of consistency controller, which manages the correct user input, have to be provided for each attribute. As shown in figure 3, a set of predefined types of controllers (DefaultController, NumericController) can be reused or new controller types can be added to JIMA by extending the class Controller. For example, in order to integrate the DMS called BEMI, which is described in section 5, only the application-specific attributes had to be provided in the class `BEMIProperties`.

4.1 Specifying the attributes of a DMS

Each document is characterized by a set of attributes. Whereas the document itself is usually stored in an archive, the attribute values are made persistent in a database. Each DMS has its own model of attributes. In order to integrate a DMS into JIMA, each attribute has to be identified and described. Therefore, each attribute of a DMS in JIMA is an instance of class `RechercheEntry`, as shown in figure 3. This class serves as a container for describing an attribute in an uniform way. Each time JIMA accesses an attribute, it asks `RechercheEntry` for the specified meta information regarding the attribute. Basically, a symbolic name, an unique identifier, some default values for retrieval and visualization and the kind of consistency controller, which manages the correct user input, have to be provided for each attribute. As shown in figure 3, a set of predefined types of controllers (DefaultController, NumericController) can be reused or new controller types can be added to JIMA by extending the class Controller. For example, in order to integrate the DMS called BEMI, which is described in section 5, only the application-specific attributes had to be provided in the class `BEMIProperties`.

4.2 Determining the communication handling

The communication handling between JIMA on the client-side and the application logic on the server-side is determined by the type of communication channel (low-level protocol) and by the type of application-specific handshake protocol (high-level protocol) between client and server. The low-level protocol is responsible for delivering the data between JIMA and the application server. The interface `CommunicationManager` is the manager for establishing, opening and closing connections to the server, as depicted in figure 4. For example, the class `DefaultSocketCommunicationManager` is bound to socket communication. Generally, the communication channel can be switched at runtime without affecting the high-level protocol. A high-level protocol is an application-specific agreement about the semantic interpretation of the delivered content. The class `Request` determines a uniform interface for every high-level protocol in JIMA. In order to integrate a new protocol into JIMA, the class `Request` has to be extended. Then its automatically plugged into the framework and can be processed by the generic `RequestHandler`. The classes `DefaultLogin` and `QueryDocument` are examples of high-level protocols in JIMA as shown in figure 4. Both are used in ZGDOK and BEMI to log into the system and to query documents in the database.
5 JIMA in Practice

In this section we describe three DMS at EDMD which have already been integrated into JIMA. Each of them is characterized by a three-tier client-server architecture and a wide range of different user requirements. In the center of consideration are systems related to the automobile industry as examples of ZGDOK and BEMI. However, NormMaster caters for a broader type of industries.

ZGDOK manages approximately 1.2 million technical documents for constructing Mercedes-Benz automobiles. Key properties of the software architecture are distributed databases with a daily update of servers. About 3000 clients which are installed as standalone applications are connected to ZGDOK world-wide. Up to now about 30 clients are using JIMA as registered users. It is expected that the number of Java-clients will increase in the near future. In 1997 the JIMA framework was established.

BEMI is responsible for managing about 1 million documents about production facilities for constructing Mercedes-Benz automobiles. Key properties of BEMI are the immense volume size of the persistently stored documents and the wide range of different document formats which are delivered from external companies. There are about 40 Java-based clients at this time. For example, the integration of BEMI into JIMA took about three hours for an experienced programmer since the complete communication infrastructure of ZGDOK, which was previously integrated into JIMA, could be reused. Only the attributes of BEMI had to be implemented in the class BEMIProperties, as described in section 4.1.

NormMaster is a DMS for managing technical standards and industry norms for distinct business domains. Thus the structure of the GUI components of NormMaster are rather different from car-related industries. Since norms are unique, they are usually entered and served from a central location. Since they are needed throughout a company, it is especially important that they can be viewed through a browser based system. Companies like DaimlerChrysler, the Bosch Group and the Nokia Group are using NormMaster. Altogether there are approximately 500 clients which take use of JIMA over the internet.

6 Summary and Outlook

In this paper, we have presented the design of an object-oriented, client-based framework that offers the potential of reusing software in DMS successfully. A layered approach provides software reusability at the architectural as well as at the implementation level. The notion of component-based configuration policies is especially important for domain-specific frameworks because they enable a rapid software development process. Our practical experiences have shown that dealing with validation and testing of frameworks is a challenging task because of the higher level of abstraction. However, establishing such a common, abstract software architecture exposes economic potentials that in our case outweigh all drawbacks. The concept of using design patterns systematically has proven to be a valid and successful development strategy.

One aspect of our future work involves an attempt to incorporate DMS-interoperable functionalities into JIMA. Thus end-users have the opportunity to move data between different client-applications. Hence part of the application logic has to be shifted back from the application server to the client-based framework. In addition, reusing software components on the server uniformly is an ongoing research activity at EDMD.

References


[3] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, Massachusetts, 1995.


Acknowledgements

We would like to acknowledge contributions by the members of the development team at EDMD to various aspects of JIMA.
Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research notes. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications of Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript

Manuscripts for review should be prepared according to the following guidelines:

- Use wide margins and 1½ or double spacing.
- The first page should include:
  - the title (as brief as possible)
  - the author's initials and surname
  - the author's affiliation and address
  - an abstract of less than 200 words
  - an appropriate keyword list
  - a list of relevant Computing Review Categories
  - Tables and figures should be numbered and titled.
- References should be listed at the end of the text in alphabetic order of the (first) author's surname, and should be cited in the text according to the Harvard method. References should also be according to the Harvard method.

Manuscripts accepted for publication should comply with guidelines as set out on the SACJ web page, http://www.cs.up.ac.za/sacj, which gives a number of examples.

SACJ is produced using the \LaTeX\ document preparation system, in particular \LaTeX\ 2e. Previous versions were produced using a style file for a much older version of \LaTeX, which is no longer supported.

Please see the web site for further information on how to produce manuscripts which have been accepted for publication.

Authors of accepted publications will be required to sign a copyright transfer form.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect typesetting, reproduction and other costs. Currently, the minimum rate is R30.00 per final page for contributions which require no further attention. The maximum is R120.00, prices inclusive of VAT.

These charges may be waived upon request of the author and the discretion of the editor.

Proofs

Proofs of accepted papers may be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words. Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book Reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000.00 per full page per issue and R500.00 per half page per issue will be considered. These charges exclude specialised production costs, which will be borne by the advertiser. Enquiries should be directed to the editor.