

Boston, Massachusetts
July 9-11, 2007

2007



PROCEEDINGS

SEKE 2007

**The 19th International Conference on
Software Engineering &
Knowledge Engineering**

Sponsored by

Knowledge Systems Institute Graduate School, USA

Technical Program

July 9-11, 2007

Hyatt Harborside Hotel, Boston, Massachusetts, USA

Organized by

Knowledge Systems Institute Graduate School

Copyright © 2007 by Knowledge Systems Institute Graduate School

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher.

ISBN 1-891706-20-9 (paper)

Additional Copies can be ordered from:
Knowledge Systems Institute Graduate School
3420 Main Street
Skokie, IL 60076, USA
Tel:+1-847-679-3135
Fax:+1-847-679-3166
Email:office@ksi.edu
<http://www.ksi.edu>

Proceedings preparation, editing and printing are sponsored by
Knowledge Systems Institute Graduate School

Printed by Knowledge Systems Institute Graduate School

Table of Contents

Foreword	iii
Conference Organization	iv
Keynote	1
<i>Dr. Joe Urban</i>	
Software Engineering Methodology	
Constructing Self-Adaptive Systems with Polymorphic Software Architecture <i>Xiaoxing Ma, Yu Zhou, Jian Pan, Ping Yu, Jian Lu</i>	2
Odyssey-MDA: A transformational approach to component models <i>Natanael E. N. Maia, Ana Paula Terra Bacelo, Claudia M. Werner</i>	9
An Empirical Expository Study on Inferring Developers' Activities from Low-Level Data <i>Irina Coman, Alberto Sillitti (S)</i>	15
TRAP.NET: A Realization of Transparent Shaping in .NET <i>S. Masoud Sadjadi, Fernando Trigos</i>	19
A Framework for Selecting Agile Practices and Defining Agile Software Processes <i>Patricia Vilain, Priscila Basto Fagundes, Thiago Leao Machado (S)</i>	25
A Proposal to Delegate GUI Implementation using a Source Code based Model <i>Marco Monteiro, Paula Oliveira, Ramiro Goncalves (S)</i>	29
Cost-based Analysis of Multiple Counter-Examples <i>Flavian Vasile, Samik Basu</i>	33
Common Coupling as a Measure of Reuse Effort in Kernel-Based Software <i>Liguo Yu, Stephen R. Schach, Kai Chen</i>	39
An Approach to Validating Translation Correctness From SAM to Java <i>Yujian Fu, Zhijiang Dong, Gonzalo Argote-Garcia, Leyuan Shi, Xudong He (S)</i>	45

Software Processes and Engineering Practice

QSEE Project: An Experience in Outsourcing Software Development for Space Applications <i>Valdivino Santiago, Fatima Mattiello-Francisco, Ricardo Costa, Wendell Pereira da Silva, Ana Maria Ambrosio</i>	51
Broadening the Use of Process Patterns for Modeling Processes <i>Hanh Nhi Tran, Bernard Coulette, Bich Thuy Dong</i>	57
A Framework for Tailoring Software Process <i>Lisandra M. Fontoura, Roberto T. Price (S)</i>	63
Analyzing Configuration Management Repository Data for Software Process Improvement <i>Shihong Huang, Christopher Lo (S)</i>	67

Aspect-Based Software Development

Smooth Quality Oriented Component Integration through Product Line Based Aspect-Oriented Component Adaptation <i>Yankui Feng, Xiaodong Liu, Jon Kerridge</i>	71
Modular Specification of Aspect-oriented Systems and Aspect Conflicts Detection <i>Hui Liang, Jing Sun (S)</i>	77
Avoiding Bad Smells in Aspect-Oriented Software <i>Eduardo K. Piveta, Marcelo Hecht, Ana Moreira, Marcelo S. Pimenta, Joao Araujo, Pedro Guerreiro, R. Tom Price</i>	81

Software Testing and Quality Assurance

Metrics of Credibility and Interaction Quality: Design and Evaluation <i>Nilda Perez Otero, Marcelo Perez Ibarra, Sandra Mendez, Adelina Garcia, Ma. del Pilar Galvez Diaz, Viviana E. Quincoces, Hector Liberatori, Beatriz Fiorito, Cecilia Maria Lasserre</i>	87
Predicting Order of Likelihood of Defective Software Modules <i>Rattikorn Hewett, Phongphun Kijsanayothin, Alta van der Merwe</i>	93
Automated Test Code Generation from UML Protocol State Machines <i>Dianxiang Xu, Weifeng Xu, W. Eric Wong</i>	99

Validating A Layered Decision Framework for Cost-Effective Network Defense <i>Huaqiang Wei, Jim Alves-Foss, Du Zhang</i>	105
Toward Modeling and Analysis for Software Installation Testing <i>Jerry Gao, Sujana Tirumalasetti, Chien-Pin Hsu, Yip Cheong, Anne Colendich, Todd Fitch (S)</i>	111
Automatic Test Generation for Database-Driven Applications <i>Zhenyu Dai, Mei-Hwa Chen</i>	117
Fault-Based Testing of Data Schemas <i>Maria Claudia F. P. Emer, Sílvia Regina Vergilio, Mario Jino</i>	123
NLForSpec: Translating Natural Language Descriptions into Formal Test Case Specifications <i>Daniel Leitao, Dante Torres, Flavia Barros</i>	129
Enhanced Random Testing for Programs with High Dimensional Input Domains <i>F.-C. Kuo, K.-Y. Sim, Chang-ai Sun, S.-F. Tang, Zhi Quan Zhou</i>	135
On Test Case Distributions of Adaptive Random Testing <i>Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu (S)</i>	141
Reducing the Number of Test Cases for Performance Evaluation of Components <i>Joco W. Cangussu, Kendra Cooper, Eric Wong</i>	145
Combining Decorated Classification Trees with RCPS Stochastic Models to Gain New Valuable Insights into Software Project Management <i>Antonio Juarez Alencar, Gelson Guedes Rodrigues, Eber Assis Schmitz, Armando Leite Ferreira</i>	151
Towards a Reference Architecture for Software Testing Tools <i>Elisa Yumi Nakagawa, Adenilso da Silva Simao, Fabiano Ferrari, Jose Carlos Maldonado</i>	157
Controlling Restricted Random Testing: An Examination of the Exclusion Ratio Parameter <i>Kwok Ping Chan, T.Y. Chen, Dave Towey (S)</i>	163
An Approach to Software Testing of Machine Learning Applications <i>Christian Murphy, Gail Kaiser, Marta Arias (S)</i>	167

System Requirements Analysis, Modeling and Specification

Agile Methods and Quality Models: Towards an Integration in Requirements Engineering	173
<i>Alexandre Lazaretti Zanatta, Patricia Vilain</i>	
A Framework of Hierarchical Requirements Patterns for Specifying Systems of Interconnected Simulink/Stateflow Modules	179
<i>Changyan Zhou, Ratnesh Kumar, Devesh Bhatt, Kirk Schloegel, Darren Cofer</i>	
In the Requirements Lies the Power	185
<i>Rand Waltzman, Kristina Winbladh, Thomas A. Alspaugh, Debra J. Richardson</i>	
Data and Process Analyses of Data Warehouse Requirements	191
<i>Estella Annoni, Franck Ravat, Olivier Teste</i>	
Requirement Analysis Evolution through Patterns	197
<i>Luca Vetti Tagliati, Roger Johnson, George Rousso (S)</i>	
Automatic Generation of Use Case Diagrams from English Specifications Document	203
<i>Nathalie Rose T. Lim, Christobal T. Cayaba, Joseph Astrophel E. Rodil</i>	
REM4j - A framework for measuring the reverse engineering capability of UML CASE tools	209
<i>Steven Kearney, James F. Power</i>	
FlexUML: A UML Profile for Flexible Process Modeling	215
<i>Ricardo Martinho, Dulce Domingos, Joao Varajao</i>	
A Formal Specification for Product Configuration in Software Product Lines	221
<i>Huilin Ye, Yuqing Lin</i>	
Designing a Platform-Independent Use-Case for a Composite Application using a Reference Architecture	227
<i>Helge Hofmeister, Guido Wirtz (S)</i>	
Synchronization of UML Based Refactoring with Graph Transformation	232
<i>Y. Kosker, A. B. Bener</i>	
Using Formal Composition of Use Cases in Requirements Engineering	238
<i>Rabeb Mizouni, Aziz Salah, Rachida Dssouli</i>	

Web-Based Applications

Real-Time Trust Management in Agent Based Online Auction Systems <i>Rinkesh Patel, Haiping Xu, Ankit Goel</i>	244
Geometric Thumbnails For Web Searching <i>Chris Dunn, Beomjin Kim (S)</i>	251

Web Technology and Web Engineering

ADKwik: Web 2.0 Collaboration System for Architectural Decision Engineering <i>Nelly Schuster, Olaf Zimmermann, Cesare Pautasso</i>	255
Improving Usability of Web Systems with Similar Business Objectives <i>Rashid Ahmad, Zhang Li, Farooque Azam</i>	261
Processing Manipulations of Context Information on the Web <i>Roberto De Virgilio</i>	268
A Tag-Level Web-Caching Scheme for Reducing Redundant Data Transfers <i>Steven E. Cox, Du Zhang, Jinsong Ouyang</i>	274

Software Project and Resource Management

Methodology for Reusing Human Resources Management Standards <i>Asuncion Gomez-Perez, Jaime Ramirez, Boris Villazon-Terrazas</i>	280
An Adaptive Resource Management Approach for a Healthcare System <i>Claudia Raibulet, Luigi Ubezio, Stefano Mussino</i>	286
Study of the Relationships between Personality, Satisfaction and Product Quality in Software Development Teams <i>Marta Gomez, Silvia T. Acuna (S)</i>	292

Software Reuse and Component Technology

Towards Constructing High-available Decentralized Systems via Self-adaptive Components <i>Xi Sun, Li Zhou, Lei Zhuang, Wenpin Jiao, Hong Mei</i>	296
-----------------------------------------------------------------------------------------------------------------------------------------------------------	-----

SAFES: A Static Analysis for Field Security in Java Components <i>Aiyu Shi, Gleb Naumovich</i>	302
Reuse of Database Access Layer Components in JEE Product Lines: Limitations and a Possible Solution (Case Study) <i>Ding Peng, Stan Jarzabek, Damith C. Rajapakse, Hongyu Zhang</i>	308
Design of Wrapper for Self-Management of COTS Components <i>Michael E. Shin, Fernando Paniagua</i>	314
QoS-Optimized Integration of Embedded Software Components with Multiple Modes of Execution <i>Zonghua Gu, Qingxu Deng</i>	320
A C++ Framework for Developing Component Based Software Supporting Dynamic Unanticipated Evolution <i>Andre Rodrigues, Hyggo Almeida, Angelo Perkusich</i>	326
Representing Design Rationale to support Reuse <i>Adriana Pereira de Medeiros, Daniel Schwabe (S)</i>	332
Software System Maintenance	
Telling Stories about System Use: Capturing Collective Tacit Knowledge for System Maintenance <i>Adriana Cristina de Oliveira, Renata Mendes de Araujo, Marcos R.S. Borges</i>	337
Evolution and Runtime Monitoring of Software Systems <i>Hui Liang, Jin Song Dong, Jing Sun</i>	343
On Modern Debugging For Rule-Based Systems <i>Valentin Zacharias, Andreas Abecker</i>	349
Truth Eliciting Mechanisms for Trouble Ticket Allocation in Software Maintenance Services <i>Karthik Subbian, Y. Narahari</i>	355
Knowledge Engineering, Natural Language Processing, and AI	
Graphical Notation for Natural Language and Knowledge Representation <i>Magda G. Ilieva</i>	361

A Hybrid Approach for Natural Language Query Translation <i>Pornpimon Teekayuphun, Ohm Sornil</i>	368
Effective Fault Localization using BP Neural Networks <i>W. Eric Wong, Lei Zhao, Yu Qi, Kai-Yuan Cai, Jing Dong</i>	374
Temporal Software Change Prediction Using Neural Networks <i>Mehdi Amoui, Mazeriar Salehie, Ladan Tahvildari</i>	380
Do Neural-Network Question-Answering Systems Have a Role to Play in the Deployment of Real World Information Systems? <i>Antonio Juarez Alencar, Renata Chaomey Wo, Eber Assis Schmitz, Armando Leite Ferreira</i>	386
Knowledge Conversion in Software Development <i>Olivier Gendreau, Pierre N. Robillard</i>	392
A Language Facilitating Informal Reasoning about Programs <i>J. Nelson Rushton, Dwayne Towell</i>	396

Plenary Talk

Towards Seamless Business Process and Dialogue Specification <i>Dr. Dirk Draheim</i>	402
-----------------------------------------------------------------------------------------------	-----

Database Retrieval Methods

Evaluating the Efficiency of Retrieval Methods for Component Repositories <i>Oliver Hummel, Werner Janjic, Colin Atkinson</i>	404
Benchmarking the RDF(S) Interoperability of Ontology Tools <i>Raul Garcia-Castro, Asuncion Gomez-Perez, York Sure</i>	410
A Deep Classification of Temporal Versioned Integrity Constraints for Designing Database Applications <i>Robson Leonardo Ferreira Cordeiro, Renata de Matos Galante, Nina Edelweiss, Clesio Saraiva dos Santos (S)</i>	416
Generating Linear Temporal Logic Formulas for Pattern-Based Specifications <i>Salamah Salamah, Vladik Kreinovich, Ann Q. Gates (S)</i>	422

Ontology Based Classification Generating Method for Browsing-Based Component Retrieval	428
<i>Ge Li, Lu Zhang, Bing Xie, Weizhong Shao (S)</i>	

Data

A Context-Dependent Semantic Distance Measure	432
<i>Ahmad El Sayed, Hakim Hacid, Djamel Zighed</i>	
A Semantical Change Detection Algorithm for XML	438
<i>Rodrigo Cordeiro dos Santos, Carmem Hara</i>	
XML Schema Evolution by Context Free Grammar Inference	444
<i>Julio C. T. da Silva, Martin A. Musicante, Aurora T. R. Pozo, Silvia R. Vergilio</i>	

Software Development and Design Pattern

Software Tradeoff Assistant: An Integrated Framework for Analytical Decision Making and Tradeoffs in Software Development	450
<i>Ratikorn Hewett, Vikram Patankar</i>	
Improving Separation of Concerns in the Development of Scientific Applications	456
<i>S. M. Sadjadi, J. Martinez, T. Soldo, L. Atencio, R. M. Badia, J. Ejarque (S)</i>	
Pattern-based J2EE Application Deployment with Cost Analysis	462
<i>Nyun Zhang, Gang Huang, Ling Lan, Hong Mei (S)</i>	
Exploratory Design of Derivation Business Rules Using Query Rewriting	467
<i>Roman Krenicky, David Willmor, Suzanne M. Embury (S)</i>	
Classification of Design Pattern Traits	473
<i>Jing Dong, Yajing Zhao (S)</i>	

Data Warehouse

A Proposal for a Conceptual Data Warehouse Quality Model	477
<i>Manuel Serrano, Rafael Romero, Jose-Norberto Mazon, Juan Trujillo, Mario Piattini</i>	
Integrating Complex Data into a Data Warehouse	483
<i>F. Rava, O. Teste, R. Tournier, G. Zurfluh (S)</i>	

Data Mining and Machine Learning

- Learning from Software Quality Data with Class Imbalance and Noise
Andres Folleco, Taghi M. Khoshgoftaar, Jason Van Hulse, Chris Seiffert 487

System and Software Architecture

- Architectural Elements Recovery and Quality Evaluation to Assist in Reference Architectures Specification
Aline Pires Vieira de Vasconcelos, Claudia Maria Lima Werner 494
- EvoSpaces: 3D Visualization of Software Architecture
Sazzadul Alam, Philippe Dugerdil 500
- Ontobrowse: A Semantic Wiki for Sharing Knowledge about Software Architectures
Hans-Jorg Happel, Stefan Seedorf 506

Applications

- Building Business Considerations into Enterprise Application Designs
Rattikorn Hewett, Aashay Thipse 513
- Incremental effort prediction models in Agile Development using Radial Basis Functions
Raimund Moser, Witold Pedrycz, Giancarlo Succi (S) 519
- BASS: Business Application Support through Software Services
Mateus B. Costa, Rodolfo F. Resende, Marcelo V. Segatto, Eduardo F. Nakamura, Nahur Fonseca 523

Model-Driven Software Development

- Using Model-Driven Pattern Matching to derive functionalities in Models
Ignacio Garcia-Rodriguez de Guzman, Macario Polo, Mario Platini 529
- A Model-driven Approach to Architecting Secure Software
Ebenezer A. Oladimeji, Sam Supakkul, Lawrence Chung (S) 535

Agent-Based Technology and Intelligence

An Intelligent Agent of Automatically Notify Services <i>Shuo-Yan Hsu, William C. Chu</i>	541
A Proposal for a Decentralized Multi-Agent Architecture for Virtual Enterprises <i>Andreas Grunert, Sven Kaffille, Guido Wirtz</i>	546
Traceability for Agent-Oriented Design Models and Code <i>Gilberto Cysneiros, Andrea Zisman</i>	552
ONTOMADEM: An Ontology-driven Tool for Multi-Agent Domain Engineering <i>Rosario Girardi, Adriana Leite</i>	559
A Three Level Multi-agent Architecture to Foster Knowledge Exchange <i>Juan Pablo Soto, Aurora Vizzaino, Javier Portillo-Rodriguez, Mario Piattini (S)</i>	565
An Agent Based System for Search in Distributed Environments <i>Li Sa, Yong-Sheng Ding (S)</i>	570

DB Access and Query Processing

Tree Hash Under Concurrency Control <i>Kyosuke Yasuda, Takao Miura</i>	574
Query Processing in Paraconsistent Databases in the Presence of Integrity Constraints <i>Navin Viswanath, Rajshekhar Sunderraman</i>	580
An Object-Oriented Approach to Storage and Retrieval of RDF/XML Documents <i>Ching-Ming Chao</i>	586
CXPath: a Query Language for Conceptual Models of Integrated XML Data <i>Diego de Vargas Feijo, Claudio Naoto Fuzitaki, A Ivaro Moreira, Renata de Matos Galante, Carlos Alberto Heuser</i>	592

Service-Oriented Technology and Web Technology

OWLed: Extending Knowledge for Web Ontology Language <i>Hichem Zait, Aicha Mokhtari</i>	598
Using Ontologies to Represent Software Project Management Antipatterns <i>Dimitrios Settas, Ioannis Stamelos</i>	604

Service Composition Using Planning and Case-Based Reasoning <i>Kuan-Hsian Huang, Alan Liu</i>	610
MDA-based Ontology Development: A Study Case <i>Eluzai Souza dos Santos, Celia Ghedini Ralha, Hervaldo Sampaio Carvalho</i>	616
Towards Domain-Centric Ontology Development and Maintenance Frameworks <i>Faezeh Ensan, Weichang Du (S)</i>	622
Service Oriented Architecture Empirical Study <i>Mohammad Abu-Matar, Jeff Offutt (S)</i>	628
Semantic Support to Reformulate Public Services in Terms of Life Events <i>Luis Alvarez Sabucedo, Luis Anido Rifon (S)</i>	632
A Component-Based Solution and Architecture for Dynamic Service-Based Applications <i>Alessio Colzi, Tommaso Martini, Paolo Nesi, Davide Rogai (S)</i>	637
System Reliability and Verification	
Adequacy of Composite Parametric Software Reliability Models <i>Lance Fiondella, Swapna S. Gokhale</i>	643
Evaluation of the OORT Techniques for Inspection of Requirements Specifications in UML: an empirical study <i>Tereza G. Kirner, Erik R. da Cruz (S)</i>	649
Agent Modeling/Methodology	
Adjudicator: A Statistical Approach for Learning Ontology Concepts from Peer Agents <i>Behrouz Far, Abdel-Halim Hafez Elamy, Nora Houari, Mohsen Afsharchi</i>	654
Agent Applications	
DALICA: Intelligent Agents for User Profile Deduction <i>Stefania Costantini, Leonardo Mostarda, Arianna Tocchio, Panagiota Tsintza (S)</i>	660

Human Interaction and GUI Development

- An Approach to Multimodal Input Interpretation in Human-Computer Interaction
Fernando Ferri, Patrizia Grifoni, Stefano Paolozzi 664
- Sketch Style Recognition in Human Computer Interaction
Daniilo Avola, Fernando Ferri, Patrizia Grifoni 670
- Human-Computer Interaction for a Novel Arm-wrestling Robot
Chul-goo Kang, Ho-yeon Kim (S) 676

Service

- Managing XML Versions and Replicas in a P2P Context
Deise de Brum Saccol, Nina Edelweiss, Renata de Matos Galante, Carlo Zaniolo 680
- Knowledge sharing through a simple release planning method for web application development
Sven Ziemer, Ilaria Canova Calori 686
- Distributed BPEL Processes
Luciano Baresi, Andrea Maurino, Stefano Modafferi 692
- SAM: Semantic Advanced Matchmaker
E.S. Ilhan, G.B. Akkus, A. B. Bener 698
- A development platform for distributed user interfaces
Anders Larsson, Magnus Ingmarsson, Bo Sun (S) 704

Security

- A Dynamical System Approach to Intrusion Detection Using System Call Analysis
Nitin Kanaskar, Remzi Seker, S. Ramaswamy 710
- Multi-level Anomaly Detection with Application-Level Data
Swapna S. Gokhale, Jijun Lu 718

Grid Technology

- A Four-layered Semantic Grid Architecture
Celia Ghedini Ralha, Jose Nelson C. Allemand, Alba C. M. Melo 724

Software Metrics, Measurement and Evaluation

Performance Analysis of the Active Object Pattern in Middleware <i>Paul J. Vandal, Swapna S. Gokhale, Aniruddha S. Gokhale</i>	730
Analyzing the Applicability of a Theoretical Model in the Evaluation of Functional Size Measurement Procedures <i>Nelly Condori-Fernandez, Oscar Pastor (S)</i>	736
Software Documents: Comparison and Measurement <i>Tom Arbuckle, Adam Balaban, Dennis K. Peters, Mark Lawford (S)</i>	740

Industrial Workshop

Workflow Management and Service Oriented Architecture <i>Theodorich Kopetzky, Dirk Draheim</i>	749
Implementing Agile Development - More than Changing Methodology <i>Chuck Fredrick</i>	751
Knowledge Modelling using UML <i>A. J. Rhem</i>	755
Reviewers' Index	758
Authors' Index	760

Note: (S) means short paper.

A Proposal for a Conceptual Data Warehouse Quality Model

Manuel Serrano, Rafael Romero, José-Norberto Mazón, Juan Trujillo, and Mario Piattini

Abstract—Data warehouses are considered a cornerstone of a decision support system, since they provide the adequate information resources for decision making in an integrated manner. Therefore, the quality of a data warehouse is a key issue in any business organization, and several works have pointed out the importance of measuring the quality in data warehouse domain. However, to the best of our knowledge, the current approaches that deal with the quality aspects of data warehouses only consider isolated quality metrics, without being part of a comprehensive quality model that allows designers to assess the quality of a data warehouse in a systematic and objective way. Having considered this limitation in current data warehouse research, in this paper we present a quality model for the data warehouse domain. This model allows designers to state the meaning of quality in the data warehouse domain, thus measuring the quality of a data warehouse conceptual model in a systematic and objective way. We have used a well-known methodology to build this quality model, and we have related the elements of this quality model to our previously defined and validated quality metrics for conceptual models of data warehouses.

Index Terms—Data Warehouse, Quality, Quality Model

I. INTRODUCTION

Nowadays, data warehouses have become one of the most crucial components of decision support systems, since they efficiently and effectively allow the integrated management of data in order to provide information resources that support effective problem and opportunity identification, critical decision-making, and strategy formulation, implementation, and evaluation. Therefore, data warehouse technology has become an adequate solution, providing such resources in an environment where increasing competition, sophisticated and informed costumers, unpredictable market

Manuscript received March 8, 2007. This work was supported in part by the CALLIA Project supported by the University of Castilla – La Mancha and the DIMENSIONS Project (PBC-05-012-1) supported by Consejería de Educación y Ciencia – Junta de Comunidades de Castilla – La Mancha. José-Norberto Mazón is funded by the Spanish Ministry of Education and Science under a FPU grant (AP2005-1360)

Manuel Serrano (corresponding author, Phone: +34 926295300) and Mario Piattini are with the Alarcos Research Group, University of Castilla – La Mancha, Ciudad Real, Spain (e-mail: Manuel.Serrano@uclm.es and Mario.Piattini@uclm.es).

Rafael Romero, José-Norberto Mazón and Juan Trujillo are with the Department of Software and Computing Systems, University of Alicante, Alicante, Spain (e-mail: romero@dlsi.ua.es, jnmazon@dlsi.ua.es and jtrujillo@dlsi.ua.es).

fluctuations, and changing regulatory environments are putting much pressure on business organizations [1].

As a result of this importance, companies are increasing their investment in the development of data warehouses. In fact, the investment in this kind of systems is around several millions a year and increases by 20% per year [2]. However, despite the potential of data warehouses and the huge amount of money invested, success is not necessarily guaranteed. As stated by several authors, a data warehouse project is a real risk [3] and more than 60% of data warehouses do not meet the user expectations [4]. Therefore, designers have to deal with quality issues to avoid these pitfalls and guarantee the success of the data warehouse project [5].

Dealing with quality issues is not an easy task, since quality is an abstract and subjective aspect, for which there is no universal definition: it is usually said that there is a quality definition for each person. In this way, it is very complex to measure or assess the quality of a software product in an objective way. Several guides have been proposed to address the complexity of data warehouse quality, but most of the time, guides are not enough, since they can help designers in their work, but they imply rather subjective decisions, and this can lead to “not-so-good” products.

In order to overcome this inherent subjectivity of quality in data warehouse projects, in this work we introduce a quality model that helps designers to assess the quality of a data warehouse in an objective way, thus guaranteeing success in designing a good data warehouse. In this paper, we focus on the quality of conceptual data warehouse models and we present the first steps towards building a data warehouse quality model. We focus on the quality of the conceptual models, as a *good* design may (or may not) lead to a *good* system, but a *bad* design will surely render a *bad* product of low quality. Thus, it is important to focus on the quality of the product from the first steps of its development, i.e. the conceptual model.

The remainder of this paper is structured as follows. The next section shows related work regarding data warehouse quality and models. In Section 3, a method for developing quality models based on ISO 9126 [6] is presented. In Section 4, the method is applied to the building of a data warehouse quality model. The last section presents some conclusions and future work arising from this work.

II. RELATED WORK

From the beginnings of data warehousing, quality has been an important issue, from the perspectives of both data and

schema. This interest in quality is motivated by the importance of the information quality in the decision support systems.

Several techniques have been used in an attempt to improve the quality of data warehouses, such as software process improvement models. Among these techniques we can find the Capability Maturity Model [7] or SPICE [8], which relies on the belief that "good processes deliver good software", but unfortunately good processes do not guarantee good software. Good processes only increase the probability that good software may be obtained.

In this search for quality, several quality standards have been applied to data warehouse development (such as ISO 9126 [6] and IEEE 1061 [9]), usually complemented by GQM techniques [10, 11]) to attempt to measure the quality of the data warehouse. These standards are too general and complex and are usually not concrete enough to be useful in assessing the quality of the system.

With regard to data warehouse quality measurement we can find several proposals for measuring some quality dimensions of data warehouses such as the metrics proposed by [12] and [5]. These proposals are good approaches to data warehouse measurement but they are not complete, since they are not part of a quality model that allows designers to use them in a systematic and objective way.

Lastly, we can find the proposal of a data warehouse quality model, named DWQ [13] which attempted to assure the quality of the data stored inside the data warehouse to improve the data warehousing experience; this project had some quality dimensions and focused on data quality. We think that data warehouse quality should not only be assessed in terms of data quality and that schema quality is as important an issue as data quality. In this paper we propose a quality model for conceptual data warehouse models, based on the ISO 9126 standard.

III. QUALITY MODEL DEVELOPMENT METHOD

Developing a quality model is not an easy task. Several aspects should be considered and the whole process should be carried out in an objective and methodological way. For this purpose, Franch and Carvallo [14] proposed a method to build quality models based on the ISO 9126-1 standard [6]. This method is now briefly described.

The proposed methodology comprises of six steps and also considers a preliminary activity (step 0). A graphical view of the methodology is shown in Figure 1. Although the steps seem to be sequential, these steps can be intertwined and repeated if needed.

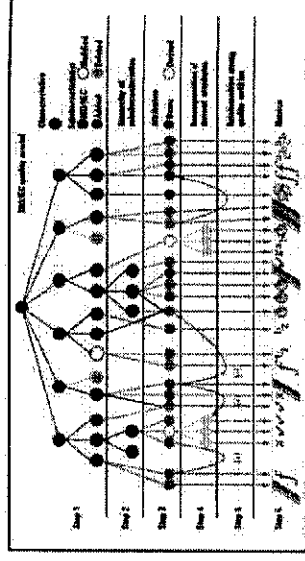


Fig. 1. Franch and Carvallo [14] six steps methodology

Step 0. Defining the domain

Previous to the application of the methodology, it is necessary to carefully examine and describe the domain of interest, with the help of experts. In order to describe the domain the use of conceptual modelling to keep track of relevant concepts is recommended.

Step 1. Determining quality subcharacteristics

The decomposition of characteristics into subcharacteristics that appear in the ISO standard is quite reasonable and should be used unless good reasons for not doing so emerge during domain analysis. In this case, new subcharacteristics are added, some of them are refined and it is even permissible to remove some of the subcharacteristics that do not apply to the domain.

Step 2. Defining a hierarchy of subcharacteristics

Typically, further decomposition of subcharacteristics with respect to some factors is needed. In this way, we obtain a hierarchy of subcharacteristics.

Step 3. Decomposing subcharacteristics into attributes

Quality subcharacteristics provide a comprehensible abstract view of the quality model. But we need to decompose these abstract concepts into more concrete ones - the quality attributes. An attribute keeps track of a particular observable feature of the packages in the domain. Attributes should be precisely defined to clarify the underlying quality concepts that they represent and to link them with the appropriate subcharacteristics.

As the standard itself mentions, it is not possible, from a practical point of view, to measure all the subcharacteristics in the entirety of a software product, but we can create a complete list of those which are most relevant.

Step 4. Decomposing derived attributes into basic ones

Some of the attributes obtained in Step 3 can be directly measured, but others may be so abstract that they require more decomposition. In this way, attributes are classified as being 'derived' and 'basic'. Derived attributes should be decomposed until they are completely expressed in terms of those which are basic.

Step 5. Stating relationships between quality entities

If we wish to obtain a complete quality model, we must state the relationships between quality entities. The model becomes more exhaustive and the implications of quality requirements become clearer.

We can classify the relationships into three types:

- Collaboration: Growing the first quality entity implies growing the second quality entity.
- Damage: Growing the first quality entity implies decreasing the second quality entity.
- Dependency: Some values of the first quality attribute require that the second one fulfills certain conditions.

Step 6. Determining metrics for attributes.

It is not only necessary to identify the attributes but also to select metrics for all the attributes. For this task, we can use the general theory of metrics.

IV. DATA WAREHOUSE QUALITY MODEL

In this section, we explain how to apply the method described in the previous section to develop a quality model for data warehouse domain. We have developed each of the proposed steps, and we have even related our previously defined metrics to this quality model.

Step 0. Defining the domain

In this step we define the domain for which we want to build a quality model, in this case the conceptual data warehouse schemata. Next, we outline an approach to data warehouse conceptual modeling, based on the UML (Unified Modeling Language). This approach has been specified by means of a UML profile¹ which contains the stereotypes necessary to carry out conceptual modeling successfully [15]. This profile contains the stereotypes which are necessary to elegantly represent main multidimensional (MD) properties at the conceptual level (see Table 1). Specifically, the structural properties of MD modeling are represented by means of a UML class diagram in which the information is clearly organized into facts and dimensions. These facts and dimensions are respectively represented by Fact and Dimension classes. Fact classes are defined as composite classes in shared aggregation relationships of n Dimension classes. The minimum cardinality in the role of the Dimension classes is 1 to indicate that every fact must always be related to all the dimensions. A fact is composed of measures or fact attributes. These are represented as attributes with the FactAttribute stereotype. By default, all measures in the Fact class are considered to be additive. For non-additive measures, additive rules are defined as constraints and are included in the Fact class. Furthermore, derived measures (indicated by /) and their derivation rules can also be explicitly represented as bagged values of a FactAttribute.

¹ A profile is a set of improvements that extends an existing UML type of diagram to a different use. These improvements are specified by means of the extensibility mechanisms provided by UML (stereotypes, properties and restrictions) in order to be able to adapt it to a new method or model.

Our approach also allows the definition of degenerate dimensions, thereby representing other fact features in addition to the measures for analysis. These degenerated dimensions are represented as stereotyped attributes of the Fact class (DegenerateDimension stereotype).

The many-to-many relationships between a fact and a specific dimension are specified by means of the cardinality 1..n in the role of the corresponding Dimension class. In this case, we usually need to describe specific attributes to provide further features for every instance combination in this particular relationship. In doing so, the provided attributes are usually called degenerate facts. These degenerate facts are represented as an association class attached to a many-to-many aggregation relationship between a Fact class and a Dimension class. This DegenerateFact class may contain FactAttributes and DegenerateDimensions.

With respect to dimensions, each level of a classification hierarchy is specified by a Base class. Every Base class may contain several dimension attributes (DimensionAttribute stereotype), an identifying attribute (OID stereotype), and must also contain a Descriptor attribute (D stereotype). An association (represented by a stereotype called Rolls-UpTo) between Base classes specifies the relationship between two levels of a classification hierarchy. The only prerequisite is that these classes must define a Directed Acyclic Graph (DAG) rooted in the Dimension class (the DAG constraint is defined in the stereotype Dimension). The DAG structure can represent both multiple and alternative path hierarchies. A Dimension class contains a unique first hierarchy (or dimension) level called a terminal dimension level. A roll-up path is a subsequence of dimension levels, which starts in this terminal level (lower level of detail) and ends in an implicit level (not graphically represented) that represents all the dimension levels.

We use roles to represent the way in which the two Base classes see each other in a Rolls-UpTo association: role R represents the direction in which the hierarchy rolls-up, whereas role D represents the direction in which the hierarchy drills-down. Moreover, we use roles to detect and avoid cycles in a classification hierarchy, and therefore, to help us to achieve the DAG condition.

Due to the flexibility of UML, we can also consider non-strict hierarchies (an object at a hierarchy's lower level belongs to more than one higher-level object) and complete hierarchies (all members belong to one higher-class object and that object consists of those members only). These characteristics are specified, respectively, by means of the cardinality of the roles of the associations and by defining the stereotype Completeness in the association between Base classes. Lastly, the categorization of dimensions is considered by means of the generalization/specialization relationships of UML.

Our profile is formally defined and uses the Object Constraint Language (OCL) to express well-formed rules of the new defined elements (see Table 1), thereby avoiding its arbitrary use. We refer the reader to [15] for a further explanation of this profile and its corresponding OCL constraints.

TABLE 1.
MAIN STEREOTYPES OF OUR UML PROFILE FOR MD MODELING OF DW.

Stereotype	Notation
Fact	
Dimension	
Base	
DegenerateFact	
FactAttribute	
DegenerateDimension	
DimensionAttribute	
OID	
Descriptor	
Rolls-UpTo	
Completeness	

Step 1. Determining quality subcharacteristics

In this step the quality characteristics are decomposed into subcharacteristics following the ISO 9126 standard and the characteristics of the domain we are working with.

The ISO 9126 proposes the decomposition shown in Table 2.

TABLE 2.
ISO/IEC 9126-1 CHARACTERISTICS AND SUBCHARACTERISTICS

Characteristics Functionality	Subcharacteristics
Reliability	Suitability Accuracy Interoperability Security
Usability	Maturity Fault Tolerance Recoverability Understandability Learnability Operability Attractiveness
Efficiency	Time behaviour Resource utilization
Maintainability	Analyzability Changeability Stability Testability
Portability	Adaptability Installability Coexistence Replaceability
Usability	Suitability Accuracy Security Understandability Learnability
Maintainability	Analyzability Changeability Stability

In our domain, data warehouse conceptual models, we firmly believe that some of the dimensions of the ISO 9126 model are not applicable, since the quality of a conceptual data warehouse schema is related to the correctness, completeness, understandability, security and stability of that schema. Those dimensions that are not related to these characteristics have been removed. In Table 3 we can find the subcharacteristics that are taken into account after removing those that are not applicable to our domain.

TABLE 3. FINAL SET OF QUALITY SUBCHARACTERISTICS

Characteristics Functionality	Subcharacteristics
Usability	Suitability Accuracy Security Understandability Learnability
Maintainability	Analyzability Changeability Stability

Step 2. Defining a hierarchy of subcharacteristics

In our context (quality of conceptual data warehouse models), there is no need to decompose the subcharacteristics stated in the previous step into smaller ones.

Step 3. Decomposing subcharacteristics into attributes

As we have previously stated, quality subcharacteristics provide a comprehensible abstract view of the quality model, but they are too abstract and difficult to measure. In this step, the abstract concepts are decomposed into attributes. For each relevant characteristic and subcharacteristic we provide a set of attributes that are related to that characteristic (see Table 4).

TABLE 4. ATTRIBUTES RELATED TO EACH QUALITY SUBCHARACTERISTIC

Characteristics Functionality	Subcharacteristics	Attribute
Usability	Suitability	Adequacy of data modelled to requirements Completeness Well-Specified Semantic correctness
	Accuracy	Accuracy to requirements
	Security	Presence of security constraints Adequacy of security rules
Learnability	Understandability	Ease of understanding Schema Complexity Schema Size Schema Depth
	Learnability	Ease of understanding Schema Complexity Schema Size Schema Depth
	Analyzability	Ease of understanding Schema Complexity Schema Size Schema Depth Coupling Cohesion Normalization Modularity
Changeability	Changeability	Ease of understanding Schema Complexity Coupling Cohesion Normalization Modularity
	Stability	Schema Complexity Coupling Cohesion Normalization Modularity
	Stability	Schema Complexity Coupling Cohesion Normalization Modularity

Step 4. Decomposing derived attributes into basic ones

After Step 3 we may find that some attributes are too abstract to be easily measured and we should decompose them into more concrete ones. However, the attributes presented in Table 4 are concrete enough, so it is therefore unnecessary to further decompose them.

Step 5. Stating relationships between quality entities

In this step we state the relationships between quality entities, so the model becomes more exhaustive and its implications become clearer. Table 5 shows the relationships between quality attributes. In this table 'COL' means Collaboration, 'DEP' means Dependency and 'DMG' means Damage.

TABLE 5. RELATIONSHIPS BETWEEN QUALITY ATTRIBUTES

	Completeness		Well-Specified		Semantic correctness		Accurate to requirements		Presence of security constraints		Ease of understanding		Schema Complexity		Schema Size		Schema Depth		Coupling		Cohesion		Normalization	
	Adequacy of data modelled to requirements	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL	COL
Completeness	COL																							
Well-Specified	COL	COL																						
Semantic correctness	COL	COL	COL																					
Accurate to requirements	COL	COL	COL	COL																				
Presence of security constraints	DEP	COL	COL	COL	COL																			
Adequacy of security rules	COL	COL	COL	COL	COL																			
Ease of understanding		DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG
Schema Complexity		DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG
Schema Size		DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG
Schema Depth		DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG	DMG
Coupling																								
Cohesion																								
Normalization																								
Modularity																								

For example in Table 5, we can see that the Modularity collaborates with the Ease of understanding because if the system is modular it is probably easier to understand. This table is a summary of all the relationships between the quality attributes.

Step 6. Determining metrics for attributes.

After identifying the attributes, we must assign metrics to each attribute. Table 6 shows the proposed metric for each attribute.

TABLE 6. METRICS ASSOCIATED TO EACH QUALITY CHARACTERISTIC.

Characteristics	Subcharacteristics	Attribute	Metric
Functionality	Suitability	Adequacy of data modelled to requirements	Number of redundant requirements
		Completeness	Number of requirements mapped into the model
Accuracy	Well-Specified	Well-Specified	Number of missing requirements
		Accurate to requirements	Number of missing attributes
		Accurate to requirements	Number of missing elements
		Accurate to requirements	Number of missing relationships
		Accurate to requirements	Number of wrong specifications
Modularity	Semantic correctness	Semantic correctness	Number of redundant specifications
		Semantic correctness	Number of requirements mapped into the schema
		Semantic correctness	Adequacy to requirements
		Semantic correctness	Number of missing requirements
		Semantic correctness	Number of elements that are not represented in the requirements
Usability	Understandability	Understandability	Number of requirements mapped into the schema
		Understandability	Ratio of names relationships
		Understandability	Ratio of relationships
		Understandability	Ratio of relationships
		Understandability	Ratio of relationships
		Understandability	Ratio of relationships
		Understandability	Ratio of relationships
		Understandability	Ratio of relationships
		Understandability	Ratio of relationships
		Understandability	Ratio of relationships
Learnability	Analyzeability	Analyzeability	Number of classes
		Analyzeability	Number of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships
Maintainability	Analyzeability	Analyzeability	Number of classes
		Analyzeability	Number of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships
		Analyzeability	Ratio of relationships

Cohesion		Schema cohesion
Changeability	Normalization	Normalization complaint
	Modularity	Number of modules Ratio of modules
	Ease of understanding	Number of classes Number of relationships Ratio of relationships
	Schema Complexity	Adequacy of names Number of relationships Ratio of relationships
	Coupling	Schema coupling
	Cohesion	Schema cohesion
	Normalization	Normalization complaint
	Modularity	Number of modules Ratio of modules
	Ease of understanding	Number of classes Number of relationships Ratio of relationships
	Schema Complexity	Adequacy of names Number of relationships
	Coupling	Schema coupling
	Cohesion	Schema cohesion
	Normalization	Normalization complaint
	Modularity	Number of modules
	Stability	Ratio of modules

REFERENCES

- [1] B. Shin, "An Exploratory Investigation of System Success Factors in Data Warehousing," *Journal of Association for Information Systems*, vol. 4, pp. 141-170, 2003.
- [2] T. Chenoweth, D. Schuff, and R. St. Louis, "A method for developing Dimensional data marts," *Communications of the ACM*, vol. 46, pp. 93-98, 2003.
- [3] P. Vassiliadis, "Galiliver in the land of data warehousing: practical experiences and observations of a researcher," presented at International Workshop on Design and Management of Data Warehouses (DMDW2000), Stockholm (Sweden), 2000.
- [4] C. Stedman, "Warehousing Projects Hard to Finish," *Computerworld*, vol. 32, pp. 29, 1998.
- [5] N. Prat and S. S.-S. Cherfi, "Multidimensional Schemas Quality Assessment," presented at The 15th Conference on Advanced Information Systems Engineering (CAISE '03) Workshops, Klagenfurt/Velden (Austria), 2003.
- [6] ISO/IEC, "9126-1: Software Engineering - Product quality - Part 1: Quality model," 2001.
- [7] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber, "The Capability Maturity Model for software," Software Engineering Institute, Carnegie Mellon University CMU/SEI-93-TR-024, February 2003 1993.
- [8] K. El Emam, J. N. Drouin, and W. Melo, "Spice: The Theory and Practice of Soft-ware Process Improvement and Capability Determination," IEEE Computer Society 1998.
- [9] IEEE, "IEEE Std 1061-1998 IEEE Standard for a Software Quality Metrics Methodology," 1998.
- [10] V. Basili and D. Weiss, "A Methodology for Collecting Valid Software Engineering Data," *IEEE Transactions on Software Engineering*, vol. 10, pp. 728-738, 1984.
- [11] V. Basili and H. Rombach, "The TAME project: towards improvement-oriented software environments," *IEEE Transactions on Software Engineering*, vol. 14(6), pp. 728-738, 1988.
- [12] M. Serrano, "Definition of a set of metrics for assuring data warehouse quality," Univeristy of Castilla - La Mancha (Spain), 2004.
- [13] M. Jarke, M. Lenzertini, Y. Vassiliou, and P. Vassiliadis, *Fundamentals of Data Warehouses*, second edition ed: Springer-Verlag., 2002.
- [14] X. Franch and J. P. Carvallo, "Using Quality Models in Software Package Selection," *IEEE Software*, vol. 20, pp. 34-41, 2003.
- [15] S. Luján-Mora, J. Trujillo, and L.-Y. Song, "A UML profile for multidimensional modeling in data warehouses," *Data & Knowledge Engineering (DKE)*, vol. 59, pp. 725-769, 2006.

V. CONCLUSIONS AND FUTURE WORK

Nowadays, data warehouses have become an important software product used in decision support systems. Due to the environment in which data warehouses are used, assuring the quality of these systems is a crucial issue. When dealing with data warehouse quality, it is important to assess the quality of both data and schemas, as the quality of the data and also the whole system is affected by the quality of the model that supports them.

In order to assess the quality of the data warehouse schema, it is necessary to define a quality model in order to help designers in assuring the quality of the final system as it is being developed in a systematic and objective way.

In this paper we have reviewed a method for creating a quality model based on the ISO 9126 standard and we have applied its six steps to create a first proposal of a conceptual data warehouse quality model.

As a result of this method we have obtained a set of important quality characteristics which are broken down into sub-characteristics and quality attributes. As a final step we have proposed a set of initial metrics for assessing those quality attributes, and we have related them to our previously defined and validated metrics.

The next step in this process of proposing a quality model is to validate it. We are planning to begin with the validation of the proposed metrics in order to assure their usefulness and adequacy. After, we believe that we should attempt to validate the whole quality model in an industrial environment.