Model-Driven Traceability in Healthcare Information Systems Development

Ståle Walderhaug [a+b], Gunnar Hartvigsen[b] and Erlend Stav[a]

[a] *Health Informatics Group, SINTEF ICT, Trondheim, Norway*
[b] *Medical Informatics and Telemedicine Group, Department of Computer Science, University of Tromsø, Norway*

Abstract

To improve the quality of software used in healthcare information systems, traceability can play an important role. The concept of traceability establishes explicit trace links in the design, development and maintenance processes, keeping documentation complete and updated. Trace information allows validating bodies, domain experts, system designers and programmers to easily navigate along artefact dependencies and perform simple traceability analysis such as coverage and change impact. This paper presents a novel solution for traceability applied in model-driven development for services in a distributed healthcare environment. The results demonstrate the feasibility of explicitly modelling dependencies using a formal language such as UML. Based on the experience from implementing two full-scale homecare systems in the EU-IST MPOWER project, the potential improvements and challenges with a traceability solution are discussed.

Keywords:

Continuity of care, service oriented architecture, model-driven development, MDA, UML, homecare

Introduction

The software developed for use in healthcare systems should not fail during execution, and ideally, it should have no errors or flaws. However, testing can only reveal errors and not guarantee flawlessness. Organisations such as the Food and Drug Administration (FDA) require a strict validation process before approving software for use with medical devices. Factors such as documentation of the software itself and the development process become important in the validation and testing of software. In [1], FDA describes some general validation principles that they consider important for the validation process. A core principle is related to traceability: "A traceability analysis should be conducted to verify that the software design implements all of the software requirements" (page 19). Furthermore, it states that "a source code traceability analysis should be conducted and documented to verify that: each element of the software design specification has been implemented in code; modules and functions implemented in code can be traced back to an element in the software design specification and to the risk analysis;"(page 21).

Traceability is a concept where the relationships between system artefacts are explicitly described to become a part of documentation, as well as direct input to software development phases and system analysis. With the advances of model-driven software development, traceability has evolved into a concept that includes all system artefacts, from initial mission documents, through requirements, design, tests, deployments, and to operational system versions. Traceability can be used for trace link navigation, coverage, orphan and change impact analysis [2], also know as the core traceability services.

Studies have shown that traceability is considered useful and may have a positive impact on development and maintenance of software [3]. During development and maintenance work, readily access to information about which features that are implemented, why (rationale) and which dependencies that exist between different features and components, is vital for correct (valid) implementation. Arisholm et al have found that "for complex tasks and past a certain learning curve, the availability of UML documentation may result in significant improvements in the functional correctness of changes as well as the quality of their design. However, there does not seem to be any saving of time." [4]

To explore how a traceability solution can be implemented within today's development paradigm, the EU-IST MPOWER project[1] has designed a traceability enabled model-driven development methodology and applied it in development of 25 healthcare specific software services. This paper presents the traceability methodology and results focusing on the core traceability services. Using Unified Modeling Language (UML) [5] as the main notation for domain modelling, requirements modelling, system design and system development, explicit traceability links were created and used for dependency navigation and analysis. Based on the experience from the development project, an extended traceability solution is discussed.

Methods and materials

Development methodology

As described in [6], the developers in the MPOWER project applied a model-driven development approach where *user*

[1] http://www.mpower-project.eu

scenarios developed by *domain stakeholders* (dementia experts, patients, patients' family and caregivers) were incorporated into a UML model for software *service* designs. A total of 137 stakeholders were involved in the requirements development phase: 62 senior citizens (22 in Netherlands, 40 in Poland); 11 Family carers of persons with dementia (5 in Austria, 6 in Norway); 49 Healthcare professionals (all in Poland); and, 15 Dementia experts (4 in Austria, 11 in Norway). The 18 scenarios (2-pages each) constitute the requirements in the domain needs, and is main input to the software service development process. In accordance with the SOA4HL7 Methodology [7], recommendations in [8] and OMG's Model Driven Architecture (MDA) [9], a complete service development process was designed as shown in Figure 1. The development is structured into the three MDA phases.

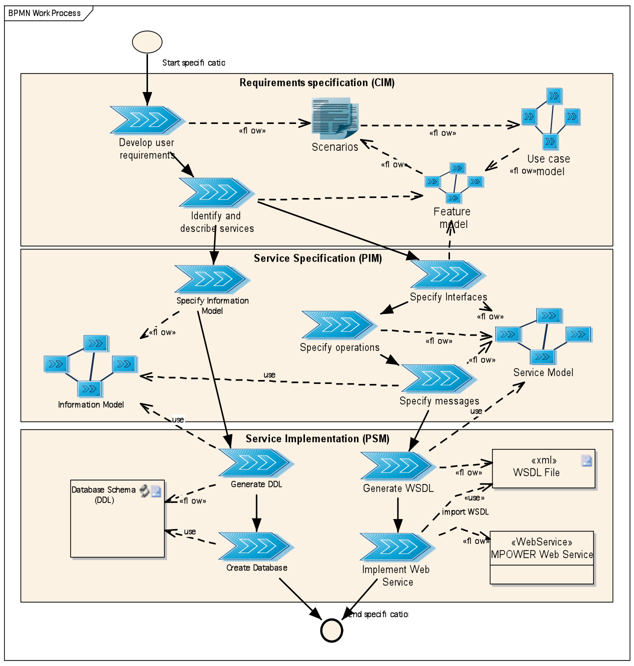


Figure 1- Development process and artifacts

Traceability model

In the development methodology, explicit and implicit trace links are established between the core development artefacts:

- Scenario ↔ Use case: implicit link by documenting the scenarios in each use case
- UseCase ↔ feature: explicit link using UML Dependency stereotyped with <<*trace*>>.
- Feature ↔ Service Model: explicit trace link between a set of features and a service. Implicit from feature to information model through the design of Service Messages (request/response)
- Service Provider ↔ WSDL file. Inserted feature traces as <*wsdl:documentation*> elements in the file.
- Documentation: direct export of navigable (HTML) models and text (RTF) from the design model.

The metamodel for explicit trace links uses the properties of a UML dependency, and includes trace link name/alias, source artefact name, target artefact name, free text field and, source and target roles.

Traceability Services

The trace information is stored within the design model and is used by the core traceability services [2][11]:

1. *Trace navigation*: from any traced artefact (modelling element), navigate along (trace) dependencies back and forth.
2. *Coverage and orphan analysis*: query trace data for a list of traced elements and un-traced elements. Mostly used to find requirements that are not fulfilled (coverage) and elements that have no dependencies to other elements (orphans)
3. *Change impact analysis*: query trace data for information about which elements that will be directly or indirectly affected by a change in a specific element. Used to estimate cost of change requests.

A sound development methodology should support all these services.

Results

A total of 25 software services were designed using UML, realizing 168 features that were derived from 50 use cases, which involved 16 different actors and described 60 sub-activities from the 18 scenarios. 16 developers were involved in the design of the services during a 10-month period, following the methodology described in Figure 1. The complete actor and service model is reported in [6].

Trace links in the Patient Management Service

To demonstrate the traceability results, the Patient Management service is used as an example. Figure 2 shows the "Stakeholder management" use case and how it is related to actors and other use cases.

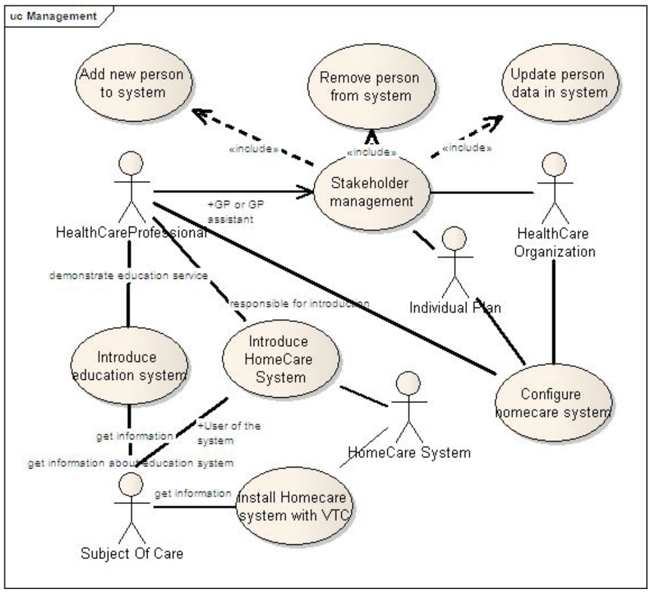

uc Management
Add new person to system
Remove person from system
Update person data in system
«include»
«include»
«include»
Stakeholder management
HealthCareProfessional
+GP or GP assistant
HealthCare Organization
Individual Plan
demonstrate education service
responsible for introduction
Introduce education system
Introduce HomeCare System
Configure homecare system
get information
+User of the system
get information about education system
HomeCare System
get information
Install Homecare system with VTC
Subject Of Care

Figure 2- Use Cases for Management scenarios

The use case diagram shows an overview of the system "context", and more details can be found when looking at the properties for each use case element. The relationship from the *use case* to the *scenarios* is documented as a property of the use case element. A use case can be traced to more than one scenario description.

From the initial scenarios and use cases, a set of features are derived. A feature represents a high-level requirement for the system, and each *feature* is directly related to one or more *use cases* as shown for the Stakeholder Management use case in Figure 3. Using the use cases and features as input, the domain information model is designed and services identified following the process recommended by the OMG/HL7 Healthcare Service Specification Project in [7] and Erl in [8].

Each service design has a service rationale that traces from feature to service in a separate diagram. The features to be realized by the service provide useful information about the operations that are required on the services' interface, e.g., feature *"Get System user information for the person"* is implemented as the operations *getUserForPerson()* and *getUserForPatient()*.

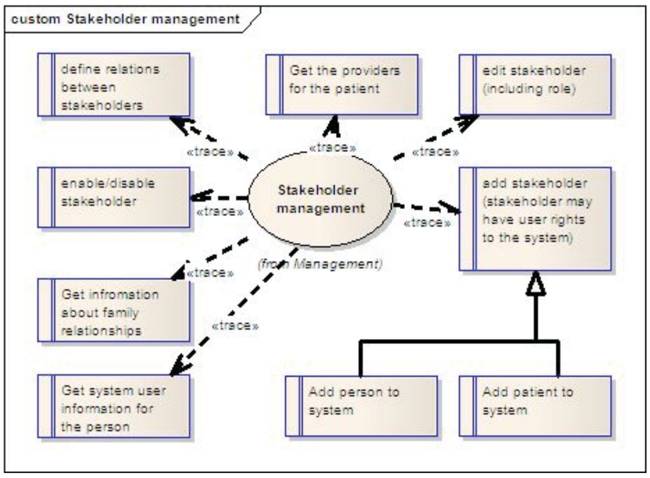

custom Stakeholder management
define relations between stakeholders
Get the providers for the patient
edit stakeholder (including role)
enable/disable stakeholder
Stakeholder management
(from Management)
add stakeholder (stakeholder may have user rights to the system)
Get infromation about family relationships
Get system user information for the person
Add person to system
Add patient to system
«trace»

Figure 3- Features derived from the Stakeholder management use case

Traceability Services

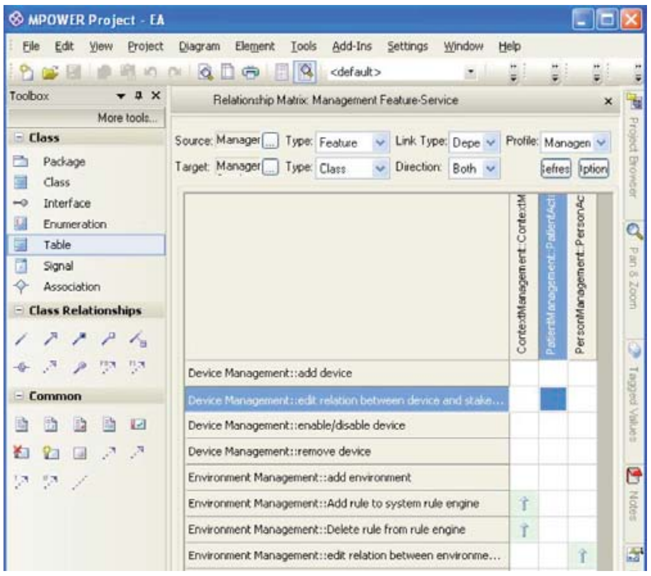
From the models presented above, it is possible to provide the three core traceability services.

Trace Navigation

To navigate along (trace linked) elements one can use the modelling tool itself (e.g. Sparx Enterprise Architect), or an exported version of the model using any standard html browser (e.g. Opera, Firefox, Safari). Each model element is described with properties, appearance in diagrams and relationships to other model elements.

Traceability Matrix: Coverage and Orphan Analysis

To perform *coverage and orphan analysis*, most UML tools provide a relationship matrix query where the source, target, relationship type and direction can be used as query parameters. The result is a matrix showing which elements that have a relationship, indicated with a green arrow as shown in Figure 4. From the matrix it is possible to get an overview of which services that realizes which features. If a feature is not realized by any service, this indicates that not all features are *covered*, and a thorough inspection should be conducted. Similarly, if a service does not realize any of the features in the design model, it is an indication of an *orphan service*. A special situation can be identified if two services realize the same set of features.



Relationship Matrix: Management Feature-Service

Source: Manager Type: Feature Link Type: Depe Profile: Managen
Target: Manager Type: Class Direction: Both

	ContextManagement::ContextM	PatientManagement::PatientActi	PersonManagement::PersonAc
Device Management::add device			
Device Management::edit relation between device and stake...			
Device Management::enable/disable device			
Device Management::remove device			
Environment Management::add environment			
Environment Management::Add rule to system rule engine	↑		
Environment Management::Delete rule from rule engine	↑		
Environment Management::edit relation between environme...			↑

Figure 4- Coverage and orphan analysis matrix for management services and features

Change Impact Analysis

This analysis service can give product owners, administrators and project leaders a qualified estimate on the cost of making a change to the system design, based on trace links in the design model. A query to the design model can find all artefacts (services, features, actors, use cases, etc.) that have some dependency to the proposed system design change. Using the semantics in the model along with the experience from implementation, a qualified estimate on cost, risk factors and required man-hours can be made. Figure 5 shows a simple change impact visualization diagram. From the Patient Management service it is possible to visually trace the artefacts that are directly or indirectly dependent on its design. E.g., replacing the *Individual Plan* system with another system will require a reimplementation of five *features*, that each has a property value stating its difficulty.

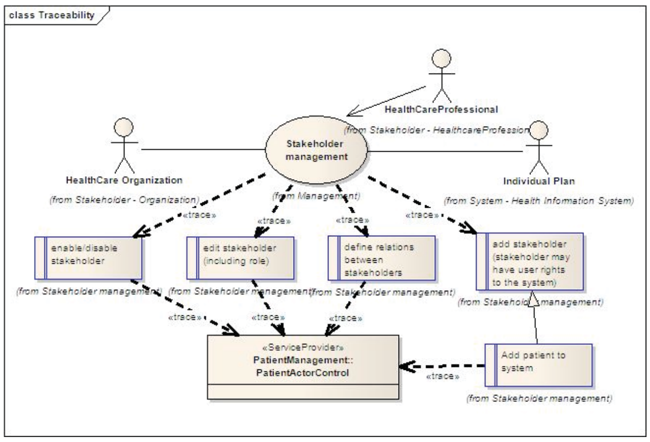

class Traceability
HealthCareProfessional
(from Stakeholder - HealthcareProfession
Stakeholder management
(from Management)
HealthCare Organization
(from Stakeholder - Organization)
Individual Plan
(from System - Health Information System)
enable/disable stakeholder
(from Stakeholder management)
edit stakeholder (including role)
(from Stakeholder management)
define relations between stakeholders
(from Stakeholder management)
add stakeholder (stakeholder may have user rights to the system)
(from Stakeholder management)
«ServiceProvider» PatientManagement:: PatientActorControl
Add patient to system
(from Stakeholder management)
«trace»

Figure 5- Example of model based view of traceability. All dependencies from between the actors, use cases, features and the final software service are shown in one diagram

Discussion

Using model-driven development and traceability solutions may improve documentation quality, and provide valuable information for many stakeholders in the design, development and maintenance phases of a system's lifecycle. However, a complete set of traceability services are not inherently incorporated in model-driven development and must be enforced by a design and development methodology. The proposed methodology offers these traceability services by using stereotyped UML dependency associations in implicit and explicit models.

Developer effort

For the developers, only minor effort is required to create the trace links. The view-based concept [13] used in MDD ensures that *model elements* are reused across *views (or diagrams)*, providing consistency and persistence of relationships. As a general principle, the more information that is incorporated in the trace data, the more advanced analysis services can be executed [2]. Nevertheless, for simple analysis services as is demonstrated herein, no additional trace information must be provided besides what is already in the model elements. Navigation in models using a html browser, reviewing relationships matrices for coverage / orphan analysis, and creating "query diagrams" for simple change impact analysis, are services that can be easily provided and used during design, development and maintenance using the proposed tools and methodology.

Benefits for healthcare information system development

To take full advantage of the trace information, the development methodology must incorporate both creation and inspection of trace information. An evaluation in the MPOWER project showed that traceability was especially useful for the developers [3].

For approval of software for medical devices, FDA strongly recommends traceability as a tool. The analysis services associated with traceability are considered powerful for improving the quality and documentation of the software. Furthermore, maintenance of legacy systems is a complex and costly process [15][16][17], also in the healthcare domain. Most hospitals have one or more systems that were implemented and put into production in the eighties. These systems are subject to maintenance to respond to new architectures, updates in standards, vocabularies and nomenclatures. There are many different standards and they are continuously being revised and new versions are ratified and made public many times during a system's lifetime.

Traceability services could be useful for managing maintenance processes.

Relationship to other approaches

Since 2004, the European Conference on Model Driven Architecture has organized a workshop on Traceability[2]. The conference papers reports on successful traceability projects. Also related to traceability is the increased use of business process modelling and simulation of care processes in modelling tools. IBM's Business Driven Development in Healthcare approach uses business process models and trace links to conduct advanced analysis and simulation [19].

Another model-based tool that utilizes traceability services are the three layer graph-based meta modelling tool (3LGM[2])tool from University of Leipzig, Germany [20]. The primary objective of this tool is "to describe, evaluate and plan health information systems." The 3LGM[2] tools could be used in the methodology described in the Figure 1, but in view of the fact that the 3LGM[2] metamodel is different from UML, the model elements cannot be reused in the succeeding system design and development phases.

Another model-driven healthcare software development process is the HL7 Development Framework [21]. The current version describes an approach were dependencies between artefacts are explicitly being modelled in UML. However, at the time of writing, the framework does not incorporate any traceability services.

Extending the traceability information

In terms of software management and maintenance, the meta information for trace link information play an important role, especially for the change impact analysis. Using UML as the core modelling language, one way to *extend expressiveness of trace links* is to use stereotypes and tagged values in a UML Profile. A stereotype can have tagged values such as implementation difficulty, importance level, creation date, creator and version dependencies. In addition, a profile can refine the graphical presentation of model elements and associations. Figure 6 shows an example diagram where green arrows indicate easy and red are critical/expensive implementation.

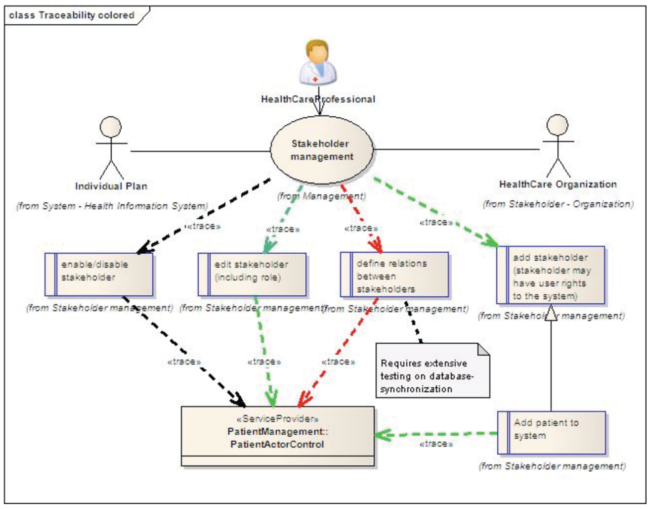

class Traceability colored
HealthCareProfessional
Stakeholder management
Individual Plan
(from System - Health Information System)
(from Management)
HealthCare Organization
(from Stakeholder - Organization)
«trace»
«trace»
«trace»
«trace»
enable/disable stakeholder
(from Stakeholder management)
edit stakeholder (including role)
(from Stakeholder management)
define relations between stakeholders
(from Stakeholder management)
add stakeholder (stakeholder may have user rights to the system)
(from Stakeholder management)
«trace»
«trace»
«trace»
Requires extensive testing on database-synchronization
«ServiceProvider» PatientManagement:: PatientActorControl
«trace»
Add patient to system
(from Stakeholder management)

Figure 6- Example of a stereotyped change impact analysis diagram

In a larger system design, the colour coding of trace links (red for critical, black normal and green loose coupling), explicit notes and stakeholder icons (as shown in Figure 6) would make the visual analysis process more effective.

Problems with implementation of traceability

There are some significant problems in implementing full traceability in software system development, as is discussed in

[2] http://www.ecmda-fa.org

[2][22][23]. The main problems are related to trace information sharing between tools and trace semantics. It is however possible to provide a partial traceability solution with existing tool interfaces and metamodels, and extensions could be provided from standardization organizations such as OMG, HL7 or even the FDA to further enrich the trace information database.

Concluding remarks

Traceability information can improve system development and maintenance processes. The core traceability services illustrated by Walderhaug et al [2] can be provided with a design methodology that utilizes the built-in UML dependency mechanism in a UML modeling tool. The results presented herein demonstrate that relevant and updated documentation can be made available to all stakeholders involved in a system's lifecycle.

References

[1] U.S. Department Of Health and Human Services, Food and Drug Administration Center for Devices, and Radiological Health Center for Biologics Evaluation and Research, "General Principles of Software Validation; Final Guidance for Industry and FDA Staff," 2002.

[2] S. Walderhaug, E. Stav, U. Johansen, and G. K. Olsen, "Traceability in Model-driven Software Development," in *Designing Software-Intensive Systems - Methods and Principles*, P. F. Tiako, Ed. Hersey, PA: IGI Global, Information Science Reference, 2008, pp. 133-160.

[3] S. Walderhaug, M. Mikalsen, I. Benc, G. Loniewski, and E. Stav, "Factors affecting developers' use of MDSD in the Healthcare Domain: Evaluation from the MPOWER Project," in *From code-centric to model-centric develpoment, Workshop at European Conference on Model-Driven Architecture*, Berlin, Germany, 2008.

[4] E. Arisholm, L. C. Briand, S. E. Hove, and Y. Labiche, "The impact of UML documentation on software maintenance: An experimental evaluation," *IEEE Transactions on Software Engineering,* vol. 32, pp. 365-381, 2006.

[5] Object Management Group (OMG), "UML 2.1.2 Superstructure and Infrastructure," Object Management Group (OMG)2007.

[6] S. Walderhaug, E. Stav, and M. Mikalsen, "Reusing models of actors and services in smart homecare to improve sustainability," *Studies in health technology and informatics,* vol. 136, pp. 107-12, 2008.

[7] A. Honey and B. Lund, "Service Oriented Architecture and HL7 v3: Methodology," HL7 Service Oriented Architecture Special Interest Group (SOA SIG)November 10, 2006 2006.

[8] T. Erl, *Service-Oriented Architecture Concepts, Technology, and Design*. Crawfordswille, Indiana, USA: Prentice Hall, 2006.

[9] J. Miller and J. Mukerji, "MDA Guide Version 1.0.1," Object Management Group (OMG) omg/2003-06-01, 2003-06-13 2003.

[10]CEN TC251, "EN 13940-1: Health Informatics - System of Concepts to Support Continuity of Care - Part 1: Basic Consepts," European Committee for Standardization, September 26 2006.

[11] N. Aizenbud-Reshef, B. T. Nolan, J. Rubin, and Y. Shaham-Gafni, "Model Traceability," *IBM Systems Journal,* vol. 45, pp. 515-526, 2006.

[12] S. Johnston, "UML 2.0 Profile for Software Services." vol. 2009: IBM, 2005.

[13] International Telecommunication Union, "ITU-T Rec. X.906|ISO/IEC 19793: Information technology - Open distributed processing - Use of UML for ODP system specifications," ITU, 2004.

[14] Standish Group International, "The Chaos Report," Standish Group International1994.

[15] A. Eastwood, "Firm fires shots at legacy systems," *Computing Canada,* vol. 19, p. 17, 1993.

[16] L. Erlikh, "Leveraging legacy system dollars for e-business," *IT professional,* vol. 2, pp. 17-23, 2000.

[17] C. Huff, "Elements of a realistic CASE tool adoption budget," *Communications of the ACM* vol. 35, 1992.

[18] J. Moad, "Maintaining the competitive edge," *Datamation,* pp. 61-62,64,66, 1990.

[19] J. Wang and A. Asthana, "BPM Enabled SOA to Maximize ROI for Healthcare Transformation."

[20] A. Winter, B. Brigl, G. Funkat, A. Huber, O. Heller, and T. Wendt, "3LGM2-Modeling to support management of health information systems," *International Journal of Medical Informatics,* vol. 76, pp. 145-150, 2007.

[21] L. Coller and N. Daoust, "HL7 Development Framework," 2009.

[22] N. Aizenbud-Reshef, R. F. Paige, J. Rubin, Y. Shaham-Gafni, and D. S. Kolovos, "Operational Semantics for Traceability," in *European Conference on Model Driven Architecture - Traceability Workshop 2005*, Nuremberg, Germany, 2005.

[23] J. Champeau and E. Rochefort, "Model Engineering and Traceability," in *UML 2003 SIVOES-MDA Workshop*, San Francisco, California, USA, 2003.

Address for correspondence

Ståle Walderhaug
Department of Computer Science
University of Tromsø, 9000 Tromsø, Norway
Mobile: +47 90766069
Email: stale.walderhaug@sintef.no