FASSI 2015

The First International Conference on Fundamentals and Advances in Software Systems Integration

ISBN: 978-1-61208-448-0

August 23 - 28, 2015

Venice, Italy

FASSI 2015 Editors

Chris Ireland, Open University, UK

Petre Dini, Concordia University, Canada / China Space Agency, China

FASSI 2015

Foreword

The First International Conference on Fundamentals and Advances in Software Systems Integration (FASSI 2015), held between August 23-28, 2015 in Venice, Italy, dealt with software system integration.

Despite a legacy of projects over decades and the likelihood of continued if not increased connectivity between software systems in the future, there is little by way of a sound theory as to the cause of the problems of software integration and how we might address them.

On the surface the question of how to integrate two software systems appears to be a technical concern, one that involves addressing issues, such as how to exchange data (Hohpe 2012), and which software systems are responsible for which part of a business process. Furthermore, because we can build interfaces between software systems we might therefore believe that the problems of software integration have been solved. But those responsible for the design of a software system face a number of trade-offs. For example the decoupling of software components is one way to reduce assumptions, such as those about where code is executed and when it is executed (Hohpe 2012). However, decoupling introduces other problems because it leads to an increase in the number of connections and introduces issues of availability, responsiveness and synchronicity of changes (Hohpe 2012).

The objective of this conference was to work towards understanding of these issues, the trade-offs and the problems of software integration and to explore strategies for dealing with them.

We take here the opportunity to warmly thank all the members of the FASSI 2015 Technical Program Committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to FASSI 2015. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the FASSI 2015 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that FASSI 2015 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the field of software system integration.

We are convinced that the participants found the event useful and communications very open. We hope Venice provided a pleasant environment during the conference and everyone saved some time for exploring this beautiful city.

FASSI 2015 Chairs:
Chris Ireland, Open University, UK

FASSI 2015

Committee

FASSI 2015 Chair

Chris Ireland, Open University, UK

FASSI 2015 Technical Program Committee

Hany Ammar, West Virginia University, USA
Marco Autili, University of L'Aquila, Italy
Christian Bird, Microsoft Research, USA
Bara Buhnova, Masaryk University, Czech Republic
Graeme Burnett, Xcordis Fintech, UK
Haipeng Cai, University of Notre Dame, USA
Danilo Caivano, University of Bari, Italy
Ip-Shing Fan, Cranfield University, UK
Fabio Fioravanti, University of Chieti-Pescara, Italy
Matthias Galster, University of Canterbury, New Zealand
Anup Gupta, Cognizant Technology Solutions (CTS), UK
Ibrahim Habli, University of York, UK
Alan Hayes, University of Bath, UK
Vladimir Itsykson, St. Petersburg State Polytechnic University, Russia
Foutse Khomh, Ecole Polytechnique de Montréal, Canada
Chris Lokan, UNSW Canberra, Australia
Carol Long, Advanced Computer Software Plc, UK
Fergal McCaffery, Lero - Irish Software Research Centre | Dundalk Institute of Technology, Ireland
Richard Mordinyi, Vienna University of Technology, Austria
Henry Muccini, University of L'Aquila, Italy
Marc Novakouski, Software Engineering Institute, USA
Ipek Ozkaya, Carnegie Mellon SEI, USA
Tarmo Ploom, Credit Suisse - Zurich, Switzerland
Dewayne E. Perry, University of Texas at Austin, USA
Patricia Roberts, University of Brighton, UK
Philip Ross, Endava Ltd, London, UK
Richard Selby, Northrop Grumman, USA
Massimo Tivoli, Università di L'Aquila, Italy
Tayssir Touili, Paris Diderot University - Paris 7, France
Gunter Saake, Otto-von-Guericke University Magdeburg, Germany
Mauro Santoro, University of Milano - Bicocca, Italy
Corrado Aaron Visaggio, University of Sannio, Italy

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Towards a Metrics Model for DevOps,

Results of a Case Study in an Industrial Company

Jos Trienekens
University of Technology,
Faculty of Industrial Engineering and Innovation Sciences
Eindhoven, The Netherlands
email: j.j.m.trienekens@tue.nl

***Abstract*—Recently in the software industry, a methodology called DevOps has emerged, which aims at the integration of software development and deployment (i.e., operations/maintenance) to improve the performance of the overall software process. DevOps contributes to the multi-dimensional problem of software integration, approaching this problem from an organizational point of view. DevOps originates from lean and agile methodologies and stresses the improvement of the entire process flow, overall product quality improvement based on customer feedback. This paper presents a case study at Philips IT The Netherlands on the implementation of DevOps, in particular on the iterative identification and specification of a metrics model to monitor the effectiveness of DevOps.**

Keywords-DevOps, agile; organizational integration; metrics; case study.

I. INTRODUCTION

Philips IT is a centralized IT organization servicing three business domains, respectively Healthcare, Lighting and Consumer Lifestyle. Within IT, there exist two large parties: IT Delivery, where development projects are planned and executed, and IT Infrastructure & Operations (I&O), which is responsible for the implementation and the daily operations. The latter includes maintenance and control of the IT systems, e.g., providing (helpdesk) support. Delivery has been adopting SCRUM methods over the last three years and their software development methods and techniques become increasingly agile [2], [3]. Currently, there are over 100 SCRUM teams. These teams are multidisciplinary and collaborate with relevant partners on both a business and a technical level. Partners are located across the world, thus collaboration in the SCRUM teams takes place virtually. While Delivery has adopted agile methodologies, I&O has been working in accordance with the Information Technology Infrastructure Library framework, ITIL [4]. Over the years, the two parties have had different objectives and strategies. On the one hand Delivery is pressing for faster software releases (e.g., SCRUM cycles are currently two weeks long), and on the other hand I&O, which considers system stability of the highest importance and plans releases monthly. Recently, the management has decided that Delivery and I&O should integrate and should align their processes to improve the overall efficiency, e.g., to release deliverables in a balanced way and more often without compromising on the quality of the releases. To establish this closer collaboration between Delivery and I&O, DevOps has been introduced. This methodology originates from lean methodologies and stresses the improvement of respectively work flow, final product quality, team communication and customer feedback [1]. The methodology is process flow oriented, which means that it focuses at deliverables moving through the processes, on increasing development speed and decreasing waiting times. The implementation of DevOps has been started with a limited number of teams within Delivery. Because agile software development methods are currently in use at Delivery and also I&O is looking at ways to implement agile methods, it was decided to make explicit use of agile and lean principles in the implementation of DevOps [6], [7], [9]. To monitor and control the DevOps implementation, an initial metrics model had to be developed. In Section II, we will address the background of agile methods and techniques and the key principles of DevOps. Section III will present the methodology used in the research to develop the initial DevOps metrics model. In Section IV, a case study on the development of the metrics model will be presented, following an iterative approach within the company Philips IT. In this case study, researchers in close collaboration with Delivery and I&O practitioners have developed in three cycles an initial metrics model. Section V presents a discussion and Section VI finalizes the paper with conclusions.

II. BACKGROUND AND REFERENCE FRAMEWORK

Agile software development originated from the 'The Agile Manifesto' [5] and consists of several values and principles for faster and better software development. Four values are respectively: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan. While there is not a single definition of agility, most approaches incorporate the idea of adaptability to the environment and quick value creation [6]: "agility means to strip away as much of the heaviness, commonly associated with the traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines and the like." While this definition is

 ISBN: 978-1-61208-448-0

focused on software development, similar trends have been previously seen in other disciplines. In [7] for example, agility is related to "flexibility" and "leanness". However, several differences exist between the terms. According to [8], agility consists of two components: flexibility and speed, hereby stating that flexibility alone is not enough to be agile. In [9], particularly flexibility is addressed, with respect to decision making, and speed with respect to short iterations in development. Comparing agility to leanness, these both complement each other with regard to simplicity and quality, but the economy perspective of the approaches is different [10]. While leanness attempts to remove 'waste' entirely, agility removes waste only to the extent that it does not hinder the ability to change [11]. Next to these definitions on agile a multitude of methods have been developed. Table I reflects the characteristics of a selected set of them.

TABLE I. AGILE METHODS.

Agile method	Description
Scrum [9]	The development is organized in sprints (short iterations of about 2 to 3 weeks) by self-organizing teams. Each sprint, i.e., restricted time, goes through planning, design, testing and review. Features that need to be developed are stored in a 'Backlog' where the product owner decides, which work items will be worked on in the following sprint.
Extreme Programming (XP) [5]	Focuses on best practice and consists of twelve practices: the planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, 40h week, on-site customer collaboration, and coding standards.
Lean software development [11]	Based on seven principles: remove waste, amplify learning and knowledge management, decide as late as possible, deliver as fast as possible, empowered teams, build integrity, and see the whole picture.
Kanban [20]	Kanban is based on the theory of constraints and comes with six core practices; visualize, limit work in progress (WIP), manage flow, make policies explicit, implement feedback loops, improve collaboratively & evolve experimentally.

The agile methods show quite some similarities regarding speed (e.g., fast delivery), small releases (e.g., limit work in progress), remove waste (e.g., manage flow), implement feedback loops (e.g., customer collaboration) and learning and experimentation, and knowledge management. Scrum stresses additionally the self-organization of teams and other team-work characteristics. Since 2009, DevOps has been introduced, which focuses on the way development and deployment (i.e., operations/maintenance) can be integrated [1]. While development teams and deployment teams have often different goals or key performance indicators, DevOps attempts to align the work to be done, and to satisfy the different goals. For example, as development teams want to deploy more and more often, deployment teams strive often towards the exact opposite, i.e., to keep all systems running and stable. However, and in accordance with DevOps, an entire organization should be aligned and/or integrated. To reach this, DevOps proposes to follow three subapproaches [12], see Table II.

TABLE II. THREE APPROACHES OF DEVOPS.

Systems thinking	Stresses that it is more beneficial to look at the performance of an entire system, than at the performance of specific parts of that system.
Amplify feedback loops	Allows understanding of the customer by the teams and availability of knowledge where it is needed.
Culture of continuous experimentation and learning	Experimentation and learning helps to more quickly adapt and respond to changes or problems.

To use these three subapproaches of DevOps as a reference framework, the three approaches can be elaborated on the basis of agile principles. *Systems thinking* refers to looking at problems in relation to the performance of an entire system, also addressed as 'overall quality of work'. This approach ensures that the performance of a system as a whole is more important than the performance of separate parts of the system (e.g., a development and a deployment part). This approach can make use of agile principles (see Table I) such as remove waste, decrease incidents and continuously focus on (process) flow to increase performance. *Amplifying feedback loops* leads to early knowledge of issues and problems, so that a system can quickly be adjusted where needed. Implementing this second subapproach should lead to, with reference to agile issues in Table I, in particular an understanding of, and responding to customers. To deliver finally value, the feedback should come from the people (i.e., customers) who will use the product or service and from those who maintain it. The third subapproach, i.e., a *culture of continuous experimentation and learning*, supports the other two, to ensure that improvement should be a continuous process and should lead to, with reference to the agile principles in Table I, respectively: facilitating knowledge storage and retrieval, and reflection on deliverables and on the way of working. Regarding 'culture of learning and experimentation' references can be made to specific constructs or organizational learning [13], such as the acquisition of knowledge, either through external sources or internal development, the distribution of knowledge, and the interpretation of knowledge (i.e., the way that people within an organization share and use the knowledge).

To implement DevOps on the basis of the three foregoing subapproaches, with the references to agile principles, and to monitor the effectiveness of it, performance indicators or metrics have to be defined. Regarding the development of metrics the Goal-Question-Metric (GQM) approach will be

 ISBN: 978-1-61208-448-0

used [14]. Based on well-defined goals of a particular object under study, here the DevOps process, asking questions and getting answers regarding the achievement of the goals, will lead to a well-founded set of metrics. To support the definition of goals, the development of questions /answers, and the derivation of metrics, particular templates will be used [15].

III. METHODOLOGY OF THE CASE STUDY

The first step in the case study was defining the goals, making use of structured templates [14]. This has been done in collaboration with 'those working in the environment itself' to ensure the understandability and the applicability of the metrics [16]. In this step, we made use of the background as explored in Section II, in particular regarding the three subapproaches of DevOps and the agile pinciples identified. In step 2, a set of metrics has been derived from the defined goals. In this step, in meetings with experts from practice, questions have been developed regarding the defined goal(s) [17]. Subsequently, metrics have been derived to measure the performance. The metrics have formed together an initial metrics model. In step 3, iterations have been executed to elaborate and validate iteratively the set of metrics [18]. These iterations have been stopped in case the set of metrics didn't change significantly from its previous iterations. The first iteration has been executed with respectively the Manager I&O and the Global Demand Manager (management level above Delivery and I&O). These representatives were selected because the assignment, of the case study at hand, originated from them. A second iteration has been executed with the Delivery Manager. Its position was close to the teams in that the metrics had to be applied.

IV. TOWARDS AN INITIAL METRICS MODEL FOR DEVOPS, THE CASE STUDY

A. Goal definition for the measurement of DevOps

To support the goal definition, the following template has been applied [15].

TABLE III. GQM GOAL DEFINITION TEMPLATE.

Analyze	The object under measurement
For the purpose of	Understanding, controlling or improving the object
With respect to	The quality focus of the object that the measurement focuses on
From the viewpoint of	The people who have a stake in measuring the object
In the context of	The environment in which measurement takes place

The object under measurement, see Table III, is in this case study the integrated development and deployment process, i.e., the DevOps process within the company. The purpose for the measurement is to further understand this process and if possible to improve it. The focus will be on the three subapproaches within DevOps, respectively systems thinking, feedback loops and a culture of learning and experimentation. The people who have a stake in measuring the object, i.e., reflecting the three viewpoints are respectively the Global Demand Manager, the Delivery manager and the I&O manager. Table IV shows the goals as defined on the basis of the template.

TABLE IV. THE DEFINED GOALS FOR DEVOPS MEASUREMENT.

Goal 1	Analyze the development and deployment process within Philips IT to further understand and improve with respect to systems thinking from the viewpoint of the IT management.
Goal 2	Analyze the development and deployment process within Philips IT to further understand and improve with respect to feedback loops from the viewpoint of the IT management.
Goal 3	Analyze the development and deployment process within Philips IT to further understand and improve with respect to culture of learning and experimentation from the viewpoint of the IT management.

B. Formulating questions to derive metrics for DevOps.

Regarding the goal of 'systems thinking', it was decided to look at the performance of the process as a whole (i.e., also addressed as the 'overall quality of work') opposed to its separate parts. This has lead to the following two questions: what is the current performance of the entire process, and do changes in the process improve the performance of the entire process? Regarding the goal of 'feedback loops within the system', the following questions are formulated: what is the current state of feedback loops within the process? Is the customer satisfied with the feedback that can be given? How well can the process respond to feedback? Do changes in the process improve the state of feedback loops within the process? Regarding the the goal of 'culture of learning and experimentation', questions are formulated about the current state of the culture, and the improvement of learning and experimentation [13].

C. Deriving an initial metrics model for DevOps

Deriving initial metrics for DevOps systems thinking

Following GQM, i.e., answering the questions, metrics have been derived. To describe the performance of the entire process, the average cycle time of a user story has been discussed. While this metric only takes into account the speed of development, it was decided to choose a second metric regarding the 'overall quality of the work'. The rationale is that higher quality leads to less rework, which should lead to a better lead time [19].

TABLE V. METRICS FOR SYSTEM THINKING.

Questions to goals	Metrics
What is the current performance of the process?	Average cycle time of a user story Number of incidents after deployment Costs of a feature
Do changes in the process improve the performance (average lead time: avglt; average number of incidents: avgni) of the entire process?	Avglt of a user story after change ---------- * 100% Avglt of a user story before change Avgni after deployment after change ---------- * 100% Avgni after deployment before change

In the case study company, in particular I&O teams are already measuring the amount of incidents that occur following an implementation. Regarding changes in the process, two metrics have been derived (based on the foregoing metrics) to reflect the differences between the performance before and after a change. Table V presents the derived metrics.

Deriving initial metrics for DevOps feedback loops

Initially, the amount of feedback loops has been defined as metric. However, this metric appeared to be depended on the length of the process. To take the length of the process out of the metric, the average time between feedback moments (i.e., the contact points with customers) has been chosen. A problem with this would however be that if the only feedback moment is located at the end of the process, the average time would be same as if the feedback moment would be right in the middle of the process. To cover this, an additional metric has been defined to keep track of the maximum time within a process without feedback. When this time is very close to the average time between feedback moments, the feedback moments will be evenly spread out over the process. Regarding customer satisfaction, a qualitative metric has been defined by asking the customer whether he would like to have the next feedback moment quicker than the time since the last feedback moment. Regarding how well the system can respond to given feedback, a first suggestion was to look at the amount of work, which has to be redone within the process. This can be quantified by the amount of time spent from the moment of feedback until the process reaches the same point again. While this could be difficult to measure in practice, also an easier metric has been defined, i.e., the total time spent on rework during the process. An overview of the second set of metrics relating to feedback loops is shown in Table VI.

TABLE VI. METRICS FOR FEEDBACK LOOPS.

Questions to goals	Metrics
What is the current state of feedback loops within the system?	Average time and mMaximum time between feedback moments
Is the customer satisfied with the feedback that can be given?	Need of the customer to have the next feedback moment quicker or later than the time since the last feedback moment
How well can the system respond to feedback?	Time spent from feedback moment untill reaching the same point, total time spent on rework (after feedback)

Deriving initial metrics for DevOps culture of learning and experimentation.

Regarding the current state of learning, two metrics have been defined, respectively with respect to the fact whether new knowledge is actively being stored and whether stored knowledge can be actively retrieved. To determine if knowledge is being shared, as well as whether a mechanism is in place to make sure that knowledge is actually being stored, a metric has been defined on the reflection of a team on its work and learnings points being defined after a project (or a 'sprint'). Finally, a metric has been on the reflection of a team on their way of working (and thus takes time to improve). The metrics are shown in Table VII.

TABLE VII. METRICS FOR LEARNING AND EXPERIMENTATION.

Questions to goals	Metrics
What is the current state of learning and experimentation within the system?	Amount of new knowledge stored during the process Extent to that previously acquired knowledge can be retrieved Extent to that teams reflect on their work and learning points after a project or sprint

D. *Iterative refinement of the initial metrics model*

First iteration.

The designed metrics model has been refined in the first iteration in two separate sessions. In these two sessions, the initial metrics model was briefly explained, in particular regarding the understandability of the logic of the interrelations between goals, questions and metrics. Subsequently, the participants were asked to come up with alternatives or changes to or extensions of the metrics. Regarding the metrics for 'systems thinking', there were three (summarised from the two sessions) main points of feedback. First, regarding the 'user story', it was decided that a different unit of measurement had to be used, namely a 'feature'. The reason was that in the process, a collection of user stories moves through the process simultaneously, except for the part of the process where they are developed. Consequently, measurement of user stories would not provide information about the entire process. Secondly, it was decided that the specification of cost within the process should be further defined. Considering the fact that this process contains quite some knowledge work, and no tangible products, the cost of a feature should be calculated on the basis of the hours spent, the amount of people working on it, and the number of features being worked on. Thirdly, it was decided that by using the metric on the first question periodically or continuously, the second question on change, see Table V, would be irrelevant, and could be removed, see Table VIII.

TABLE VIII. METRICS FOR SYSTEM THINKING, BASED ON FIRST FEEDBACK.

Questions to goals	Metrics
What is the current performance of the process?	Average cycle time of a feature Average waiting time of a feature Number of incidents as a result of the feature after deployment The cost of a feature through a process: Hours spent Number of people Number of features being worked on

Regarding the metrics for 'feedback loops', in one session the participants mainly agreed on the proposed metrics and suggested some small changes in terminology. In the second session a different understanding of what should happen in feedback loops lead to discussions. On the one hand, it was understood that feedback would internally lead to more insight in how fast changes in the system were

executed, while on the other hand the importance of feedback to customers was stressed. It was also suggested that feedback moments with customers had to be changed to so-called 'touch points' for a better understanding within the company. These discussions lead finally to Table IX.

TABLE IX. METRICS FOR FEEDBACK LOOPS, BASED ON FIRST FEEDBACK

Questions to goals	Metrics
What is the current state of feedback loops within the system?	Average time between customer touch points Maximum time between customer touch points
How well can the system respond to feedback?	Time spent on feedback untill reaching the same process step Time spent on rework
How fast can the system respond to changes in a process?	The average time a change is seen at the end of the entire process

Regarding the metrics for 'culture for learning and experimentation', it was initially more difficult to find useful metrics. Some feedback included the addition of metrics related to the capabilities of the team members, and to how well people could perform the activities of other team members. However, by just measuring the capabilities, it would mean that you can get a culture of learning by simply hiring the people with excellent capabilities. Also suggestions were made that the number of value propositions should be counted. Here, a value proposition would mean a member making a suggestion for a change in the process, or a team, with an estimated value that is estimated by implementing the change. However, this suggestion was rejected because of the time that it would require. It was decided then that the focus for learning should be put on the time spent on improving the teams that perform their daily work. Thus measuring their time spent on storing and retrieving knowledge, and on learning (i.e., reflecting) and improving. Experimentation was considered as very relevant and some discussions lead to a metric on the introduction and subsequent discovery of faults by different teams, see Table X.

TABLE X. METRICS FOR LEARNING AND EXPERIMENTATION, BASED ON FIRST FEEDBACK

Questions to goals	Metrics
What is the current state of learning and experimentation within the system?	The amount of time spent to store new knowledge during the process The amount of time spent to retrieve previously acquired knowledge Amount of time spent on reflection of a team on their work and on learning points after a project or sprint? Amount of time spent on reflection of a team on the way of working after a project or sprint? Percentage of discovered faults by a team with respect to introduced faults by another team (experimentation).

Second iteration

The second iteration consisted of one session and has been carried out with only the Delivery Manager. The feedback in this session mainly consisted of small updates and clarifications. This feedback was more on the confirmation (and validation) of the changes in the foregoing session then in actually changing the metrics. Regarding the first and the second subgoal, two particular terms had to be clarified. Firstly, cycle time was changed to lead time and secondly the cost of a feature was further elaborated by adding service costs. Although the feedback consisted of serious doubts regarding the time that the extra work of experimentation would cost, i.e., introducing and discovering faults, experimentation was kept in the metrics model.

V. DISCUSSION

Metrics development to measure the performance of DevOps requires a structured aproach and a clear reference framework. The implementation of DevOps could be based on three subapproaches, with an explicit reference to agile and lean principles. The application of GQM to determine metrics could profit from this reference framework. The reference framework facilitated the development of questions, the interpretation of the answers and the initial determination of metrics. However, the reference framework is still qualitative and should be investigated furter. Although GQM is an approach that has received positive response in literature, criticism states that the outcome is rather unpredictable as it is still possible to derive many different metrics that describe a particular defined goal. However, our experience in the case study has shown that by carrying out feedback loops, it is possible to discuss and (re)define metrics and to reach consensus on metrics in close collaboration with responsible experts from practice. Although not all derived metrics have clear references to literature, interesting similarities could be found. Regarding the first DevOps subapproach of 'systems thinking', parallels have been found in lean manufacturing and agile literature with respect to average lead time of 'user stories' and the amount of 'features being worked' on simultaneously [19]. However, we couldn't find Scrum-specific similarities, e.g., regarding our metrics addressing costs and quality (e.g., number of incidents). Regarding the second DevOps subapproach of 'amplifying feedback loops', the parallels between our metrics model and literature are more hidden, but are most certainly present. For instance, the time spent on rework (after feedback) is mentioned in agile and lean literature as the percentage of 'units sent for rework' [19]. The other metrics found, such as number of approvals, are more closely related to software development in general and are less present in literature on Scrum. Regarding the third DevOps subapproach 'a culture or learning and experimentation', the derived metrics turned out to be quite different than what was previously found in literature [13]. Metrics (areas) in literature addressed appeared to be too abstract. Therefore, we have chosen

 ISBN: 978-1-61208-448-0

simpler and more direct metrics in terms of 'time spent on…'.

Reflection on the metrics model from a literature point of view showed that the agile principles identified in lean manufacturing literature turned out to be quite helpful in particular with respect to the first subapproach. However, the metrics investigated in literature on Scrum could not be used in our metrics model. The reason for this is most likely the focus of Scrum metrics. While agile and lean manufacturing metrics focus on the entire process, similar to the focus of our initial metrics model, Scrum metrics focus on teams working within this process. A preliminar conclusion could be that Scrum metrics are probably too team-specific to address the goals of an entire DevOps process. But this should be investigated further, preferably in case studies in that the initial metrics model has to be validated and elaborated further.

VI. CONCLUSIONS

This paper shows that regarding the integration of software development and deployment activities, on the basis of Devops, an initial metrics model could be developed. This metrics model has been developed in a structured way, in a small number of iterations, with responsible practitioners. The objective of the metrics model is the measurement of the effectiveness of the DevOps implementation. The structured GQM-development of the initial metrics model was facilitated by a reference framework, i.e., consisting of the three elaborated DevOps approaches and the agile and lean principles in Section II. This reference framework will also provide a basis for further refinement of the metrics model. Although interesting, and for the company useful, results have been obtained, the metrics model is still in an initial state. In future research and case studies, we will continue the iterative development of the metrics model, towards a well-founded and transparent measurement of the effectiveness of DevOps.

ACKNOWLEDGMENT

The author would like to thank Sander Kruis for his valuable MSc thesis project at TU/e and Philips IT Eindhoven, The Netherlands.

REFERENCES

[1] G. Kim, K. Behr and G. Spafford, The phoenix project: A novel about IT, DevOps, and helping your business win. IT Revolution Press, 2013.

[2] M. Mamun and J. Hansson, "Review and Challenges of Assumptions in Software Development", Chalmers University and University of Gothenburg, Sweden, (2011), http://publications.lib.chalmers.se/records/fulltext/local_154439.pdf

[3] S. Downey and J. Sutherland, "Scrum metrics for hyperproductive teams: How they fly like fighter aircraft", 46th Hawaii International Conference on System Sciences (HICCS), 2013, pp. 4870-4878. Hawaii.

[4] ITIL 2011 - The Big Picture, Retrieved 7, 2015, http://cfnpeople.com/downloads/itil_poster_the_big_picture_cfn_people.pdf.

[5] K. Beck and C. Andres, Extreme programming explained: Embrace change, Addison-Wesley, 2000.

[6] J. Erickson, K. Lyytinen and K. Siau. "Agile modeling, agile software development, and extreme programming: The state of research", Journal of database management, 16 (4), 2005, pp. 13-18.

[7] K. Conboy and B. Fitzgerald, "Towards a conceptual framework of agile methods: A study of agility in different disciplines", ACM workshop on Interdisciplinary software engineering research, 2004, pp. 37-44, Newport Beach.

[8] Z. Zhang and H. Sharifi, "A methodology for achieving agility in manufacturing organisations", International journal of operations & production management, 20 (4), 2000, 496-512.

[9] K. Schwaber and M. Beedle, Agile development with Scrum, Prentice Hall, 2001.

[10] K. W. Young, R. Muchlhaeusser, R. Pigging and P. Rachitrangsan, "Agile control systems". Journal of automobile engineering, 2001.

[11] M. Poppendieck and T. Poppendieck, Lean software development: An agile toolkit for software development managers, Boston: Addison-Wesley, 2001.

[12] G. Kim, DevOps distilled, Part 1: The three underlying principles. Retrieved 7, 2015, IBM Developerworks: http://www.ibm.com/developerworks/library/se-devops/part1/index.html

[13] S. López, J. Peón and C. Ordás, "Managing knowledge: the link between culture and organizational learning", Journal of knowledge management, 8 (6), 2004, pp. 93-104.

[14] R. Van Solingen and E. Berghout. The goal question metric method: a practical guide for quality improvement of software development, McGraw-Hill Inc, 1999.

[15] V. Basili, G. Caldiera and D. Rombach, "Goal Question Metric Paradigm", Encyclopedia of Software Engineering, 1, 1994, pp. 528-532.

[16] S. Pfleeger, "Lessons learned in building a corporate metrics program", IEEE Software, 1993, pp. 67-74.

[17] J. McNiff, Action research; Principles and practice, London & New York: Routledge, 2013.

[18] R. K. Yin, Case study research design and methods, Newbury Park: Sage Publications, 1989.

[19] D. F. Duque and L. R. Cadavid, "Lean manufactoring measurement: The relationship between lean activities and lean metrics", Estudios Gerenciales, 23 (105), 2007, pp. 69-83.

[20] D. J. Anderson, Kanban: Successful evolutionary change for your technology business, Blue Hole Press, 2010.

 ISBN: 978-1-61208-448-0

A Novel Three-layer Architecture for Information System Integration

Kamrul Ahsan
School of Information Sciences, University of Tampere
Integration Excellence, HiQ Finland Oy
Tampere, Finland
e-mail: ahsan.kamrul.x@student.uta.fi

Juha-Miikka Nurmilaakso
Integration Excellence, HiQ Finland Oy
Espoo, Finland
e-mail: juha.nurmilaakso@hiq.fi

***Abstract*— The purpose of Information System Integration (ISI) is to streamline business processes by synchronized or asynchoronized completion of a series of steps. Integration architects use the so called "integration layer" as a methodology to accomplish such tasks. To date, in the literature, three kinds of layer mechanisms are reported: No (Point-to-Point), One (Message Brokers), and Two (Message Bus). Although these three kind of layer types can solve most of the integration challenges, among them there are both design and run-time quality challenges. In this paper, a new type of layering, named "Three integration layers" is introduced to improve the quality of the integration solutions. Also, this paper argues that new integration layers can be used for ISI projects to improve IS integration quality.**

Keywords-information system integration; layer architecture; router.

I. INTRODUCTION

Information System (IS) is the combination of information technology, data, personnel and associated business functions which interact to generate information and creates an information resource which assists the organization to achieve its business goals. To accomplish its business goal, organization uses multiple IS, which leads to multiple information source. Information System Integration (ISI) is the sets of tools and methodology that allows various IS to interact each other to create aggregate business value, reduce heterogeneity of the IS, allows to adopt of new information technology (IT), facilitate e-commerce, improve business efficiency, allows managers in enhancing performance, increases complete knowledge of the enterprise and its customers in decision making process [1] [7] [8].

The ISI concept is multi-faceted and multidimensional. In terms of information integration, the architecture of the ISI can be – use either a virtualization approach or a materialization approach [1]. In the virtualization approach, the data resides in the individual data sources and the virtualization layer is defined as a virtual schema which has attributes from all the data sources. When a user query defined on the virtual schema is received by the system, it determines the relevant sources to be queried and then breaks down the query into sub-queries for the different sources [1]. On the other hand, the materialization approach, the data is materialized at the global level. This approach is generally used in data warehouses and it does not have any unstructured information [1].

In the conceptual model, a layer-based architectural pattern is used in ISI implementation projects [2]. Currently, there are three ISI layer architectures available namely No integration layers or point-to-point, one integration layer or message brokers, and two integration layers or clients / servers or Enterprise Service Bus (ESB). In general, the No integration layer or point-to-point integration, data flows directly from system to system. A point-to-point connection ensures that only one receiver receives a particular message [4]. One integration layer or message brokers (also referred to as hub-and-spoke architectural style) receive messages from multiple destinations, determine the correct destination and route the message to the correct channel. Finally, the two or Client/Server or service bus integration is the integration systems that are comprised of two logical parts: a server that provides the integration services and a client that requests services of the server. Together, they form a complete integration system with a distinct division of responsibility.

Due to the complexity of IS, constant changes of business processes, technological advancement and cloud computing, current layer-based architectural pattern of ISI has shortcoming in terms of integration quality, especially quality aspects such as design and run-time attributes. Table I outlines some of the current challenges of various layer-based architectural patterns.

In order to overcome some of the quality attributes of the current integration layering based architectural challenges, a new type of layer named three-layer or router-based layer pattern is introduced in this paper. The routing layer or router works as an orchestration, which is configurable. The router utilizes a content-based publish-and-subscribe pattern with filters and self-correlations to support integrations between the receiving and sending layers. This integration takes place by wrapping a source or target message as a payload in a routing-specific envelope. The architecture can be implemented as a standalone application using Web Services (WS) or by using any modern ESB (For example, Microsoft BizTalk).

 ISBN: 978-1-61208-448-0

TABLE I. INTEGRATION PATTERNS AND THEIR CHALLENGES [5] [6]

Patterns	Challenges
No-layer	- Number of connections increases with respect to number of applications - Tight coupling - Less extendable - Limited re-use - Less scalable - Rigid in terms of agility - Limited to technology (Technology constraint)
One-layer	- Single point of failure - High cost - Excessive use of network resources - Unable to integrate applications without enforcing a common interface - Cannot allow each application to initiate interactions with several other applications
Two-layer	- The cost of adding or removing applications increases as an integration solution grows - Unable to only send messages to the applications that are interested in receiving the messages without knowing the identities of the receivers

The proposed layer-based integration architecture tries to solve some of the quality challenges of ISI. In order to show the area of the IS, where this new artifacts fits, Section II contains two useful theory of ISI: interoperability and integration - with brief narrative descriptions and relevancy with the proposed method. Existing three types of layer-based integration architecture have been briefly presented and compared in the Section III. The proposed new type of integration architecture is shown in details in Section IV with comparisons with two layer-based architecture. In Section V, four kinds of layer-based integration architecture has been discussed in relation to some of the ISI related quality attributes. Finally, Section VI contains the concluding remark.

II. BACKGROUND

Today organizations use multiple software or information systems to operate their day to day business operations. In order to achieve aggregate business value, these IS need to be integrated. Unlike software integration, which is the practice of assembling a set of software components/subsystems to produce a single, unified software system, ISI can be defined as combination of system, software and tools integration for modernizing, consolidating, and coordinating the computer applications in the organization [9].

The architecture of ISI is layered-based due to the fact that IS architecture comprises of: information, application and technology. Thus, layer-based design patterns are suitable for logic interaction. Integration in IS can occur at the data, method, interface, portal, and process level and such variety basically represents how the application "sees" integration [9]. To address such variety, Hohpe and Woolf [3] have divided the integration types as: Business Process Integration, Messaging based, Remote Procedure Invocation (RPC), Shared Database, Managed File Transfer (MFT), User-Interface based, and Data integration.

Another important characteristic of ISI architecture is messaging. Regardless of integration type, the exchange of data is common in all ISI levels [9]. Data (for example, orders or invoices, not integers or strings) is the primary means to integrate multiple applications so that they work together by exchanging information without loss of accuracy. ISI uses messaging techniques to transfer packets of data frequently, immediately (synchronously), reliably, and asynchronously using customizable formats [3]. ISI uses a special filter named message router, which consumes a message from one message channel and republishes it to a different message channel depending on a set of conditions [3]. Message routers are categorized into the following groups: simple routers which are variants of the message router and route messages from one inbound channel to one or more outbound channels, composed routers that combine multiple simple routers to create more complex message flows, and finally the architectural patterns which describe architectural styles based on message routers [3]. Furthermore, following message routing patterns are described by [3]: Content-Based Router, Message Filter, Dynamic Router, Recipient List, Splitter, Aggregator, Re-sequencer, and Composed Message Processor.

Finally, clear understanding of the term integration and interoperability is needed when systems needs to be integrated. The integration is the practice of assembling a set of software components/subsystems to produce a single, unified software system that supports some need of an organization [15]. On the other hand, interoperability is the ability for two systems to understand one another and to use functionality of one another [16]. The word ''inter-operate'' implies that one system performs an operation for another system. Thus by definition, two integrated systems are inevitably interoperable; but two interoperable systems are not necessarily integrated [16]. For example, a good analogy could be the relationship between Video Cassette Recording (VCR) and Television (TV). A VCR and a TV bought in the same country are interoperable. One just needs to connect them together. However, a VCR bought in the US and a TV bought in the UK may need the special signal conversion services of an NTSC/PAL converter in order to integrate them.

III. LAYER-BASED INTEGRATION ARCHITECTURE

In this Section, traditional layer-based integration architectures have been described and compared. To visualize and count number of interactions needed by each integration types to implement a business process, "order creation" [17] business process is used. It is sales related or more specifically Sales Order (SO) business process where client creates a SO from a portal to Enterprise Resource Planning (ERP) system. The business process also needs to check whether subjected client exists in the Customer Relationship Management (CRM) system.

 ISBN: 978-1-61208-448-0

A. *No integration layers: point-to-point*

A point-to-point integration (Figure 1) ensures, that only one receiver receives a particular message [5]. For this to work, the sending system must know the location of the receiving node. The sending system often must translate the message into a format that the receiving system understands.

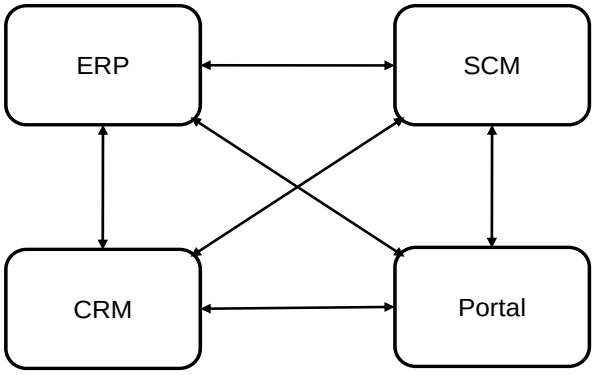


Figure 1. No integration layer-based or direct ISI

Although it is easier, less costly and requires less upfront work to implement point-to-point integration, the method is not suitable when a large number of applications need to be integrated. The number of point-to-point integrations basically increases as follows: N*(N – 1) / 2. In this calculation, N is the number of applications involved in the integration [5]. Figure 2 shows message flows among various IS to create a SO business process. Note that, the CustomerPortal which is responsible for integrations may demand expensive software customization.

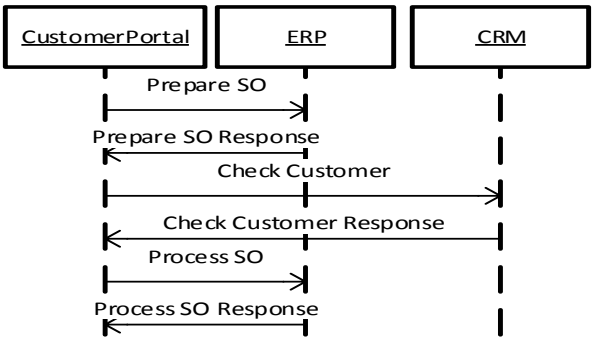


Figure 2. Sales order business process using point-to-point integration

B. *One integration layer: message broker*

A message broker can receive messages from multiple sources, determine the correct destination and route the message to the correct channel (Figure 3).

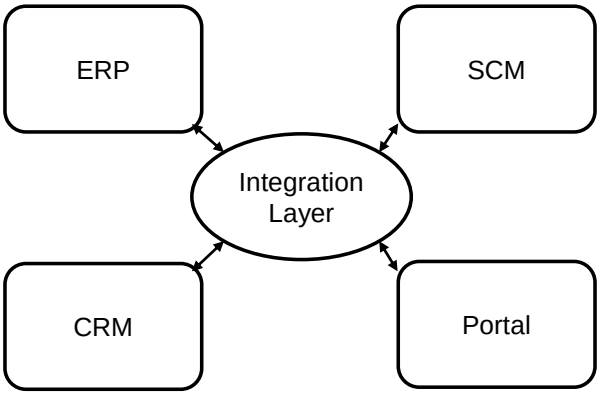


Figure 3. One integration layer-based ISI

It is a physical component that handles the communication between applications [10]. Instead of communicating with each other, applications communicate only with the message broker. An application sends a message to the message broker, providing the logical name of the receivers. The message broker looks up applications registered under the logical name and then passes the message to them. Figure 4 shows message flows among various IS to create a SO business process using one layer-based integration.

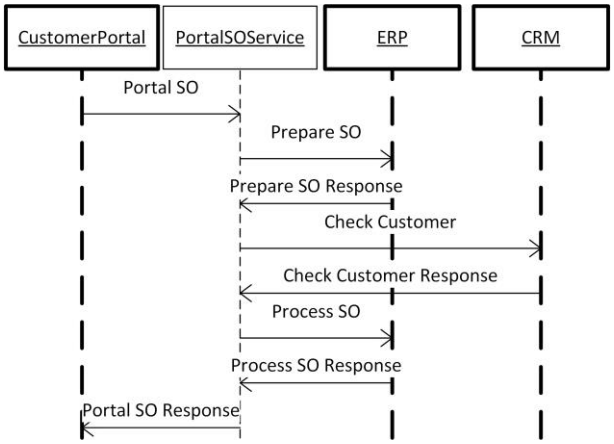


Figure 4. Sales order business process using message broker integration

 ISBN: 978-1-61208-448-0

TABLE II. COMPARISIONS BETWEEN ONE-LAYER AND NO-LAYER

		No-layer	One-layer
Design qualities			
Maintainability	Standardization of data, process and technology	It is required that business process and data model as well as technology must be included inside both the source and target system's integration interface so that the integration between systems works. This requires customization both in source/target system increasing the number of the interfaces.	No such requirements thus the number of interfaces reduces significantly.
	Add, update or delete integration process	It is challenging to change business processes or other integration related changes in source/target system and in their interfaces when required.	It is possible to modify, enrich, route and operational logic in the integration interfaces without changing the source/target system.
	Process monitoring and alerts	Fewer of the source / target system are able to provide an overall picture of the progress of the business processes as control of the feature depends of the subjected system.	Since all the processes use a single platform, it is relatively easy to monitor business processes and integration of technical performance, as well as to obtain consistent alarms faults using custom or built-in custom tools.
Run-time qualities			
Scalability		Less saleable and number of connections increases with respect to number of applications.	The integration substrate may be nodes, of which one or more are active at a time, and which are linked to one another either directly or with the help of the integration database. Thus, one layer architecture can provide a number of options for scalability.

C. Two integration layers: clients and servers / service bus

Two integration layer (Figure 5) or client/server or ESB or a message bus allows applications to connect through a logical component and it specializes in transporting messages between applications [11]. A message bus contains three key elements: A set of agreed-upon message schemas, a set of common command messages [3], and a shared infrastructure for sending bus messages to recipients.

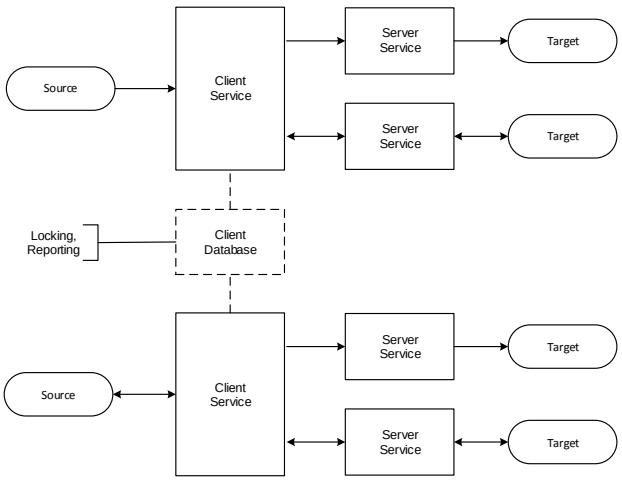


Figure 5. Conceptual model of two integration layer-based ISI

In two layer integration, an application that sends a message no longer has individual connections to all the

applications that must receive the message. Instead, the application merely passes the message to the message bus, and the message bus transports the message to all the other applications that are listening for bus messages through a shared infrastructure. Likewise, an application that receives a message no longer obtains it directly from the sender. Instead, it takes the message from the message bus. In effect, the message bus reduces the fan-out of each application from many to one [11].

The two integration layer-based ISI is capable of scaling pervasively across enterprise applications, regardless of physical location and technology platform [12]. Any application can plug into an ESB network using a number of connectivity options, and immediately participate in data sharing with other applications that are exposed across the bus as shared services. This is why the ESB is often referred to as an integration network or integration fabric [12]. Figure 6 shows message flows among various IS to create a SO business process using two layer integration.

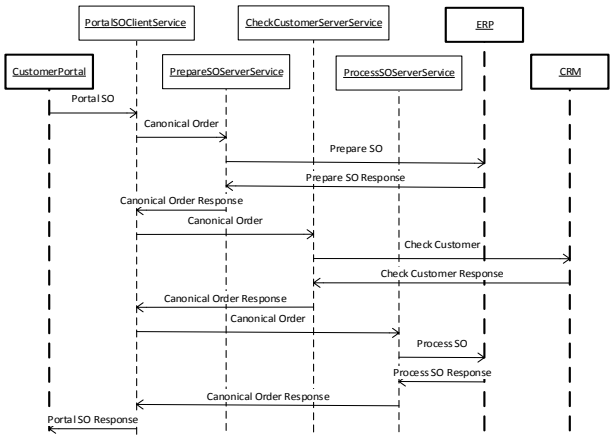


Figure 6. Sales order business process using two layer integration

Finally, Table III compares some of the ISI quality attributes between two-layer and one-layer.

IV. PROPOSED THREE-LAYER ARCHITECTURE A

The conceptual model of the proposed three layer-based integration is composed of three layers namely receiving, routing and sending (Figure 7).

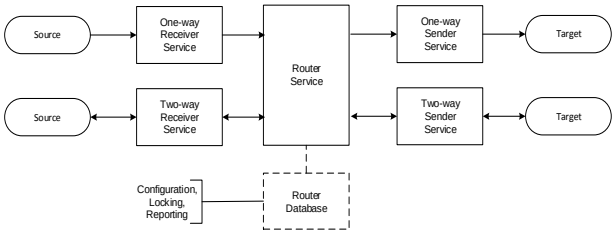


Figure 7. The conceptual model of three layer-based ISI

TABLE III. COMPARIEIONS BETWEEN ONE AND TWO LAYER

		One-layer	Two-layer
Design qualities			
Reusability		It does not contain any re-useable artifact	Server layer (component) services are reusable.
Run-time Qualities			
Flexibility	Data format	It does not support data modelling functionality.	It uses a CDM. In addition, applications can use adapters, so it is not mandatory all applications use the same data format.

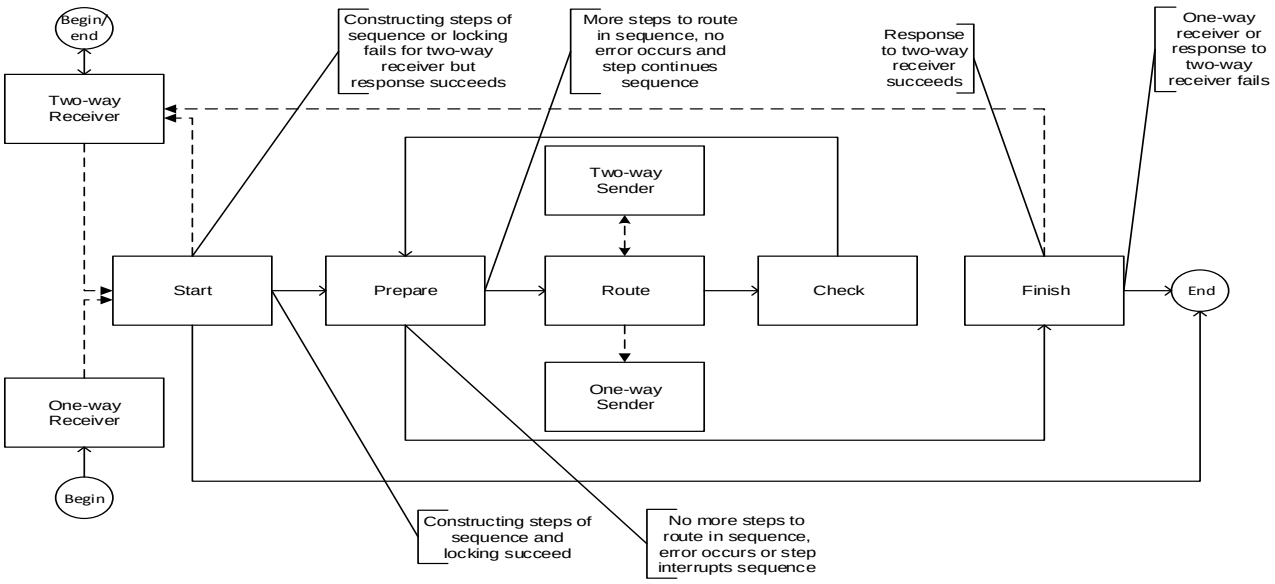


Figure 8. Details architecture of the three layer-based ISI

Compared to the Client Interface Layer (CIL) – Concept Layer (COL) architecture, which has two layers and CIL components integrate the external (party) systems and COL components integrate internal systems, the router architecture does not make such a distinction (Figure 8).

Figure 9 shows message flows among various IS to implement SO business processes using router-based integration.

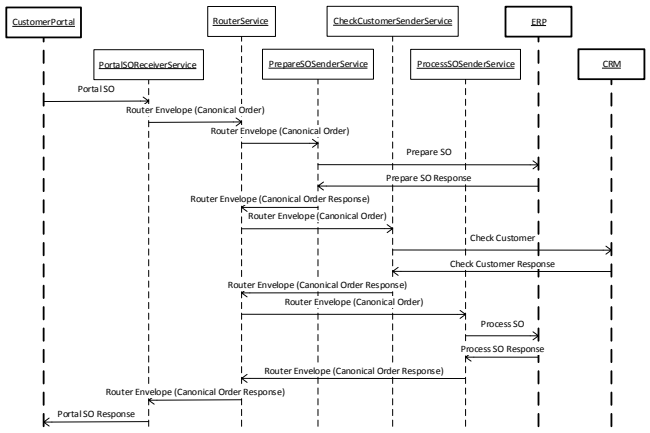


Figure 9. Sales order business process using router-based integration

A. *Router*

The routing layer or router is an orchestration, which is configurable by the RoutingDB (Name of the database) database. The router utilizes a content-based publish-and-subscribe pattern with filters and self-correlations to support integrations between the receiving and sending layers. This integration takes place by wrapping a source or target message as a payload in a routing-specific envelope. Applying the XPath (It is a language for addressing specific parts of an XML document) rules to the envelope, the router can choose how to use the sending layer. In addition, the router can employ time-based locking to ensure that only one router instance handles the same object (e.g., order or invoice etc.) at the sending layer. There are four kinds of routers:

- A non-locking one-way router receives a request envelope but does not send a response envelope or lock the object related to the envelope.
- A non-locking two-way router receives a request envelope and sends a response envelope but does not lock the object related to the envelope.
- A locking one-way router receives a request envelope, locks, relocks and unlocks the object related to the envelope but does not send a response envelope.
- A locking two-way router receives a request envelope, locks, relocks and unlocks the object related to the envelope and sends a response envelope.

B. *Receiver layer*

The receiving layer or receivers enable one-way and two-way integrations with external and internal source systems. The one-way/two-way receiver is an orchestration or a one-way/request-response receive port with a map and, if necessary, a custom pipeline. The receiver receives a source message using a source protocol, transforms the source message to an envelope and sends this envelope to the router. The two-way receiver also receives a response envelope, transforms it to a response message and sends the message to the source system using the source protocol. It is possible but not recommended, that the one-way receiver sends the response message or the two-way receiver does not send the message.

C. Sender layer

The sending layer or senders enable one-way and two-way integrations with external and internal target systems. The one-way/two-way sender is an orchestration or a one-way/solicit-response send port with a map and, if necessary, a custom pipeline. The sender receives an envelope, transforms this envelope to a target message and sends this target message to the target system using a target protocol. The two-way sender also receives a response message using the target protocol, transforms it to a response message and sends the message to the router. It is possible but not recommended that the one-way sender receives the response message or the two-way receiver does not receive the message.

Finally, Table IV compares ISI related quality attributes between router-layer and two layer integration pattern.

TABLE IV. COMPARISIONS BETWEEN TWO AND THREE LAYER

	Two-layer	Three-layer
Design qualities		
Reusability	Server layer (component) services are reusable.	Both receiver and sender layer (component) services are reusable
Run-time Qualities		
Flexibility	It does not have such functionalities.	Router layer (composite) services are flexibly configurable and provide reusable locking and monitoring functionalities for process integration.

V. DISCUSSION

Layered architecture is widely used architectural patterns in software design practice. It helps to structure applications that can be decomposed into n groups of subtasks in which each group is at a particular level of abstraction with well-defined interfaces [2]. Proposed new ISI router-based layer pattern improves quality and brings more flexibility to the integration architect over existing no, one and two layered-based ISI architecture.

Modularity

Parnas [13] defined information hiding as an approach to devising modular structures for software designs. The purpose of modular design is to decouple design decisions that are likely to change so that they can be changed independently and at the same time to improve the reusability and maintainability. Maintainability refers to the ease with, which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment [14]. On the other hand, reusability is the degree to which a software module or other work product can be used in more than one computer programs or software systems [14]. In the followings paragraphs, the proposed three layer-based ISI architecture is compared with other existing layer-based architectures with respect to maintainability and reusability.

First of all, at the center of modular design is the module it-self where business processes and integrations are mostly formed by independent modules. They are better known as services or producers/consumers. In such scenario, modules are made of standardized interfaces and functionalities. In addition, in order to communicate with other modules, interfaces must be compatible and follow standard data structure. The Canonical Data Model (CDM), which is sufficiently comprehensive and independent regardless of source and target systems, can be used for this purposes. It provides additional level of indirection between application's individual data formats. If a new application is added to the integration solution only transformation between the CDM has to be created, independent from the number of applications that already participate [3]. Another important requirements of the modularity is that orchestration, data structures and transformations required by the integration are not tightly coupled, but they are loosely coupled in the form of individual packages. In general, orchestration is integration specific, while some of the data structures and modifications may be generic.

Since no or direct ISI forms “tightly coupled” connections between components or source/target IS (Figure 1), it is impossible to design such integration using modular design principles. Thus, such integration is difficult to maintain. In addition, both interfaces and functionalities of direct integration often cannot be reused.

Two- and three-layer architectures are modular. As Figure 10 suggests, these approaches do not exclude each other and do not forbid use of orchestration which is an integral part of integration artifacts. For example (Figure 10), when some changes occur in the basic information of system A, then Master Data Management (MDM) and systems B, C, D will also be immediately updated. In addition, the MDM solution can provide details of the change for the basic information. If update procedure is successful in all relevant systems, change of basic

 ISBN: 978-1-61208-448-0

information in MDM solution can be confirmed. Otherwise changed but unconfirmed basic information can be updated in the A, B, C and D systems in a batch mode from the MDM solution. Note that, in this example, A and D systems only support asynchronous interfaces, whereas B and C systems support synchronous data transfer.

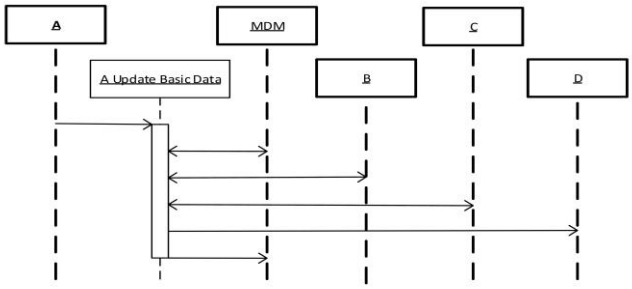


Figure 10. Example of layer-based ISI using basic orchestration

Two-layer architecture consists of CIL- and COL-services (Figure 11). The purpose of CIL-service is to transfer data with the source system, modify and enrich data in source system and the internal data structure for the data between the content and the distribution of the data content of the COL services. CIL-service can be of a stateless or state-full. On the other hand, COL-service is responsible for modifications and enrichment of the information content in the target system as well as transfer information within the target system. However, regardless of integration platform, the information transformation between CIL- and COL-services is done either using request-response or publisher-subscriber mechanism.

The three-layer architecture proposed in this paper is composed of receiver-service, router, and sender-service (Figure 12). Router has two versions which can serve as both one- and two-way receiver-services. Sender and receiver service support the internal data structure, and based-on such mechanism router service decide which router versions to use.

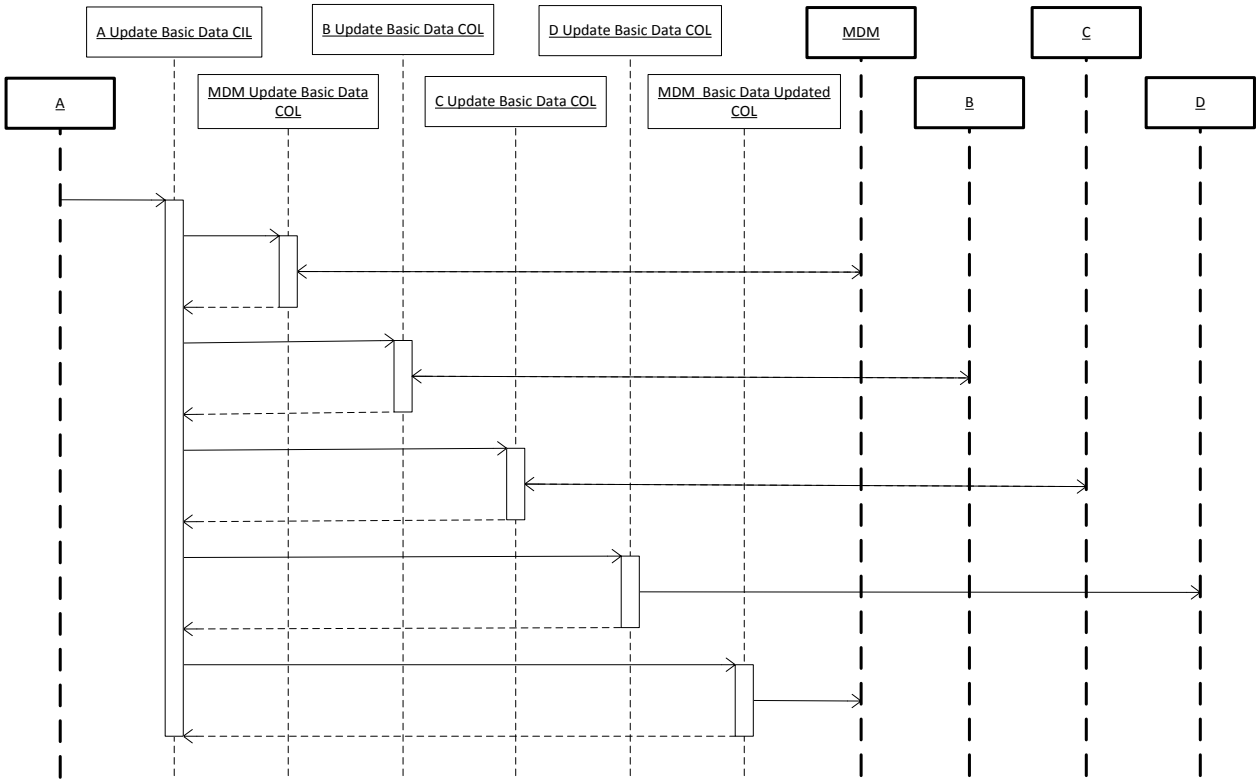


Figure 11. Detail example of two layer-based ISI

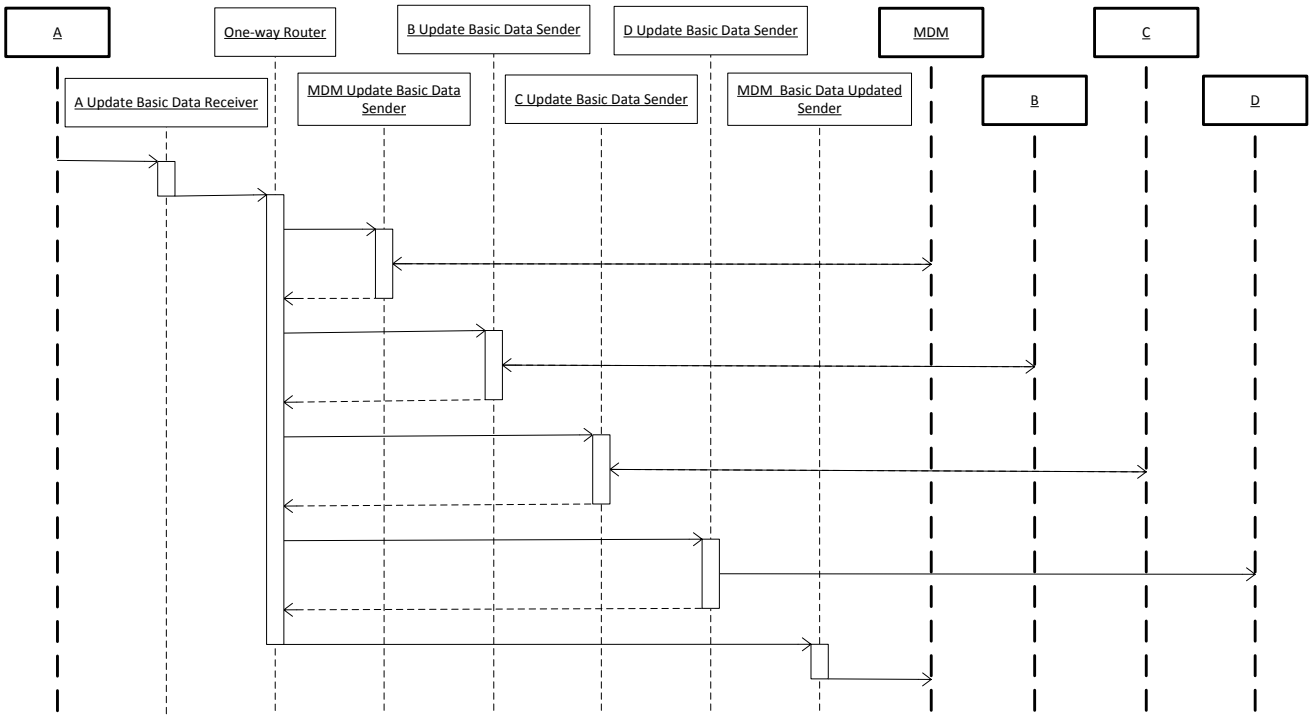


Figure 12. Detail example of three layer-based ISI

Figure 8 presents the behavior of the router service. The router server receives envelopes from one-way receivers, while it interacts with two-way receivers by request and response envelopes. To lock an object such as a sales order or to report its identity and state, the router service must create the object based to the envelope and store it into the database. The router service can lock and unlock a given object so that only one router service instance per this object can be performed at the same time. The router service instance has a sequence which consists of steps. Evaluating configured rules against the envelope determines which step the router service instance should perform next in the sequence. A step determines how the router service sends the envelope to a one-way sender or interacts with a two-way sender by the request and response envelopes. The router service can maintain the object state according to these response envelopes.

Router-service, two-way communication and Sender-Receiver with the services take place between CIL- and COL services. One-way service, this data is to be either one-way or publisher-subscriber relay model of integration depending on the substrate.

VI. CONCLUSIONS

Due to multiple known and unknown components (business and technology) both in-source, integration platform and destination systems, layer-based architecture is the right architectural fit for ISI. Although all the layer-based integration methods have advantages and disadvantages over one another, in relatively complex and routine integration projects, various quality attributes need to be considered. In this paper, both existing and proposed layer-based ISI architecture have been compared in terms of design and run-time quality category. Among all ISI architecture types, the proposed three layer-based or router based architecture provides more modularity and flexibility. Even though layer-based architecture also directly involve other quality category such as system and user qualities in the ISI, comprehensive comparisons of such quality categories are the natural direction for the future work of the proposed three-layer based ISI. The authors are carrying out further research including an empirical study to compare these architectures using a real life industrial business Use-case.

ACKNOWLEDGMENT

Authors would like to gratefully and sincerely thank Dr. Eleni Berki (University Of Tampere) and Elli Georgiadou (Middlesex University, UK) for the discussions, reviewing the paper and for their valuable comments during the writing process. In addition, authors also like to take the opportunities to thank HiQ Finland Oy and University Of Tampere for being very co-operative and supportive to carry out this research activities.

 ISBN: 978-1-61208-448-0

REFERENCES

[1] M. Mohania and M. Bhide,"New trends in information integration," ICUIMC '08 Proceedings of the 2nd international conference on Ubiquitous information management and communication, 2008, pp. 74-81

[2] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, "Pattern-Oriented Software Architecture, Volume 1, A System of Patterns," John Wiley and Sons, 2000, ISBN-10: 0471958697

[3] G. Hohpe and B. Woolf, "Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions," Addison Wesley, Oct. 2006, ISBN : 0-321-20068-3

[4] Microsoft patterns and practices, integration patterns, June. 2004. [retrieved: May, 2015]. Available from : https://msdn.microsoft.com/en-us/library/ff647958.aspx#intpatt-ch05_pointtopointconnection

[5] W. Roshen, "SOA-Based Enterprise Integration: A Step-by-Step Guide to Services-Based Application Integration," The McGraw-Hill, 2009, ISBN: 978-0-07-160553-3

[6] Microsoft patterns and practices, Integration Topologies, June. 2004. [retrieved: May, 2015]. Available from : https://msdn.microsoft.com/en-us/library/ff647958.aspx#intpatt-ch05_broker

[7] C. S. Chapman and L. Kihn, "Information system integration, enabling control and performance," Elsevier, Accounting, Organizations and Society Volume 34, Issue 2, Feb. 2009, pp. 151–169, ISSN: 0361-3682

[8] K. Ravi and M. Robinson, "E-business 2.0: Roadmap for Success," Addison-Wesley, 2001, ISBN 0201721651

[9] A. Vasconcelos, M. Mira da Silva, A. Fernandes, and J. Tribolet, "An information system architectural framework for enterprise application integration," System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on , vol., no., Jan. 2004, pp.9, doi: 10.1109/HICSS.2004.1265551

[10] Microsoft patterns and practices, Message broker, June. 2004. [retrieved: May, 2015]. Available from : https://msdn.microsoft.com/en-us/library/ff648849.aspx

[11] Microsoft patterns and practices, Message bus, June. 2004. [retrieved: May, 2015]. Available from : https://msdn.microsoft.com/en-us/library/ff647328.aspx

[12] D. Chappell, "Enterprise Service Bus," O'Reilly, Jun. 2004, ISBN 0-596-00675-6

[13] D. L. Parnas, "On the criteria to be used in decomposing system into modules," Communications of the ACM, Vol. 15, No. 12, Dec. 1972, pp.1053–1058.

[14] IEEE Std 610.12-1990, "Glossary of software engineering terminology," in software engineering standards collection, IEEE CS Press, Dec. 1990, doi: 10.1109/IEEESTD.1990.101064

[15] V. Stavridou,"Integration in software intensive systems," Journal of Systems and Software, vol. 48, Issue 2, Oct. 1999, pp. 91–104, doi:10.1016/S0164-1212(99)00049-7

[16] D. Chena, G. Doumeingtsb, and F. Vernadatc,"Architectures for enterprise integration and interoperability: Past, present and future," Computers in Industry, vol. 59, Issue 7, Sep. 2008, pp. 647–659, doi:10.1016/j.compind.2007.12.016

[17] J. Nurmilaakso and J. Kauremaa,"Business-to-business integration: Applicability, benefits and barriers in the telecommunications industry," Computers in Industry, vol. 63, Issue 1, Jan. 2012, pp. 45-52, doi:10.1016/j.compind.2011.10.006

 ISBN: 978-1-61208-448-0

Medical Device Software as a Subsystem of an Overall Medical Device

The MDevSPICE® Experience

Fergal McCaffery, Marion Lepmets, Paul Clarke
Regulated Software Research Centre & Lero,
Department of Computing & Mathematics
Dundalk Institute of Technology
Co. Louth, Ireland
Email{fergal.mccaffery, marion.lepmets, paul.clarke}@dkit.ie

***Abstract*—Embedded software is a sub-system that needs to be integrated with the electrical and mechanical subsystems for a functional medical device to be developed and marketed. In order to be able to develop a medical device system through integrating its sub-systems, the complete system requirements should be known at the start of the project and managed throughout development. Software requirements are then derived from the systems requirements. We have developed and piloted a medical device software process assessment framework called MDevSPICE® that integrates processes from various medical device software standards as well as generic software development standards. This paper describes how the MDevSPICE® framework has been designed so as to enable medical device software developers to produce software that will be safe and easily integrated with other sub-systems of the overall medical device. We also describe the lessons learned from piloting MDevSPICE® in the medical device industry and challenges medical device software developers meet in tracing requirements and risks to and from the system level.**

Keywords- software integration; medical device software; MDevSPICE®; medical device risks; medical device.

I. INTRODUCTION

Safety-critical software systems are increasingly affecting our lives and welfare as more and more software is embedded into medical devices, cars and airplanes each day. New approaches and international standards are being developed to ensure the safety of these systems before they are delivered. In order to market a medical device, for example, the manufacturer has to satisfy a number of regional regulatory requirements commonly achieved by following international standards and guidance issued by international standardizing bodies and regional regulatory authorities. To help software companies in the medical device domain in their attempt to reach regulatory compliance, we have developed an integrated framework of medical device software development best practices called MDevSPICE®. This framework integrates generic software development best practices with medical device standards' requirements enabling robust software process assessments to be performed. The "SPICE" in MDevSPICE® reflects its foundation in the ISO/IEC 15504 (SPICE) [25] series of standards for process assessment. Through validating the MDevSPICE® framework we provide evidence of the importance of traceability between the system and software levels of development – and explain how the establishment of robust requirements interfacing between these levels can support more effective software integration

In Section II, we describe the regulatory requirements medical device software development companies face before they are able to market their devices. In Section III we describe the development of the MDevSPICE® framework We then focus in Section IV on the lessons we learned when validating the framework in expert reviews and in industry through MDevSPICE® pilot assessments. We also discuss the importance of traceability between system and software development processes when developing an embedded medical device software system as it increases the safety and quality of the developed medical device. The paper concludes in section V.

II. MEDICAL DEVICE REGULATION

A medical device can consist entirely of software or have software as a component of the overall medical device system. In order to be able to market a medical device within a particular region it is necessary to comply with the associated regulatory demands. Two of the largest global bodies responsible for issuing and managing medical device regulation belong to the central governing functions of the US and EU. In the US, the Food and Drug Administration (FDA) issues the regulation through a series of official channels, including the Code of Federal Regulation (CFR) Title 21, Chapter I, Subchapter H, Part 820 [1]. Under US regulation, there are three medical device safety classifications: Class I, Class II and Class III. The medical device safety classification is based on the clinical safety of the device. Class I devices are not intended to support or sustain human life, and may not present an unreasonable risk of harm. A thermometer is a Class I device. Class II devices could cause damage or harm to humans. An example of a Class II medical device is a powered wheelchair. Class III medical devices are usually those that support or sustain human life, and are of significant importance in the prevention of human health impairment. An example of a Class III device is an implantable pacemaker. All

 ISBN: 978-1-61208-448-0

implantable devices are Class III medical devices as the surgery required carries with itself additional high risks from anesthesia and possible infections that go beyond the safety risks of the medical device.

In the EU, the corresponding regulation is outlined in the general Medical Device Directive (MDD) 93/42/EEC [2], the Active Implantable Medical Device Directive (AIMDD) 90/385/EEC [3], and the *In-vitro* Diagnostic (IVD) Medical Device Directive 98/79/EC [4] - all three of which have been amended by 2007/47/EC [5]. Similarly to the US, the EU device safety is also based on the clinical safety of the device embodying similar classifications and limitations, where Class I in the EU corresponds to Class I in the US, Class IIa and IIb to Class II, and Class III to Class III.

A further safety classification applies to the software in medical devices as outlined in IEC 62304:2006 [6], where the safety classification is determined based on the worst possible consequence in the case of a software failure. In the case of failure of software that is of safety Class A, no injury or damage to health of a patient can occur. When software of safety class B fails, injury may occur but it is not serious or life-threatening. Class C medical device software is of highest risk and in the case of failure of such software death or serious injury can happen. Depending on the functionality of software within the medical device, the software safety classification may vary from the overall medical device safety class. When software is of critical functionality of the medical device, it will carry the same classification as the device, i.e., Class C software in Class III device. The safety classification of software may be lower but cannot be higher than the overall medical device safety class, e.g., software of safety Class B, may be embedded in Class III device but there cannot be software of safety Class C, in a Class I or Class II device.

Medical device manufacturers in the US as well as in EU must satisfy quality system requirements to market their developed devices. In the medical device domain, ISO 13485:2003 (ISO 13485 from hereon) [7] outlines the requirements for regulatory purposes from a Quality Management System (QMS) perspective in medical device domain. ISO 13485, which is based on ISO 9001 [8], can be used to assess an organization's ability to meet both customer and regulatory requirements in the medical device domain. ISO 13485 does not, however, include requirements for software development. IEC 62304, which can be used in conjunction with ISO 13485, does offer a framework for the lifecycle processes necessary for the safe design and maintenance of medical device software. As a basic foundation, IEC 62304 assumes that medical device software is developed and maintained within a QMS such as ISO 13485, but does not require an organization to be certified against ISO 13485. Therefore, IEC 62304 can be considered to be a software development specific supplement to ISO 13485, similar to ISO 90003 for ISO 9001.

IEC 62304 is based on ISO/IEC 12207:1995 [9] which although a comprehensive standard for software development lifecycle processes has effectively been decommissioned following the publication of the more extensive ISO/IEC 12207:2008 [10]. Furthermore, other developments in the ISO and IEC communities for software development, such as ISO/IEC 15504 [11], have provided significant additional levels of software process detail to support ISO/IEC 12207:2008. IEC 62304 is a critical standard for medical device software developers as it is the only standard that provides recommendations for medical device software implementations based on the worst consequences in the case the software failure causing hazards. Furthermore, for general medical device risk management, IEC 62304 is used in conjunction with ISO 14971 [12] and IEC 80002-1 [13] that provides guidance on the application of ISO 14971 for software development.

Since IEC 62304 considers a medical device system to consist of software as a sub-system, the system or product level requirements are not included within IEC 62304 but instead within the medical device product standard IEC 60601-1 [14]. Due to the increasing importance of usability of devices within the medical device industry, organizations should also adhere to the medical device usability engineering process requirements outlined in IEC 62366 [15]. When Medical Device Directives were amended in 2007 [5], it allowed standalone software to be defined as a medical device in its own right. Previously, software had always been seen as a subsystem embedded in a medical device. This amendment revealed a gap in international standards as none of the published standards were addressing the concerns for standalone software as a medical device. Today, IEC CD 82304-1 [16] applies to the safety of health software that is designed to operate on general purpose IT platforms and that is intended to be placed on the market without dedicated hardware, e.g., iPad applications.

All companies planning to market a medical device in the United States need to register their product with the US FDA. Most Class I devices can be self-registered but most Class II devices require a 510(k) submission. For Class III devices, a Pre-Market (PMA) submission is needed. To support manufacturers in addressing the relevant guidance, the FDA has issued an overview of their guidance documents for medical device manufacturers and software developers [17]. The FDA Guidance on Premarket Submissions [18] provides guidance and recommendation for premarket submissions for software devices, including standalone software applications and hardware-based devices that incorporate software. Premarket submission includes requirements for software-related documentation that should be consistent with the intended use of the Software Device and the type of submission. The FDA Guidance on Off-The-Shelf Software Use in Medical Devices [19] was published in 1999 with the purpose of describing the information that should be provided in a medical device application that uses off-the-shelf (OTS) software. Many of the principles outlined in this guidance document may also be helpful to device manufacturers in establishing design controls and validation plans for use of off-the-shelf software in their devices. The FDA General Principles of Software Validation [20] outlines general validation principles that the FDA considers to be applicable to the validation of medical device software or the validation of software used to design, develop, or manufacture medical devices. This guidance describes how

 ISBN: 978-1-61208-448-0

certain provisions of the medical device Quality System regulation apply to software. The scope of this guidance is somewhat broader than the scope of validation in the strictest definition of that term to support a final conclusion that software is validated.

The challenge that software development companies in the medical device domain face when they want to market a device is in the adherence to a large number of regulatory requirements specified in various international standards (that can often become overwhelming). In order to help these companies better prepare for demanding and costly regulatory audits, we developed the MDevSPICE® framework. MDevSPICE® includes requirements from the previously mentioned standards and guidance documents rendering the task of regulatory compliance much less complex. Following is a description of the development of the MDevSPICE® framework that integrates the requirements from various international medical device standards and guidance documents with the generic software development best practices while providing a possibility to assess processes.

III. MDEVSPICE® FRAMEWORK

This section describes the development of the MDevSPICE® process reference model, how MDevSPICE® provides support for integration, and how MDevSPICE® was piloted in industry.

A. *Development of the MDevSPICE® Process Reference Model*

A process reference model (PRM) describes a set of processes in a structured manner through a process name, process purpose and process outcomes where the process outcomes are the normative requirements the process should satisfy to achieve the purpose of the process. In order to develop a PRM that integrates requirements from various standards allowing the processes to be evaluated in terms of their achievement of their purpose statements, we followed the format of the process description illustrated in ISO/IEC 24774 [21]. With that in mind, we first mapped and integrated the requirements from ISO/IEC 12207:2008 and IEC 62304 into what today is called the PRM for IEC 62304 that also reflects the updates to ISO/IEC 12207 from the 1995 to the 2007 version. A systematic approach of memoing and constant comparison, which is based on the principles of Grounded Theory [22] was followed when developing the PRM, further details of which are to be found in [23]. The Process Reference Model of IEC 62304 was published in June 2014 as IEC TR 80002-3 [24].

While IEC 62304 describes only the software life cycle processes, additional processes should be in place for system development in the case where software is not embedded as part of an overall medical device. These additional processes were derived from ISO/IEC 12207:2008. Design and development related requirements from ISO 13485 and ISO 14971 were also added to the MDevSPICE® Process Reference Model. Both ISO 13485 and ISO 14971 are *de facto* standards for medical device software organizations. ISO 13485 requirements are primarily related to system level processes and ISO 14971 is concerned with risk management (and therefore aligned with the Software Risk Management process of the PRM.

The final MDevSPICE® PRM consists of 23 processes of which 10 are system life cycle processes, 8 are software life cycle processes and the remaining 5 support both the system and life cycle processes as can be seen in Figure 1.

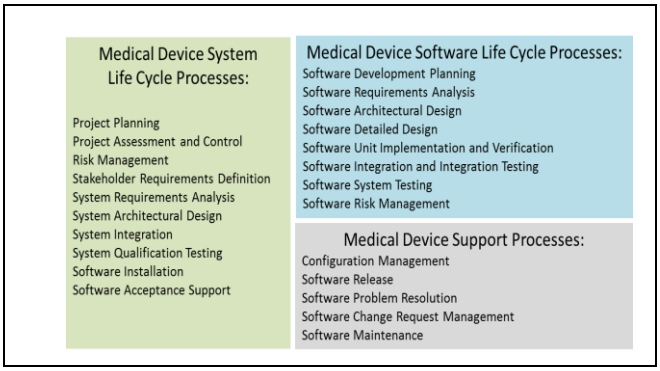


Figure 1. Processes of MDevSPICE® PRM

The MDevSPICE® PRM was then extended with additional elements to create a process assessment model (PAM). The aim of the MDevSPICE® PAM is to provide a comprehensive model for assessing the software and systems development processes against the widely recognized medical device regulations, standards and guidelines that a software development organization in the medical device domain has to adhere to. The MDevSPICE® PAM, similar to ISO/IEC 15504-5 (SPICE) [25], has two dimensions – a process dimension and a capability dimension. The process dimension lists three groups of processes from various models and standards, i.e., systems life cycle processes, software life cycle processes and support processes. Each process is described in terms of a Process Name, Process Purpose, Process Outcomes, Base Practices, Work Products and Work Product Characteristics.

The MDevSPICE® PRM is based on IEC 62304, ISO/IEC 12207:2008, ISO 14971 and ISO 13485. The MDevSPICE® PAM then extends this PRM with base practices and work products, some of the latter also being normative as they are described in IEC 62304, ISO 14971 or ISO 13485 as requirements. Where process outcomes are derived from ISO/IEC 12207:2008, their corresponding base practices and work products are derived from ISO/IEC 15504-5. Where process outcomes are derived from ISO 14971, their corresponding base practices are derived from IEC 80002-1. In addition to these sources, FDA guidance on premarket submissions, software validation and off-the-shelf software have been added to the informative base practices where the base practice did not already address the requirements of the corresponding FDA guidance. Product safety requirements have been added to the MDevSPICE® PAM from both IEC 60601-1 and IEC CD 82304-1, while

 ISBN: 978-1-61208-448-0

the usability engineering requirements have been incorporated from IEC 62366.

The capability dimension of the MDevSPICE® PAM is derived directly from ISO/IEC 15504 together with the Capability Levels, Process Attributes, Generic Practices, Generic Resources and Generic Work Products.

While integrating processes from different standards and guidance documents for the MDevSPICE® PRM and PAM , a focus on the traceability between and within system and software life cycle processes was maintained [26]. Both the FDA General Principles of Software Validation [20] and ISO/IEC 12207 [10] incorporate traceability of risks, changes and requirements throughout the development life cycle. This interaction and traceability of requirements is a key enabler of subsequent integration, and it has a vital role to play in raising the safety of medical device software.

B. MDevSPICE® Framework support for integration

The MDevSPICE® framework contains key facilities for integrating medical device software. Since MDevSPICE® is grounded in IEC 62304, the software sub system decomposition is consistent with the requirements of IEC 62304, meaning that the language of a *software unit*, a *software item* and a *software system* is adopted.

A software system is the integrated collection of software items to accomplish a specific function or set of functions; a software item is any identifiable part of a computer program; and a software unit is a software item that is not subdivided into other items. This software system hierarchy has an important role to play when a software developer wishes to decompose a system into parts of varying software safety classification. A benefit of such decomposition is that those parts of the software subsystem that are vital for safety (and which require additional safety activities when under development) can be isolated until they are later integrated with the other software components. It is also important that when the components are integrated that the safety implications are reflected in test cases that are pre-defined, then tested and the results are checked to ensure that they match the expected results. Otherwise sign-off cannot take place at the various levels – unit tests, integration tests and system tests.

Integration activities in the MDevSPICE® framework start by integrating software units into software items, and thereafter software items are further integrated with each other (and possibly with other units as well) into the software subsystem (which in turn is integrated into the overall medical device system). There are therefore several levels of integration and they must take into consideration the safety implications at each step. It is further the case that the bi-directional traceability of requirements (including requirements related to safety) from the product level right down to the individual software units is supported in MDevSPICE® thus further supporting medical device software safety at the integration stage and beyond.

C. Piloting the MDevSPICE® Framework

The MDevSPICE® framework has been validated in various stages of its development by different parties through both international expert review and industrial trials. The foundation of the MDevSPICE® PAM, IEC TR 80002-3 (the development of which was led by the authors), was published after several iterations of development and analysis by the standardization working group responsible for the publication of IEC 62304 (i.e., ISO/IEC Sub-Committee 62A, Joint Working Group 3). An international standard is published only after the national delegates of the standard's working group have agreed on every detail of that standard.

In addition to working with the international medical device standards community, the MDevSPICE® PAM has also been developed together with and analyzed by experts in process assessment working group 10 of ISO/IEC Joint Technical Committee 1, Sub-Committee 7, responsible for the development and maintenance of the series of process assessment standards. These standards are currently being revised from ISO/IEC 15504 series to ISO/IEC 330xx series of standards. MDevSPICE® framework keeps abreast of these updates as well as with the updates of any other standard and guidance document information from which is contained in the MDevSPICE® framework.

Upon successful completion of international expert review, the MDevSPICE® process assessment framework was then validated in the medical device software industry through pilot assessments over the past two years. MDevSPICE® process assessments were conducted in different types of organizations: (1) a small software company wishing to supply software to a large medical device manufacturer who wants them to demonstrate that they are capable of developing safe medical device software and provide the medical device manufacturer with a feeling that they will not jeopardize the safety of their overall medical device or the reputation of their organization; (2) three different assessments (across a 2 year period) were performed in two different international sites of a multinational medical device manufacturer who wants to ensure that they are incorporating best practices within their software development processes to not only achieve regulatory compliance but also reduce the likelihood of recalls through developing better quality and more robust software; (3) a software development company seeking to achieve regulatory compliance against IEC 62304 so that they can become medical device software suppliers; and (4) a large automotive manufacturer experienced in developing safety-critical embedded automotive software now wishing to also develop embedded medical device software.

IV. LESSONS LEARNED FROM PILOTING MDEVSPICE®

As a result of the MDevSPICE® pilot assessments we have witnessed different types of needs and challenges in companies where MDevSPICE® pilot assessments were conducted.

 ISBN: 978-1-61208-448-0

In companies that manufacture medical devices as well as develop the embedded software for their devices, the traceability and integration between system and software life cycle processes is well managed. This might be due to systems and software engineers working closely together for safe medical device development where the software developers are aware of the system risks and requirements.

For software companies that develop software for large medical device manufacturers though, it can be difficult for the third party software developers to become aware of the overall system level requirements and risks before software development project commences. When the system requirements are not provided to the software developers, this hinders the traceability engineering and integration of the subsystems of the medical device. But medical device manufacturers working on innovative devices are sometimes reluctant to provide their software subcontractors with the details of their device design as this could jeopardise device novelty or competitive advantage. Yet, the safety risks related to the performance of medical devices can outweigh the business risks, which can be diminished with proper legal knowhow, for the medical device manufacturer. We would therefore recommend medical device manufacturers to more openly communicate with their software subcontractors in order to best support risk and requirements management throughout their device design – even if this only encompasses those product requirements which are related to software requirements (and especially those which are safety related). The ultimate goal for all device providers is to have a safe medical device on the market and not risk liability or damage of their brand as a result of a recall of a faulty device.

V. CONCLUSION

Safety-critical domains are characterized by heavy regulatory demands that companies have to adhere to before they can place their devices on the market. Regulatory audits are conducted regularly to evaluate these companies and the safety of their devices. In order to pass these audits, medical device manufacturers have to ensure that all regulatory requirements have been adhered to in the design and development of each of the medical device subsystems.

In this paper, we have explained the medical device regulatory requirements and the related standards and guidance documents, and how MDevSPICE® addresses all of these concerns in a single medical device software framework. Key to developing this framework was an acknowledgement that the overall medical device requirements have a direct impact on the safety of the device, and it is therefore critical that top level product requirements are fully realized in the software system and its related requirements. This can be especially difficult to achieve in environments where device manufacturers may choose to outsource software development without necessarily sharing all top level product requirements subcontractors. To address this critical interface, the MDevSPICE® framework incorporates not just software development lifecycle processes but also system level process. Hence, system requirements that have an impact on software requirements are identified in MDevSPICE®, and through the implementation of bilateral requirements traceability, decisions taken during the software subsystem development are fed back to the top level system requirements – thus providing a closed loop for requirements management which can help to raise the overall safety of the device.

Requirements management is a key activity when integrating software subsystems and when integrating software into higher level systems (such as is the case for embedded medical device software). Closely aligned to requirements management is the management of safety related risks, and these too are supported in a bilateral top-to-bottom (system to subsystem) mechanism in MDevSPICE, with the result that software integration for medical devices is conducted in an environment that fully harmonises both general requirements and safety concerns. While such steps may not be desirable or economically viable in the case of general non-safety critical software, they do provide a mechanism for thorough requirements management, even in the case where subcontracting is undertaken – and this is a positive development in terms of supporting robust and effective software integration on all levels.

ACKNOWLEDGMENT

This research is supported by the Science Foundation Ireland Principal Investigator Programme, grant number 08/IN.1/I2030 (the funding of this project was awarded by Science Foundation Ireland under a co-funding initiative by the Irish Government and European Regional Development Fund),and by Lero - the Irish Software Research Centre (http://www.lero.ie) grant 10/CE/I1855 & 13/RC/20194.

REFERENCES

[1] FDA. Chapter I - Food and drug administration, department of health and human services subchapter H - Medical devices, Part 820 - Quality system regulation. Available from: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=820. Last date accessed - 28th May 2015.

[2] Directive 93/42/EEC of the European Parliament and of the Council concerning medical devices. 1993. European Commission, Brussels, Belgium. pp. 43.

[3] Council directive 90/385/EEC on active implantable medical devices (AIMDD). 1990. Brussels, Belgium. pp. 35.

[4] Directive 98/79/EC of the european parliament and of the council of 27 october 1998 on in vitro diagnostic medical devices. 1998. Brussels, Belgium. pp. 43.

[5] Directive 2007/47/EC of the European Parliament and of the Council concerning medical devices. 2007. EC: Brussels, Belgium. pp. 35.

[6] IEC 62304: Medical Device Software - Software Life-Cycle Processes. 2006. IEC: Geneva, Switzerland. pp. 151.

[7] ISO 13485: Medical Devices - Quality Management Systems - Requirements for Regulatory Purposes. 2003. ISO: Geneva, Switzerland. pp. 57.

[8] ISO 9001:2000 - Quality Management Systems - Requirements. 2000. Geneva, Switzerland. pp. 27.

[9] ISO/IEC 12207:1995 - Information Technology - Software Life-Cycle Processes. 1995. ISO/IEC: Geneva, Switzerland. pp. 106.

[10] ISO/IEC 12207:2008 - Systems and Software Engineering - Software life cycle processes. 2008. ISO/IEC: Geneva, Switzerland. pp. 138.

 ISBN: 978-1-61208-448-0

[11] ISO/IEC 15504-2. 2004. Information technology - Software process assessment - A reference model for processes and process capability, in 15504.

[12] ISO 14971 - Medical Devices - Application of Risk Management to Medical Devices 2009. ISO: Geneva, Switzerland. pp. 82.

[13] IEC TR 80002-1 - Medical Device Software - Part 1: Guidance on the Application of ISO 14971 to Medical Device Software. 2009. IEC: Geneva, Switzerland. pp. 58.

[14] IEC 60601-1 - Medical electrical equipment – Part 1: General requirements for basic safety and essential performance 2005. IEC: Geneva, Switzerland. pp. 20.

[15] IEC 62366 - Medical devices - Application of usability engineering to medical devices. 2007. IEC: Geneva, Switzerland. pp. 104.

[16] IEC 82304-1: Health software -- Part 1: General requirements for product safety. 2012. IEC: Geneva, Switzerland. pp. 30.

[17] FDA. 2015. Guidance Documents (Medical Devices and Radiation-Emitting Products).

[18] FDA Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices. 2005. FDA: USA. pp. 20.

[19] FDA's Guidance for industry, FDA reviewers and compliance on - Off-The-Shelf Software Use in Medical Devices. 1999. FDA: USA. pp. 26.

[20] FDA's General Principles of Software Validation; Final Guidance for Industry and FDA Staff. 2002. FDA: USA. pp. 43.

[21] ISO/IEC 24774 - Systems and Software Engineering - Life Cycle Management - Guidelines for Process Description. 2010. Geneva, Switzerland. pp. 15.

[22] B. Glaser, and A. Strauss 1976. The Discovery of Grounded Theory: Strategies for Qualitative Research, ed. A.d. Gruyter. Hawthorne, NY, USA.

[23] M. Lepmets,P. Clarke, F. McCaffery, A. Finnegan, and A. Dorling 2014. Development of a Process Assessment Model for Medical Device Software Development, in Industrial Proceedings of the 21st EuroSPI Conference: Luxembourg. p. 2.25-2.35.

[24] IEC TR 80002-3: Medical device software -- Part 3: Process reference model of medical device software life cycle processes (IEC 62304). 2014. IEC: Geneva, Switzerland. pp. 23.

[25] ISO/IEC 15504-5. Information technology - process assessment - Part 5: an exemplar process assessment model. 2012. pp. 211.

[26] G. Regan, M. Biro, F. Mc Caffery, K. McDaid, and D. Flood 2014. A Traceability Process Assessment Model for the Medical Device Domain, in EuroSPI 2014, Springer: Luxembourg. p. 206-216.

 ISBN: 978-1-61208-448-0

Enterprise Integration Modeling

A Practical Enterprise Data Integration and Synchronization Solution

Mihaela Iridon
Cândea LLC
Dallas, TX, USA
e-mail: iridon.mihaela@gmail.com

***Abstract*— As line-of-business software systems take shape and evolve over time within an organization, so does the need for such systems to interact with each other and exchange data, making it imperative to design flexible, scalable integration architectures and frameworks to support a robust and well-performing enterprise system. System integration is a multi-faceted undertaking, ranging from low-level data sharing (Shared Repository or File Sharing), to point-to-point communications (Remote Procedure Invocation via Service Orientation), to decoupled data exchange architectures (Messaging). It is common to build entire integration sub-systems responsible not only for exchanging information between systems (commands and notifications) but also for potentially more complex business logic orchestration across the entire enterprise (Message Broker). This paper is contemplating a practical data notification and synchronization integration solution that allows multiple enterprise domains to share data that is critical for business operations. The article presents a real-world integration architecture achieving this business objective, together with the corresponding system models and design artifacts, and shows how the data integration is realized using a broker-based messaging approach employing various enterprise integration patterns.**

Keywords-Enterprise integration; system modeling; data integration; canonical model; integration patterns.

I. INTRODUCTION

Within an enterprise, system integration solutions are almost always designed and implemented as an afterthought, as an attempt to build or to expand a new or existing enterprise architecture comprised of heterogeneous legacy system. It may be safe to say that most companies do not start off with an integrated enterprise architecture but rather a core domain (also referred to as a vertical), which will eventually grow and become part of a larger enterprise system. In many cases, such integration is achieved by employing various off-the-shelf integration products, such as Microsoft's BizTalk [7] or TIBCO.

Software system integration comes in different flavors, depending on the business objectives, the overall enterprise architecture, and ultimately the realization approach chosen. In Section II we will investigate these driving factors and then present a concrete implementation approach and its models in Section III, as it has been proposed and adopted by a provider of the nation's largest portfolio of benefit and payroll products and services designed to help more than 200,000 small and medium-sized businesses.

This paper presents a data integration and synchronization blueprint aimed at implementing the "Maintain Data Copies" data integration pattern [8] by means of a decoupled integration mechanism realized on a custom broker-based messaging architecture [10] [12]. The data payloads exchanged between the loosely coupled sub-systems abide to a *ubiquitous* integration language, referred to as the *canonical model* [7] as described in Section IV. This model is the unified abstraction of the data structures that must be shared and synchronized between these systems.

II. COMPARING AND CONTRASTING FUNCTIONAL AND DATA INTEGRATION

When building a large enterprise software system by bringing together multiple domain applications, the first question that must be answered involves the level of abstraction at which the integration specifications are being defined: Do the sub-systems only need the data that allows them to carry out their own functions, or do they also require access to cross-domain exposed functional features? In other words, should a system expose data only or features as well?

The answers to these questions will determine the type of integration that must be realized: data or functional integration, and, perhaps even further, it will help discern between the need of a flexible, lightweight, loosely-coupled integration architecture and one that adds enterprise features and interactions, transcending domain system boundaries. It is also possible that, in some cases, a hybrid approach may be pertinent, either to realize a quick and simple integration with a narrower scope (e.g. a test product implementation), or to overcome deep architectural and data model discrepancies between the existing systems. In this case, the solution must fulfill some imperative enterprise needs - whether they are related to exposing new system features in a short amount of time or at a lower cost until further market research proves the worthiness of additional funding for a comprehensive, scalable, extensible, and suitable solution.

A. Functional Integration

This type of integration involves exposing data and behavior [9] to systems that participate in the integration in order to trigger or invoke business features exposed by these systems. Usually, a pure Service Oriented Architecture (SOA) [3] [4] would be the simplest architectural approach

 ISBN: 978-1-61208-448-0

that could realize this requirement, but it would introduce system coupling and would not be easily scalable [5]. Web Services implement in effect the Remote Procedure Invocation integration pattern paradigm [7] and this implies mutual awareness of the presence of – and the functionality provided by - each of the integrating systems.

Complexity becomes apparent when more than two systems must interact at a logical and/or functional level of abstraction by invoking these exposed features and generating chattiness across the network, or when systems evolve, possibly threatening the stability of the integration contracts and hence of the solution. Several options are available to alleviate these problems, from architectural ones to following best practices and proper functional decomposition and service encapsulation, and eventually to making the proper technology choices [4].

B. Data Integration

This type of integration assumes that the various integrating systems were not designed to work together [1], and that they do not have direct access to the entire enterprise data but only to that which they provision directly. These systems were built in order to fulfill certain functional and business requirements, rather than architectural ones. It is also possible that some systems were acquired at a later time (e.g., corporate mergers, third-party software acquisitions, etc.)

Given that the systems evolved independently, enabling them to interoperate using multiple copies of the enterprise data (i.e., multiple data sources) while providing enterprise-level business features in a unified fashion is problematic, since there is no single source of truth and, potentially, no single source of data entry. Multiple applications may allow users to enter the same type of data from different user interfaces that sit atop of different business/logic layers and, consequently, different data sources.

Achieving this type of data integration can rely on either custom solutions (for example, involving an enterprise service bus), or commercial tools (such as implementations of a Master Data Management system), which may expedite the time-to-market of such an integration, sometimes at lower costs than custom solutions [2] [7]

III. A PRACTICAL DATA INTEGRATION AND SYNCHRONZATION SOLUTION

Consider three major business domains, Human Resources (HR), Payroll, and Benefits. The common ground for all three is the demographic data that defines the companies (or clients) that these systems are servicing and their employees. As is quite often the case, neither domain was built with a true enterprise vision in mind, neither architecturally, nor functionally. Yet the main enterprise data on employees and clients served must be shared across all domains when multiple data copies exist, one per domain. These data sources were designed for a very specific purpose, making it prohibitively expensive to refactor the systems' layers and the business applications so that they rely on a single source of truth – a unified data source across the enterprise. A solution employing Master Data Management (MDM) tools has been evaluated but the business requirements did not warrant such elaborate implementations for this particular case. The proposed and agreed upon solution was to implement the "Maintain data copies" data integration pattern [8] by means of a custom scalable and extensible middleware architecture (or integrating layer [10]), reusable frameworks and models, and carefully-chosen technologies, to fulfill the business need of providing multiple services (HR, payroll, and benefits) to an array of small to large size clients.

The following subsection presents the main models of the proposed integration solution, where data notifications are being exchanged between the various domains via a broker-based messaging architecture, using various enterprise integration patterns, as depicted in the EAI pattern mapping diagram in Figure 4. The data payload for these messages is wrapped inside a context-based notification model, allowing participating systems to take the appropriate action – based on their own domain rules – using the data received from the message broker. The individual domain systems are not aware of each other, only of the message broker through which they communicate.

A. The Integration Models

All models, structural and behavioral, included in this paper are excerpts from the technical design specifications document created on behalf of the client's Enterprise Integration Solution [12] and they are being used hereby with permission from this client.

1) Structural Models: High-Level Enterprise Integration Architecture and Components

The integration middleware was designed as an extensible, highly-responsive, and scalable broker-based topology through which the integrating domain systems will exchange data notifications in near real-time and in a loosely-coupled fashion. The middleware is built on durable messaging frameworks, such as an enterprise service bus (ESB), queues, an entity mapping/correlation infrastructure, and various service endpoints (SOA).

The high-level component diagram (Figure 1) shows the three business verticals as clients to the enterprise services that provide access to features that implement cross-cutting concerns (logging, SSO, audit) while indirectly exchanging data notification messages among each other, without awareness of each other or the features they provide, using the integration middleware exposed via a service endpoint (i.e., the Data Notification Receiver Service). This design ensures system scalability and plasticity of the integration scope (data or functional), while hiding the actual technology specifics from the systems that participate in the integration.

2) Object/Data Models: The Canonical Model

The data notifications exchanged between the systems via the service-broker integration middleware is a two-layered object model, with (a) the actual data payload represented by the integration ubiquitous model, also referred to as the Canonical Model [7], and (b) the notification model which is wrapping (or encapsulating) the canonical model payload, adding context, source, and target details to the communication messages.

 ISBN: 978-1-61208-448-0

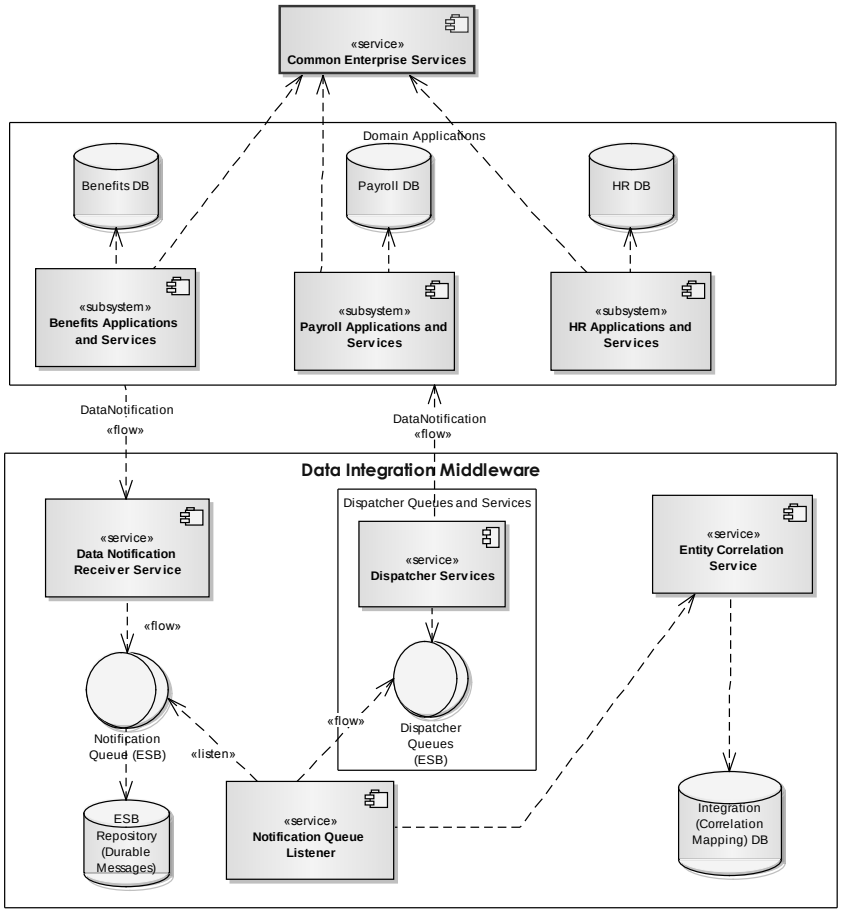


Figure 1. Overall enterprise integration topology: business verticals and integration middleware

This allows for a reusable notification model, where - by employing generic data types for the payload wrapped within the notification together with the appropriate inheritance (generic type inheriting from the non-generic type) – we can design any number of notification schemata that could encapsulate any business entity models inside a generic payload. The payload is domain-specific (or enterprise integration-specific in this case), whereas the notification model is domain-agnostic. This is depicted in the object model in Figure 2. The generic type T of the payload can be anything that one would define for a given domain: employee, client, address, benefit, participant, dependent, etc. In fact, a separate object model for the enterprise integration has been defined and is used in the implementation of this solution (see the Section IV for further details).

3) Behavioral Models: The Communication Model Describing the Enterprise Data Synchronization Process

For the implemented solution, the data notification exchange follows a very simple path through the hub-and-spoke (or star) integration middleware topology (Figure 3). However, the main challenge that had to be overcome is associating the business entities from one system to business entities in other systems, without introducing direct dependencies between these systems or awareness of other domains or domain-specific identifiers that – semantically – tie these enterprise entities together. For this purpose, an entity correlation service was introduced, using a separate repository of entity IDs that represent logically - or semantically - identical entities across the enterprise. Such correlations will be specified during an initial data setup process by administrative users or via custom automation tools and import/export facilities.

B. Noteworthy Features of the Integration Architecture

Some of the rather interesting features of this real-world integration solution are compiled below, grouped into functional and non-functional characteristics. Several design details are included to impart to the reader some level of context and comprehension of the architectural and technical approaches chosen.

1) Key Functional Attributes

a) Enterprise Data Coherence

Maintaining multiple data copies synchronized, all integrators become symmetrical systems of record for the core/common enterprise data.

 ISBN: 978-1-61208-448-0

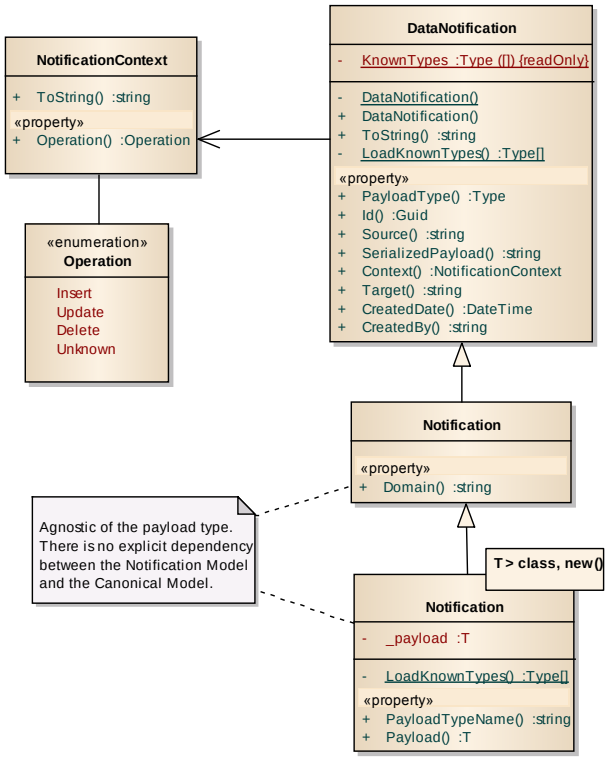


Figure 2. Data notification object model

All systems participating in the integration are able to notify the enterprise about relevant data updates in a particular line of business system without being aware of the other systems that might need this information or of the way in which this data will be consumed.

All systems participating in the integration will be notified of relevant data updates occurring across the enterprise via notifications that encapsulate data payloads following a normalized model. This in turn allows them to keep their own data copy synchronized with the data across the enterprise, while continuing to provision it independently, according to the domain's business rules.

b) Enterprise Functional Coherence

Specialized domain services offered to clients will continue to be managed and augmented within each individual vertical, without the need to cross domain boundaries, since all necessary data is available at the domain level, nearly real-time consistent with the enterprise data.

Decoupled and asynchronous notifications exchanged via the messaging broker keep systems unaware and independent of each other, while allowing the enterprise to grow as needed. Additional applications may be added; if these applications require their own data copy, they will start listening to notifications, and if they also support or require data updates that must be synced with other applications' data sources, then the new applications will also start sending notifications to the broker, to be dispatched and consumed throughout the enterprise, as needed.

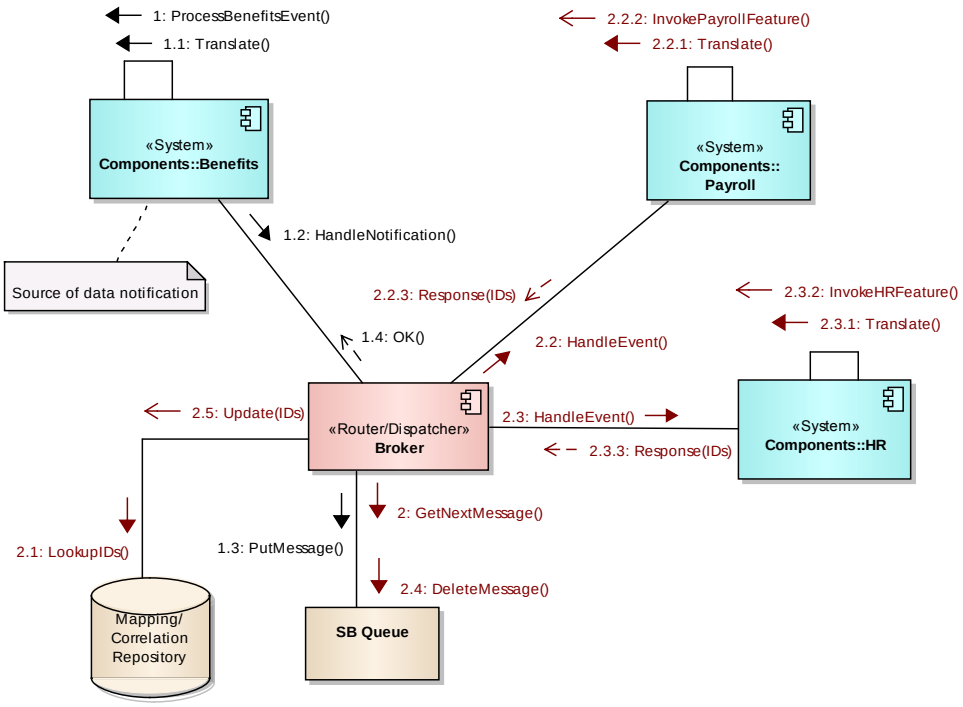


Figure 3. High-level integration communication model mapped to the service broker (star) topology

 ISBN: 978-1-61208-448-0

2) Key Quality Attributes

a) Scalability

Without any architectural changes to the integration framework or the domain systems, new systems can be added to this topology and can be enabled to participate in the integration (assuming they also use their own data source(s) that require continuous or occasional synchronization with the enterprise data). The only two-fold requirement is for these systems to expose a data notification service endpoint to handle enterprise notifications and to be able to raise and react to such data notifications appropriately, while being aware of the canonical model as the lingua franca of the enterprise integration.

b) Testability

Although additional testing frameworks for the integration components must be designed and built, individual systems will continue to be tested independently of each other or the integration middleware.

Components that simulate/generate notification traffic through the integration framework can be built to allow for independent testing of the service broker and the integration infrastructure.

c) Maintainability

The basic SOLID design principles employed, and most importantly the "separation of concerns" (or SoC) principle, ensure a highly maintainable architecture and codebase due to overall high cohesion and low coupling [5] [10].

Domain rules do not escape the boundaries of the system to which they belong, and similarly integration logic is isolated to the broker components and services.

d) High Availability

By employing load balancing and clustering around the integration services and the choice of technology (e.g., Service Bus Farm), the deployment topology was designed so as to ensure high availability as far as the integration components are concerned.

e) Performance

Assuming appropriate technology choices, the integration framework ensures a high throughput of notifications with minimal integration logic (i.e., entity correlation map lookup) required between the moment of receiving a notification and that of dispatching one.

For example, Microsoft's Windows Server Service Bus 1.1 (on premise) can process 20k messages/second (based on 1K message size) with an average latency of 20-25ms [11].

C. Enterprise Integration Patterns Mapping

The integration patterns [7] that were employed in designing and realizing the integration architecture are presented below. They can easily be mapped to the business verticals and integration middleware components as an overlay atop the simplified enterprise system block diagram, as seen in Figure 4.

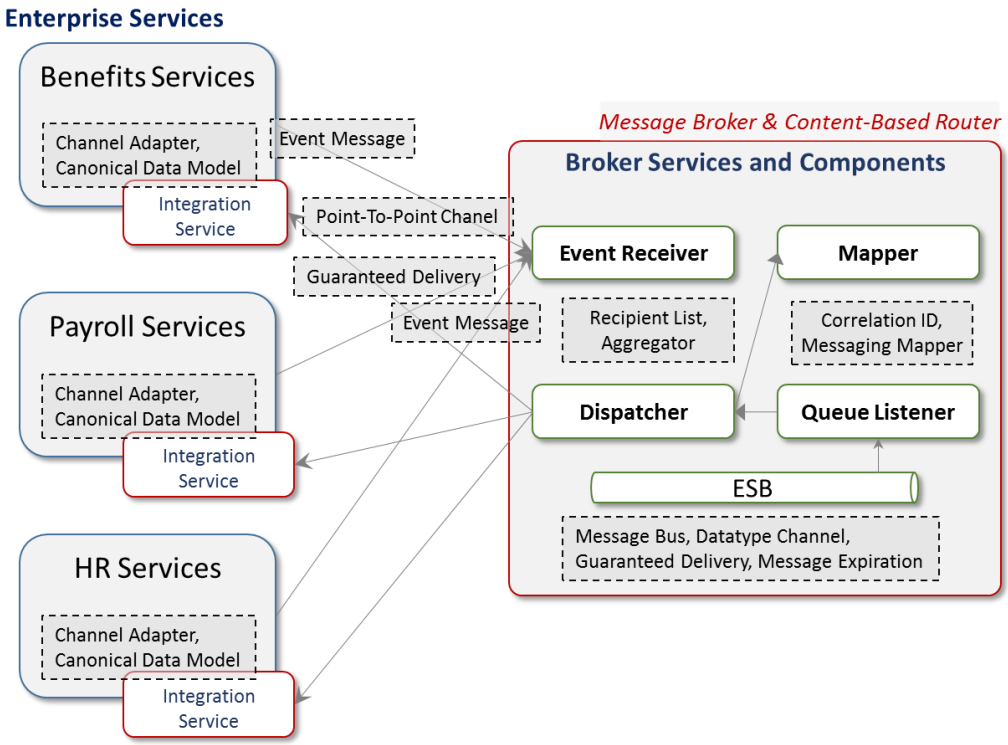


Figure 4. Mapping of enterprise integration patterns to domain systems and to integration middleware components

 ISBN: 978-1-61208-448-0

IV. SUMPPLEMENTARY INTEGRATION MODELS

A. *The Canonical Model's Base Class Details*

The Canonical Model integration pattern [7] has been the central theme of the solution implemented and is the only integration element that was allowed to permeate the enterprise (at each system's integration endpoints). This model can be envisioned as the ubiquitous integration language, which describes entities that are shared across the various domains of the enterprise. However, these entities in turn share data elements that are best modeled separately, as properties on base classes, using elemental inheritance, aggregation, and composition modeling concepts. For the domains in the presented case study, the need to support entity identifiers of different types, active timeframes, and traceability/audit features, led to the design of the model in Figure 5 where all domain entities inherit from the abstract class *EntityBase* shown in the center of the class diagram.

B. *The Canonical Model and the Main Integration Entities*

The main (aggregate root) entities in the integration's lingua franca are Group and Employee. They reflect the primary integration objective: keep Employee and Group demographics data in sync among all enterprise systems, by allowing each system to maintain and operate on their individual copy of the data. The model shown in Figure 6 is specific to the integration solution proposed for the client, aiming at integrating Benefits, Payroll, and Human Resources domains, more specifically for achieving the business goal of cross-selling services to various clients.

Noteworthy here is the fact that if we consider the canonical model as the domain of the integration, then it is following the anemic domain model design anti-pattern [6]. This is because these are simple data containers and do not encapsulate functionality as the integration framework's domain itself is behavior-less. The model's only purpose is to capture and transport data notifications across systems – so, from this (proper) perspective the model is abiding to the Data Transport Object (DTO) pattern of enterprise application architecture [5].

Generic functionality is exposed in the form of service operation contracts for handling notifications (whether a domain system raises a notification or must handle one), but no enterprise features are being implemented here, hence data representation and modeling is of essence and imperatively impacts the success of the proposed system integration solution.

C. *The Enterprise Integration Activity Model*

The overall system integration flow is modeled in the activity diagram in Figure 7, where the various integrating systems and the broker components are bounded by the vertical swim lanes, to indicate where activities and actions cross system boundaries. The diagram also shows how the correlation service is being employed to allow the integration framework to associate the same (logical) clients across domains by looking up and populating the appropriate domain identifiers, as part of the context that wraps the notification data payload passing through the broker.

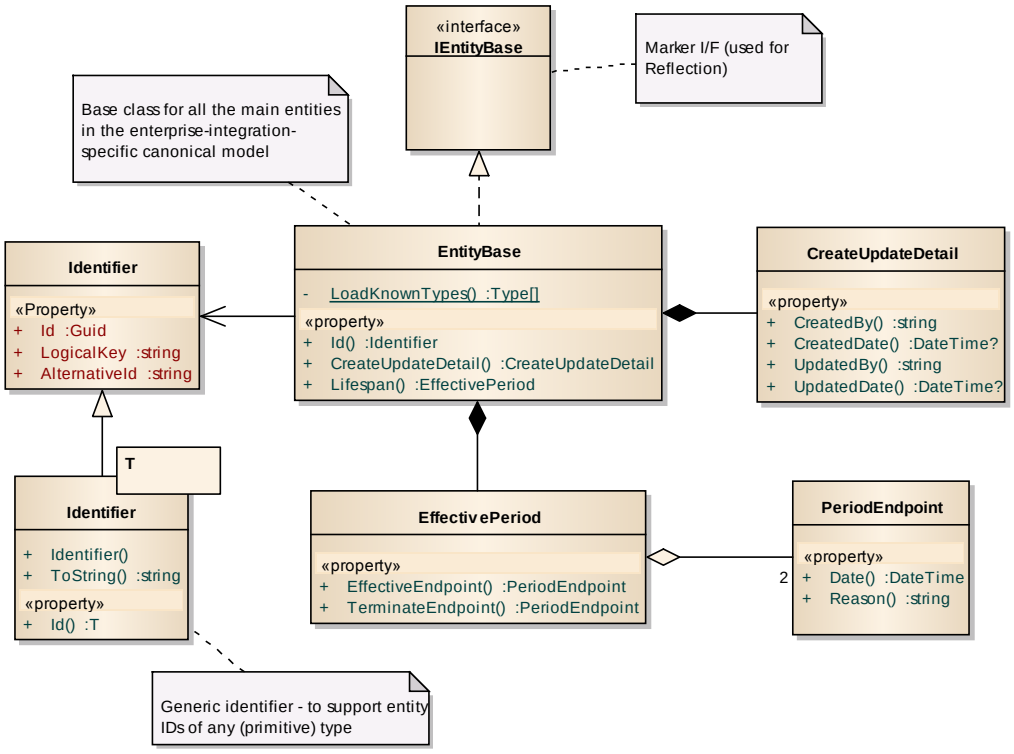


Figure 5. Base class and common elements for the canonical model types

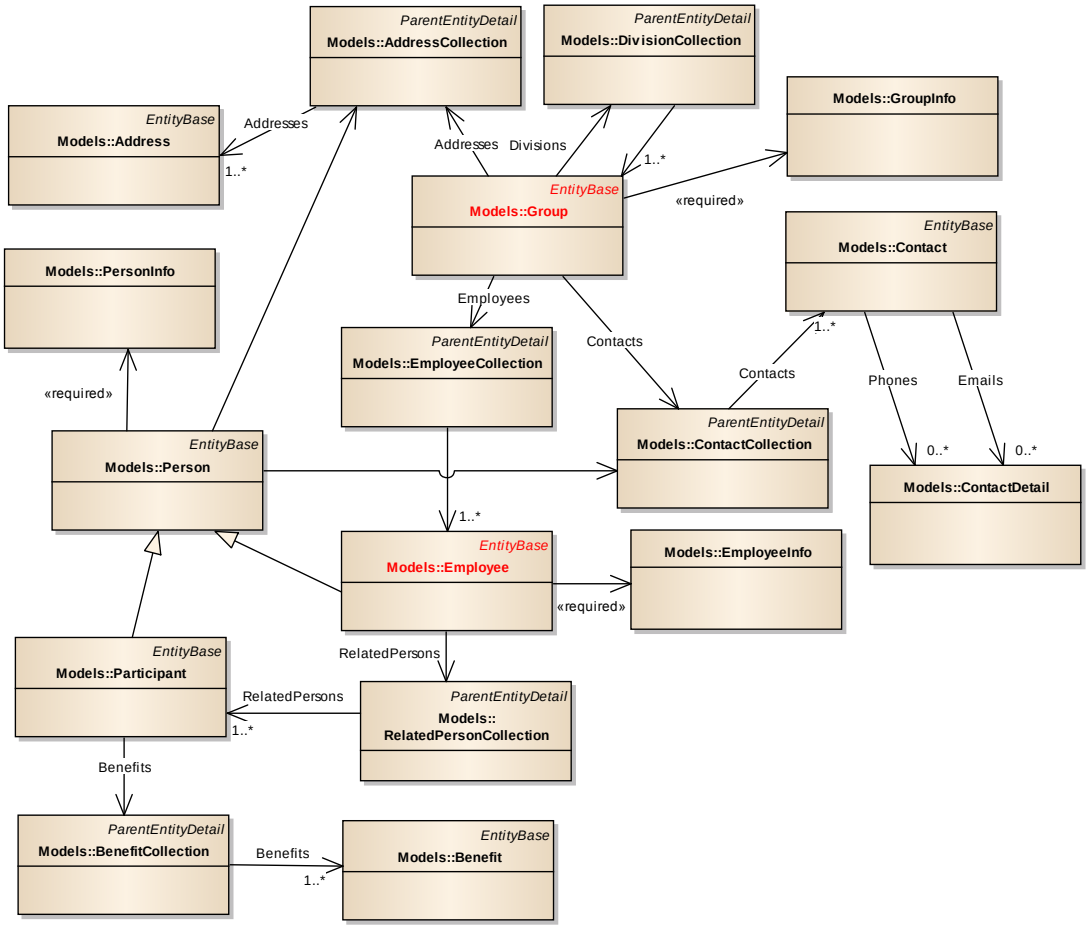


Figure 6. Canonical model's main entities: the payload of the data notifications

Behind the broker services, multiple queues were used as a durable and priority-based messaging mechanism, in order to decouple the various processes that take place at the integration framework level: receiving notifications, processing notifications and their context, and finally dispatching notifications to targeted systems.

V. CONCLUSION

Data integration and synchronization in medium to large multi-domain enterprise systems can be achieved via custom integration frameworks using various enterprise integration patterns and making appropriate technology choices.

This paper presented an actual, real-world integration solution, explained via several structural and behavioral system models, and provided details on how the "maintain data copies" data integration pattern would be realized via a broker-based messaging system. The data exchanged between the various domains is encapsulated inside a canonical model, which is the common data abstraction across the enterprise. This in turn is wrapped inside a context-based, generic, and reusable notification model, allowing systems to react to these notifications based on their own business rules.

The resulting architecture presented here features scalability, extensibility, and high-availability – to mention just a few quality attributes, while supporting near-real-time data synchronization between systems and allowing them to operate without awareness of each other, while using their individual data formats, features, and domain rules.

REFERENCES

[1] T. Erl, "Service-Oriented Architecture: A field Guid to Integrating XML and Web Services," Prentice Hall, 2004.

[2] T. Erl, "Service-Oriented Architecture (SOA): Concepts, Technology, and Design," s.l.:Prentice Hall, 2005.

[3] T. Erl., "SOA Design Patterns," Prentice Hill, 2009.

[4] T. Erl, et al., "Next Generation SOA: A Concise Introduction to Service Technology & Service-Orientation," Prentice Hall, 2014.

[5] M. Fowler, "Patterns of Enterprise Application Architecture," Addison-Wesley Professional, 2002.

[6] M. Fowler, Martin Fowler. [Online]. Available from: http://www.martinfowler.com/bliki/AnemicDomainModel.html [retrieved: June, 2015]

[7] G. Hohpe, and B. Woolf, "Enterprise Integration Patterns; Designing, Building, and Deploying Messaging Solutions," Addison-Wesley, 2012.

 ISBN: 978-1-61208-448-0

[8] Microsoft, Data Integration. [Online]. Available from: https://msdn.microsoft.com/en-us/library/ff647273.aspx [retrieved: June, 2015]

[9] Microsoft, Functional Integration. [Online]. Available from: https://msdn.microsoft.com/en-us/library/ff649730.aspx [retrieved: June, 2015]

[10] Microsoft, Integration Patterns. [Online]. Available from: https://msdn.microsoft.com/en-us/library/ff647309.aspx [retrieved: June, 2015]

[11] Microsoft, Service Bus for Windows Server Quotas. [Online]. Available from: https://msdn.microsoft.com/en-us/library/dn441429.aspx [retrieved: June, 2015]

[12] M. Iridon, Cândea LLC. "Technical Design Specifications for Enterprise Integration Solution," , 2015, unpublished/internal document.

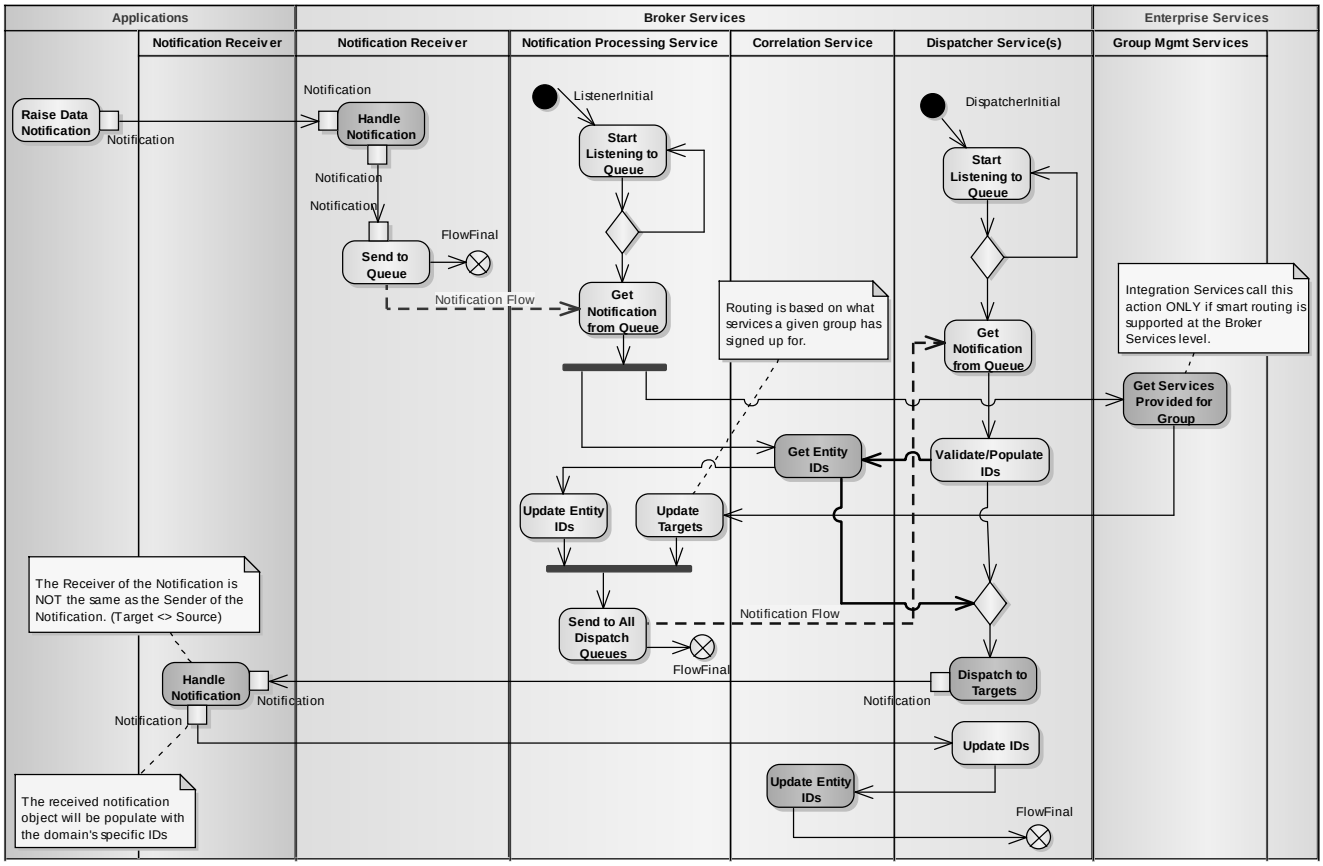


Figure 7. Enterprise integration activity model

Development of the MedITNet Assessment Method

Enabling Healthcare Delivery Organisation Self Assessment against IEC 80001-1

Silvana Togneri MacMahon, Fergal McCaffery, Frank Keenan
Department of Computing & Mathematics
Dundalk Institute of Technology
Co. Louth, Ireland
{e-mail: silvana.macmahon, fergal.mccaffery, frank.keenan}@dkit.ie

Abstract— **The provision of care to patients has moved away from episodic acute care due to the increase in chronic diseases such as diabetes. This has changed the relationship between the patient and the care team. The management of chronic disease requires the use of information technology including networked medical devices to facilitate the establishment of an ongoing relationship between the patient and care team. The use of networked medical devices can provide benefits to patients such as reduced cost of care, reductions in adverse events and improved care through the provision of accurate and up-to-date information. However, the placement of a medical device onto an IT network can lead to risks to the device. These risks may lead to incorrect or degraded performance of the device impacting patient care and negating the potential benefits of using the device. While, IEC 80001-1 was developed to assist Healthcare Delivery Organisations (HDOs) in addressing these risks, HDOs may struggle in implementing the requirements of the standard. This paper discusses the development of an Assessment Method which forms part of MedITNet, an assessment framework which can be used by HDOs to assist them in implementing the requirements of the standard by providing a flexible, consistent and repeatable approach to assessing the capability of their risk management processes relating to networked medical devices. The assessment highlights weaknesses in the process and can be used as a foundation to improve these processes.**

Keywords- Risk Management; Medical IT Networks; IEC 80001-1; MedITNet; Assessment Framework; Assessment Method.

I. INTRODUCTION

The recent downturn in the global economy has led to an increased focus on ensuring that a high standard of care is provided to the patient while reducing the cost of care. Interoperability of medical devices has been recognised for its potential to achieve this goal [1]-[3]. Such is the potential that governments have provided incentives to promote the meaningful use of interoperable medical devices and Health Information Technology (HIT), such as Electronic Health Records (EHRs) [4]-[6]. The use of interoperable medical devices has resulted from the increased prevalence of chronic conditions such as diabetes which has resulted in a move away from acute episodic care. The management of chronic disease requires the establishment of an ongoing relationship between the patient and their care team facilitated by carefully designed care processes and requiring the support of information technology [7]-[10] As a result of this change, the number of networked medical devices in use continues to increase [11]-[13].

A number of benefits of the use of networked medical are recognised. These include reducing the instances of adverse events improving patient safety, reducing the time spent by clinicians manually entering information, reducing redundant testing due to inaccessible information, improving patient care, reducing healthcare costs and ensuring comprehensive and secure management of health information [14]-[15]. These benefits have resulted in medical IT networks becoming a critical, integral component of the medical system [16]. However, as medical devices increasingly interface with other equipment and hospital information systems the integration complexity of the systems is increased and this presents additional operational risks [13][17]–[19]. Proprietary networks were traditionally used when a device was placed onto a network. However, these are being used less with medical devices being designed to be placed onto the hospitals general IT network. This means that medical device manufacturers no longer exercise control over the configuration of the network [20]. This lack of control can lead to risks which result in unintended consequences outside the control of the medical device manufacturer. The placement of the device onto the hospital network creates a new system in which the device has not been validated [21]. These risks can result in the incorrect and degraded performance of the medical device [22][23] compromising patient safety, effectiveness and the security of the IT network [24]-[26].

IEC 80001-1: *Application of risk management for IT-networks incorporating medical devices* [27] was published in 2010 to address the risks associated with the incorporation of a medical device into an IT network. However, HDOs face challenges when implementing the

 ISBN: 978-1-61208-448-0

requirements of this standard [28]. HDOs vary in size and in terms of the capability of their risk management processes [16] [29] and the regulatory requirements of the region in which they provide care differ meaning that the implementation of the requirements of the standard will vary depending on the relevant regulatory requirements. The effective performance of risk management activities requires interaction between different stakeholder groups. An understanding of the context of the HDO is also required in order to manage the identified risks [17][30]. In addition, organisational changes are required to facilitate the necessary level of interaction among stakeholders and HDOs may be unprepared for this [13] due to the fact that departments within the HDO typically operate in silos [7]. These challenges make the requirements of the standard confusing and difficult to implement.

These difficulties in implementing the requirements of the standard highlighted the need to provide HDOs with assistance. This research has focused on the development of an assessment framework which provides HDOs with a flexible approach to assessing the capability of their current risk management processes relating to medical IT networks. The use of the assessment framework enables communication among stakeholders groups allowing HDOs to implement the requirements of the standard.

The rest of this paper is organized as follows. Section II describes the development of the Assessment Method component of the MedITNet assessment framework while Section III described the stages of the Assessment while the validation of the resultant Assessment Method is discussed in Section IV. The conclusions are presented in Section V.

II. DEVELOPMENT OF THE ASSESSMENT METHOD

The Assessment Method described in this paper is one of three components which make up the MedITNet assessment framework [31][32]. In addition to the Assessment Method, MedITNet contains a Process Reference Model (PRM) and Process Assessment Model (PAM). The PRM provides a description of 14 processes which address the requirements of IEC 80001-1. The processes within the PRM are described in terms of the purpose of the process and the outcomes achieved as a result of performing the process. The PAM extends the description of the processes by including a description of the base practices or activities performed during the process and the work products used or produced as a result of performing the process. The PAM also introduces the concept of a measurement framework or scale on which the capability of the process can be measured. The Assessment Method provides a consistent approach to assessing the capability of the processes in the PAM using questions related to each of the base practices. The Assessment Method can be tailored for use based on the context in which the HDO provides care.

A. Development Approach

The approach to the development of the Assessment Method combines the learnings from a literature review with knowledge of risk management practices in a HDO. In order to understand the risk management practices within the HDO, focus groups sessions were conducted with risk management stakeholders within a HDO. These sessions were performed during the Practice-Inspired Research phase of the Action Design Research (ADR) process [33] which was used in the development of the Assessment Method and also in the development of the MedITNet Assessment framework.

B. Literature Review

In order to inform the development of the Assessment Method, a review of Assessment Methods for similar standards was completed. This review focused on ISO/IEC 15504-3 [34] and Appraisal Requirements for CMMI [35] Domain specific including Rapid Assessment for Process Improvement in Software Development (RAPID) [36], Express process appraisal (EPA) [37], Adept [38], Med-Adept [39] and Tudor IT Service Management Process Assessment (TIPA) [40] were also reviewed. While this review informed the development of the Assessment Method, the results of the review were not sufficient in themselves to develop the Assessment Method. In order to develop the Assessment Method, the results of the literature review were combined with the knowledge gained during the Practice-Inspired Research conducted as part of this study. This approach allowed the researcher to take into account the concerns which HDOs express in relation to the implementation of the IEC 80001-1 standard.

The literature review provided an understanding of the challenges that HDOs encounter when incorporating a medical device into an IT network. Each of the identified challenges was considered when developing the requirements for the Assessment Method, using a similar approach to that used by Mc Caffery and Coleman [41] using criteria for Assessment Methods as outlined by Anacleto et al. [42]. The criteria were adapted to take into account the domain in which the Assessment Method will be used, that is, within the HDO rather than in the context of software development. The development of the requirements for the Assessment Method also took into account the challenges related to the management of risk associated with the incorporation of a medical device into an IT network which were highlighted as part of the Literature Review and Practice-Inspired Research. The requirements for the Assessment Method were defined as follows:

- Due to the constraints on resources within HDOs, the Assessment Method should be lightweight in its approach and facilitate self-assessment;

 ISBN: 978-1-61208-448-0

- The Assessment Method should be based on the processes described in the MedITNet PAM;
- Guidance should be provided for tailoring the Assessment Method for use in various scales of HDOs and in different geographical contexts. The Assessment Method should also facilitate assessments based on conformance with the standard as well as those which seek to assess the capability level with which risk management processes are being performed;
- The Assessment Method should support the identification of risks and improvement opportunities;
- The Assessment Method should not assume any previous knowledge of process assessment on the part of those conducting the assessment;
- The Assessment Method should facilitate the development of tool support in the future;
- The Assessment Method should be publicly available;
- The Assessment Method should encourage a culture of communication among various multidisciplinary risk management stakeholders including those within and external to the HDO;
- The Assessment Method should be validated for use within the HDO context.

In addition to the literature review and, to augment the Practice-Inspired Research, members of the Clinical Engineering team (CE) and the Clinical Informatics team in a HDO were consulted throughout the development of the questions for the Assessment Method. This was an iterative process which is in the following section.

C. Question Development

The involvement of HDO risk management stakeholders in the development of the Assessment Method was considered to be vital as HDOs may use the Assessment Method in its form within the technical report and without reference to the PRM and PAM. The Assessment Method assesses against ISO/IEC 15504-2 compliant models i.e. the MedITNet PRM and PAM. These models describe processes at the level of the *process purpose, outcomes, practices and work products.* This approach to the development of the Assessment Method ensures its applicability beyond the HDO assisting with its development, across varying geographical and regulatory contexts. The development of the assessment questions, which form part of the Assessment Method, was completed in two phases.

a) Question Development – Phase 1

During phase 1 of the question development process, a meeting was held in the HDO with the Principal Physicist and a Physicist/Clinical Engineer. Both had taken part in the initial phase of the Practice-Inspired Research and were already familiar with the provisions of the standard and the proposed MedITNet framework.

During the previous discussions on the current risk management practices within the HDO, it was agreed that the *Risk Analysis and Evaluation Process* was the main process relating to the identification and classification of risks. It was noted during the previous focus groups session that discussion of the Risk Analysis and Evaluation process lead to discussion of other aspects of risk management which are outside the scope of that process. Therefore, it was decided that questions should be developed for this process first.

The development of these questions would inform the development of the assessment questions for the remaining processes. In order to develop the questions for the Risk Analysis and Evaluation process, each of the base practices was reviewed and the participants were asked to formulate a question that could be used to assess the base practice being described. To facilitate gaining an understanding of each of the base practices, each base practice was discussed in the context of the standard with the relevant section of the standard being consulted and reviewed if required. Once all participants were clear on the meaning of the base practice, the participants from the clinical engineering team were encouraged to think of a "real" scenario where the relevant base practice had been implemented in the past. The discussion of the scenario would focus on how the base practice was implemented in the context and any constraints that may have affected the implementation of the base practice.

Once the practice had been discussed in context, the participants were encouraged to formulate questions that could be used to assess the degree to which the base practice had been implemented during the proposed scenario. All questions which were formulated by the participants were recorded and the participants were encouraged to rephrase the questions in order to decrease the number of questions used to assess each base practice. The Risk Analysis and Evaluation Process contains five base practices against which 14 questions were eventually formulated. This draft of questions was used in the validation focus group within HDO A which was conducted as part of the ADR process. However, the set of questions (presented in Table I) does not represent the final set of questions which were developed to be used in the assessment of this process.

b) Question Development – Phase 2

During the second phase of the development of the questions, the questions for the remaining 13 processes were developed. These questions were developed with the assistance of the Clinical Informatics Manager (CIM) of the HDO. The CIM is a former nurse who oversees the systems administration tasks of the Clinical Information

 ISBN: 978-1-61208-448-0

System within the Intensive Care Unit. The CIM was briefed on the research being carried out on the development of the Assessment Method and was given the PRM and PAM to review and was briefed on the requirements of the IEC 80001-1 standard. Following the development of the assessment questions for the remaining 13 processes, the CIM was also shown the questions developed during phase 1 for the Risk Analysis and Evaluation Process. The CIM was asked to review and reformulate the questions, as required, for this process based on their experience of development of the questions for the remaining processes.

In general, one question was related to each of the base practices. However, the assessment of some base practices required more than one question. The CIM was asked to participate in the development of the questions in order to ensure that the questions were phrased in a way that could be understood by various risk management stakeholders within the HDO. The questions were also developed based closely on the base practices defined within the PAM to ensure that the questions could be applied across multiple HDO contexts and were not specific to the HDO in which the research was being carried out.

TABLE I – SAMPLE ASSESSMENT QUESTIONS

Base Practice Summary:	***Question Number:***	***Question:***
BP.1 - Identify likely hazards.	BP.1 Q.1	How do you identify likely safety hazards for individual devices?
	BP.1 Q.2	How do you analyse the system as a whole to identify likely safety hazards?
	BP.1 Q.3	How do you consider the impact of the device on the environment?
	BP.1 Q.4	How do you consider the impact of the device in terms of effectiveness?
	BP.1 Q.5	How do you consider the impact of the device in terms of data and system security?
BP.2 - Estimate associated risks.	BP.2 Q.1	Do you have a procedure for estimating risk?
	BP.2 Q.2	What approach do you use to estimate the risk associated with each source of harm?
	BP.2 Q.3	What information sources do you use to estimate the risks associated with each source of harm?
	BP.2 Q.4	Are risks reviewed throughout the life cycle?
BP.3 - List possible consequences of harm.	BP.3 Q.1	How do you identify possible consequences of harm?
BP.4 - Record results of Risk Analysis and Evaluation activities.	BP.4 Q.1	How are risk management activities recorded?
	BP.4 Q.2	Are instances where risk estimate is so low that risk reduction is not required recorded?
BP.5 - Implement Risk Control Measures.	BP.5 Q.1	How are risk control measures implemented?
	BP.5 Q.2	Are risk control measures implemented in line with risk management policy?

III. STAGES OF THE ASSESSMENT METHOD

The stages of the assessment process are illustrated in Figure 1 and discussed in the remainder of this section.

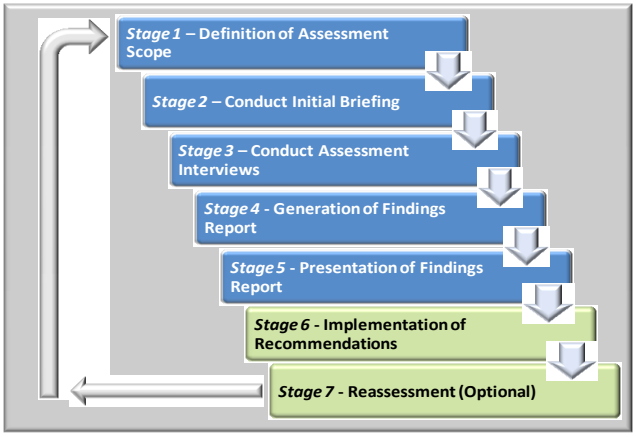


Figure 1. Stages of the Assessment Process

Participants in the assessment process include the lead assessor, a risk management stakeholder from within the HDO, who will manage the assessment on behalf of the Top Management (TM) of the HDO. Focus group interviews are used during the assessment to ensure communication among risk management stakeholders. An additional Assessor (A) may be required to assist the LA. In addition to sponsoring the assessment, TM will ensure that Risk Management Stakeholders (RMS) are available to participate in the assessment. The RMS will be drawn from a multi-disciplinary team from within the HDO and will include members of the IT, CE and Clinical Teams and any other relevant RMS as required. The RMS may also include participants who are external to the HDO such as MDMs. It should be noted that Stages 1 to 5 above complete the assessment activities. Stage 6 involves the implementation of recommendations made during the assessment. Where a follow-up assessment is required, stage 7 is performed. A reassessment can be used to confirm that the recommendations for improvements to the risk management process have improved risk management processes as envisaged.

a) Stage 1

The lead assessor meets with Top Management and the scope of the assessment is discussed. The system which is to be the focus of the assessment is defined and the context of the system is understood. At this time, the availability of relevant risk management stakeholders to participate in the assessment is confirmed.

b) Stage 2

The lead assessor meets with relevant risk management stakeholders who will be taking part in the assessment to explain the Assessment Method and give details of what their participation will involve.

 ISBN: 978-1-61208-448-0

c) Stage 3

The lead assessor conducts interviews based on the scripted questions with the relevant risk management participants and evaluates the responses. The assessor makes notes on the interviews and additional questions are asked if clarification is required. Relevant work products are reviewed at this stage.

d) Stage 4

A findings report is prepared based on the data gathered at stage 3. Each process is reviewed in turn and where relevant particular strengths and weaknesses are identified based on the evaluation and interview notes. Suggested actions to address these issues and to facilitate process improvement are outlined and discussed.

e) Stage 5

The findings report is presented.

f) Stage 6

Having allowed time for the contents of the report to be considered, the findings are discussed and a plan for improvement of the processes with specific improvement objectives is agreed.

g) Stage 7

The HDO having implemented the agreed improvements have the option of performing a reassessment to ensure that improvements have been implemented and that risk management processes have improved accordingly.

IV. VALIDATION OF THE ASSESSMENT METHOD

The Assessment Method was validated from the perspective of its utility in a specific HDO context. The first stage of validation consisted of performing an assessment of current risk management practices within a HDO context using the Assessment Method. This phase consisted of a pilot implementation of the Assessment Method by performing an assessment of the Risk Analysis and Evaluation process using the questions from the Assessment Method. A focus group session took place in the HDO with participants from various risk management stakeholder groups taking part. The assessment allowed for areas of weakness in the current risk management processes related to medical IT networks to be highlighted and addressed. A findings report was provided to the HDO and a follow-up focus groups session took place nine months later to review which recommendations had been implemented. A summary of the recommendations is provided in Table 2. This phase of the validation ensured that the developed questions could be understood by risk management stakeholders and were suited for use for the performance of an assessment in the specific HDO context. The performance of the assessment resulted in improvement to not only the risk analysis and evaluation process within the HDO, but participants also reported improvements in the overall risk management of medical IT networks within the HDO. The performance of this stage of the validation confirmed the utility of the Assessment Method in a specific HDO context.

TABLE II - SAMPLE ASSESSMENT RESULTS SUMMARY

BP.1 - Identify likely hazards
Develop a standardised process for the identification of hazards, including the identification of hazards during the tendering process
Maintain the same level of documentation in the recording of identified hazards, regardless of when in the lifecycle the hazard is identified
Store information related to risk management in a manner which can be accessed as an information source for the estimation of future risks
BP.2 - Estimate associated risks
Establish a policy detailing risk acceptability criteria
Formalize and document a procedure for the estimation of risk which stipulates which risk management stakeholders should be involved
BP.3 - List possible consequences of harm
Consider consequences of harm based on the risk acceptability criteria
Consider consequences of harm based on the risk management policy
BP.4 - Record the results of Risk Analysis and Evaluation activities
Record Risk Analysis and evaluation activities in the risk management file
Ensure accessibility of emails containing information on Risk Analysis and Evaluation activities
BP.5 - Implement Risk Control Measures
Establish a process for risk control
Ensure that risk control measures are implemented in line with the risk control process
Document risk which have been considered so low as not to require additional risk control measures

In order to confirm the generalisability of the Assessment Method across a range of HDO contexts, the Assessment Method was also validated through expert review by members of the standards community from the International Electrotechnical Commission (IEC) Sub-Committee 62A and the International Organization for Standardization (ISO) Technical Committee 215 Joint Working Group 7 (JWG7). Members of this group are drawn from risk management stakeholders within HDOs, medical device manufacturers and providers of other IT technology. They are recognised as experts in their field and represent their country in this capacity. The focus of this stage of the validation is to ensure that the Assessment Method can be used across multiple HDO contexts, regardless of the regulatory environment in which the HDO operates. During this phase of the validation the Assessment Method was circulated to members of JWG7 for review. The Assessment Method was circulated with the MedITNet PRM and PAM and members were invited to make comments on any aspect of these components of MedITNet. The review by members of this group resulted in a number of changes to the Assessment Method including the provision of sample templates which could be used by HDOs during the performance of an assessment and in the preparation of the findings report for circulation to Top Management of the HDO. In addition to the review by members of JWG7, a focus group session was conducted with a selection of experts from the group. These experts were asked to

 ISBN: 978-1-61208-448-0

comment on various aspects of the overall MedITNet framework. During this session experts reported that the use of the Assessment Method and specifically the assessment questions resulted in risk management stakeholders having a greater understanding of the requirements of the IEC 80001-1 standard. The experts also noted that the definition of the requirements of the standard at the level of processes in the PAM enabled the assessment questions to be tailored to take into account of the context in which the HDOs provide care. This was

Each of these phases was performed iteratively as part of the ADR process and changes suggested by each phase of the validation were incorporated into the next version of the Assessment Method and the overall MedITNet framework.

V. CONCLUSIONS

While IEC 80001-1 takes steps to address the risks associated with the placement of a medical device onto an IT network, HDOs may face challenges in understanding and implementing the requirements of the standard. The MedITNet framework has been developed in order to assist HDOs in addressing these challenges. The Assessment Method provides a consistent, repeatable and tailorable approach to the assessment of the capability of risk management processes related to the management of medical IT networks. An assessment of these processes can highlight weaknesses therein and can be used as a foundation for an improvement of risk management processes. Effective risk management of medical IT networks ensures that the potential benefits of networked medical devices are realised while ensuring the safety of the patient is protected, the effectiveness of the device is assured and the security of the data and system are preserved.

ACKNOWLEDGMENT

This research is supported by the Science Foundation Ireland Principal Investigator Programme, grant number 08/IN.1/I2030 (the funding of this project was awarded by Science Foundation Ireland under a co-funding initiative by the Irish Government and European Regional Development Fund),and by Lero - the Irish Software Research Centre (http://www.lero.ie) grant 10/CE/I1855 & 13/RC/20194.

REFERENCES

[1] West Health Institute, "The Value of Medical Device Interoperability - Improving patient care with more than $30 billion in annual health care savings," 2013.

[2] A. Hamilton, *et al.*, "Summary of the August 2011 Symposium on the Role and Future of Health Information Technology in an Era of Health Care Transformation," The George Washington University, 2011.

[3] I. Lee, *et al.*, "High-confidence medical device software and systems," *Computer,* vol. 39, 2006, pp. 33-38,.

[4] N. Milenkovich, March 15, 2013, [Accessed 14/07/2015]. *OCR issues new HITECH regulations* Available: http://drugtopics.modernmedicine.com/drug-topics/news/drug-topics/health-system-news/ocr-issues-new-hitech-regulations

[5] Centers for Medicare & Medicaid Services, "42 CFR Parts 412, 413, 422. Medicare and Medicaid Programs; Electronic Health Record Incentive Program; Final Rule ", Health and Human Services Ed., ed, 2010.

[6] Centers for Medicare & Medicaid Services. 10/04/2013, [Accessed 14/07/2015]. *EHR Incentive Programs*. Available: http://www.cms.gov/Regulations-and-Guidance/Legislation/EHRIncentivePrograms/index.html?redirect=/ehrincentiveprograms

[7] Institute of Medicine. (2001). *Crossing the Quality Chasm: A New Health System for the 21st Century*. Available: https://download.nap.edu/catalog.php?record_id=10027

[8] E. H. Wagner, "The role of patient care teams in chronic disease management," *BMJ: British medical journal,* vol. 320, 2000, p. 569.

[9] E. H. Wagner, B. T. Austin, C. Davis, M. Hindmarsh, J. Schaefer, and A. Bonomi, "Improving chronic illness care: translating evidence into action," *Health affairs,* vol. 20, 2001, pp. 64-78.

[10] C. Hoffman and D. Rice, "Chronic care in America: A 21st century challenge," *Princeton, NJ: The Robert Wood Johnson Foundation,* 1996.

[11] J. Comstock. 2013,. [Accessed 14/07/2015] *14M networked medical devices to ship by 2018*. Available: http://mobihealthnews.com/28295/14m-networked-medical-devices-to-ship-by-2018/

[12] Agency for Healthcare Research and Quality (AHRQ), "Health IT for Improved Chronic Disease Management," Department of Health and Human Services, Ed., ed, 2013.

[13] M. Castañeda, "Connecting devices and data on the healthcare network," *Biomedical Instrumentation & Technology,* vol. 44, 2010, pp. 18-25.

[14] J. Goldman and S. Whitehead, "Advancing the Adoption of Medical Device "Plug-and-Play" Interoperability to Improve Patient Safety and Healthcare Efficiency," 2010.

[15] K. K. Venkatasubramanian, S. K. S. Gupta, R. P. Jetley, and P. L. Jones, "Interoperable Medical Devices - Communication Security Issues," *IEEE Pulse,* vol. Sept/Oct 2010, 2010.

[16] R. Hampton and R. Schrenker, "What Does IEC 80001-1 Mean to You?," *24x7 - Technology and Service Solutions for Biomeds,* 2011.

[17] S. R. Rakitin, "Networked Medical Devices: Essential Collaboration for Improved Safety," *AAMI.org,* 2009.

[18] S. Loughlin and J. S. Williams, "The top 10 medical device challenges," *Biomedical Instrumentation & Technology,* vol. 45 2011, pp. 98-104.

[19] T. Mehta and C. Mah, "Auto-Provisioning of Biomedical Devices on a Converged IP Network," *Biomedical Instrumentation & Technology,* vol. 43, 2009, pp. 463-467.

[20] T. Gee. 2008. [Accessed 14/07/2015] *Medical Device Networks Trouble Industry*. Available: http://medicalconnectivity.com/2008/12/18/medical-device-networks-trouble-industry/

[21] S. Eagles, "An Introduction to IEC 80001: Aiming for Patient Safety in the Networked Healthcare Environment," *IT Horizons,* vol. 2008, 2008.

[22] National Cybersecurity and Communications Integration Center, "Attack Surface: Healthcare and Public Health Sector," ed, 2012.

[23] D. Talbot. 2012, [Accessed 14/07/2015] Computer Viruses Are "Rampant" on Medical Devices in Hospitals. *MIT Technology Review*. Available: http://www.technologyreview.com/news/429616/computer-viruses-are-rampant-on-medical-devices-in-hospitals/

[24] J. Graham and C. Dizikes, "Baby's death spotlights safety risks linked to computerized systems," in *Chicago Tribune*, ed, 2011.

[25] J. Shuren, "Health Information Technology (HIT) Policy Committee Adoption/Certification Workgroup - Testimony of Jeffrey Shuren, Director of FDA's Centre for Devices and Radiological Health," ONC, Ed., ed, 2010.

 ISBN: 978-1-61208-448-0

[26] S. Eagles, "IEC 80001: An Introduction," 80001-1 Experts,, Presentation from 19th Annual NCBA ConferenceSeptember 13, 2012.

[27] IEC, "IEC 80001-1 - Application of Risk Management for IT-Networks incorporating Medical Devices - Part 1: Roles, responsibilities and activities," ed. Geneva, Switzerland: International Electrotechnical Commission, 2010.

[28] F. J. Hegarty, S. T. MacMahon, P. Byrne, and F. McCaffery, "Assessing a Hospital's Medical IT Network Risk Management Practice with 80001-1," *Biomedical Instrumentation & Technology,* vol. 48, 2014, pp. 64-71.

[29] T. Cooper and K. Fuchs, "The Wireless Challenge - Technology Risk Assessment In Healthcare Facilities," *Biomedical Instruments and Technology,* vol. May/June 2013, 2013.

[30] M. Janssen and R. Schrenker, "Guidelines From 80001: Maintaining a Medical IT Network," *Biomedical Instrumentation & Technology,* vol. 45, 2011, pp. 295-299, 2011/07/01.

[31] S. T. MacMahon, F. McCaffery, S. Eagles, F. Keenan, M. Lepmets, and A. Renault, "Development of a Process Assessment Model for assessing Medical IT Networks against IEC 80001-1," presented at the Software Process Improvement and Capability Determination (SPICE) 2012, Mallorca, Spain, 2012.

[32] S. T. MacMahon, F. McCaffery, and F. Keenan, "Transforming Requirements of IEC 80001-1 into an ISO/IEC 15504-2 Compliant Process Reference Model and Process Assessment Model," presented at the European Systems and Software Process Improvement and Innovation Conference, Dundalk, Co Louth, Ireland, 2013.

[33] M. Sein, O. Henfridsson, S. Purao, M. Rossi, and R. Lindgren, "Action design research," 2011.

[34] ISO/IEC, "ISO/IEC 15504-3:2004 Information technology -- Process assessment -- Part 3: Guidance on performing an assessment," ed. Geneva, Switzerland, 2004.

[35] M. Busby, *et al.*, "Appraisal Requirements for CMMI (Registered Trademark) Version 1.3 (ARC, V1. 3)," DTIC Document2011.

[36] T. P. Rout, A. Tuffley, B. Cahill, and B. Hodgen, "The rapid assessment of software process capability," in *First International Conference on Software Process Improvement and Capability Determination*, 2000, pp. 47-56.

[37] F. Wilkie, D. McFall, and F. Mc Caffery, "The Express Process Appraisal Method," 2005.

[38] F. Mc Caffery, P. S. Taylor, and G. Coleman, "Adept: A unified Assessment Method for small software companies," *Software, IEEE,* vol. 24, 2007, pp. 24-31.

[39] F. McCaffery and V. Casey, "Med-Adept: A Lightweight Assessment Method for the Irish Medical Device Software Industry," presented at the EuroSPI, Grenoble France, 2010.

[40] B. Barafort, *et al.*, *ITSM Process Assessment Supporting ITIL : Using TIPA to Assess and Improve your Processes with ISO 15504 and Prepare for ISO 20000 Certification* vol. 217. Zaltbommel, Netherlands: Van Haren, 2009.

[41] F. Mc Caffery and G. Coleman, "The development of a low-overhead Assessment Method for Irish software SMEs," *Journal of Information Technology Management (JITM),* 2007.

[42] A. Anacleto, C. G. von Wangenheim, C. F. Salviano, and R. Savi, "Experiences gained from applying ISO/IEC 15504 to small software companies in Brazil," in *4th International SPICE Conference on Process Assessment and Improvement, Lisbon, Portugal*, 2004, pp. 33-37.

 ISBN: 978-1-61208-448-0

XML Schema for Implementing Safety Management System in Shipbuilding

Youhee Choi, Byungtae Jang
Smart Mobility Research Department
ETRI
Daejeon, Korea
e-mail: {yhchoi, jbt}@etri.re.kr

***Abstract*—With a rising demand for developing deep sea resources recently, that for offshore plant construction are getting greater. Accordingly, plant owners request more for a safety management system in the process of offshore plant building. Thus, it is required that a safety management system is built for each shipyard or offshore plant building project. In order to develop a safety management system, it is important that risk factors in each task should be properly identified. Most information with respect to work processes or risk factors can be commonly applicable. But, most of the safety management system is developed upon an assumption that such information is implicitly inherent within the system. In this respect, to ensure that the key information, such as task and hazard information for building safety management system is not inherent within the system, we defined XML (eXtensible Markup Language) Schema to ensure that such information can be expressed in standardized-format XML. By doing so, even if risk and work process contents change, XML files can be used after redefined - without changing safety management logic of a relevant system.**

Keywords-safety;shipbuilding;XML Schema; hazard; risk.

I. INTRODUCTION

The shipbuilding industry refers to ship building and repair sectors. The industry is regarded as one of the most dangerous sectors. Accordingly, safety management is essential task of project management in shipbuilding industry. This promoted consistent efforts for technology advancement concerning shipbuilding processes, facilities and equipment. Still, today most of tasks at shipyards are labor-intensive and requires the specialized, highly skilled. In addition, shipbuilding tasks are mostly performed outdoors, largely influenced by air temperature, climate and other atmospheric environmental factors. Relevant working conditions change constantly. In order to improve safety management, such real workplace environment changes should be reflected in a safety management system promptly. Nonetheless, the current safety management framework is featured by policy aspects covering safety management manuals, safety training for workers and supervision by safety management supervisors. For this reason, proper safety management is not undertaken effectively reflecting actual situations for each worker [1]. In an effort to address this problem, studies are carried out from various technological perspectives. First of all, there is a study striving to improve the current safety management framework by monitoring workers' safety utilizing building information modeling (BIM) [2], virtual reality [3] and augmented reality [4]. Also, there is an endeavor to enhance workplace monitoring technology in terms of worker location monitoring technologies [5]-[8]. Another approach is to improve safety management system in terms of risk analysis-based risk assessment model and related supporting system to prevent accidents [9][10]. While, in terms of safety management logic, it is important that risk factors in each task should be properly identified. This information can be accumulated empirically. Only some processes differ related to what is produced in the same domain. Accordingly, most information with respect to work processes or risk factors can be commonly applicable. But, most of the safety management system is developed upon an assumption that such information is implicitly inherent within the system. In this aspect, we defined XML Schema to ensure that this information is not inherent within the system but explicitly expressed in standardized format XML. By defining work process or risk factor information in XML files based on XML Schema, our approach allows to promptly reflect real-time workplace situations and be made use of by various shipyards or other industrial sites not modifying the logic of safety management systems. The rest of this paper is organized as follows. Several related researches are presented in Section II. Section III presents the XML schema for implementing the safety management system. Section IV presents XML files as examples based on the XML schema. The paper concludes with future work and conclusions in Section V.

II. RELATED RESEARCHES

In the existing approach for developing safety management frameworks, risks are identified through meetings between safety managers and task managers using plans, accident cases and empirical information, etc. These risks do not sufficiently reflect changing situations in real worksites. For these reasons, some visualization techniques, such as BIM, game technologies, virtual reality, and augmented reality have been utilized to improve the current safety management practices. In this respect, there is a study performing 6-day cycle safety plan through 3D modeling of

 ISBN: 978-1-61208-448-0

the working environment [11]. In addition, there is also an approach offering a framework providing pre-designed virtual project site model in safety management system, for risk identification [12]. Also, there is another approach proposing rule-based safety checking system for falls based on 4D BIM [13]. These researches are a part of visualization technologies for workplace. On the other hand, there are studies about monitoring technologies of real-time locations of workers in workplace. In order to prevent accidents, there is a study using RFID (Radio Frequency IDentification) technologies to track workers' location in real-time [14]. Also, there is an approach that combines wireless communication systems with sensors not using tagging technologies, such as RFID in order to detect moving objects [15]. Its focus is to identify the accurate location of moving workers and objects to prevent accidents. Also, there are studies about risk identification and assessment. Most of risk identification and assessment relies on experiences and expertise of safety management experts or work managers. A study was carried out concerning tools and evaluation models to assist such risk assessment processes [16].

III. XML SCHEMA FOR IMPLEMENTING SAFETY MANAGEMENT SYSTEM

A. *Definition of data relationship for building safety management system*

First of all, we should consider relationship among data that is necessary for building safety management system as shown in Figure 1.

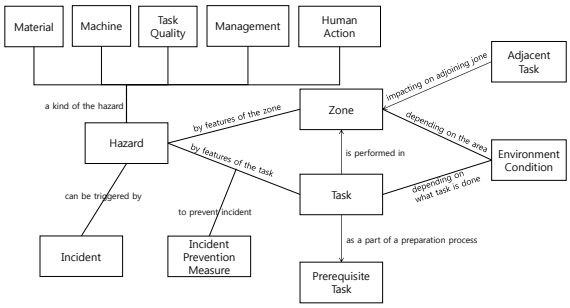


Figure 1. Data relationship for safety management system

As shown in Figure 1, most of accidents at shipbuilding sites are closely related with a task undertaken. Accordingly, a relationship is formed focusing on such task information. As hazard factors exist according to the characteristics of task, task information is related to those factors. As regards prerequisite task like equipment checking carried out before each task may not be considered as individual task but simply as part of a preparation process. Still, since such task can entail hazard factors, a relationship about prerequisite task should also be factored into. Then, each task is performed in a specific zone. This means the features of the zone can influence hazard factors and a relationship is also established with zone information. Furthermore, as climate or other outside environmental conditions can also serve as major hazard factors depending on in which area what task is done, such a relationship should also be taken into account. Especially, in shipbuilding sites, as large-sized objects are handled in a limited space, different processes of work may be performed adjacently in parallel. In this regard, hazard factors of each task can have an impact on adjoining zones, which requires relevant consideration. Hazard factors identified from each perspective can vary ranging from: those related to work materials and equipment; those in an aspect of workplace management; to those from wrong human behaviors. Ultimately, as the goal of safety management system is to prevent accidents, it is required to identify incident factors triggered by each hazard factor, and formulate and define incident prevention measures from the relationship between hazard factors and task.

B. *XML Schema for defining the relationship between Task and Hazard*

It is necessary to define the kinds of specific information required for each information listed in the previous section. It is important to define data structure and relationship in a standardized format utilizing XML formats - a standardized markup language expressing rule sets for data encoding. In this term, the definition rules for specific information - necessary for developing safety management system logic – are defined in XML Schema. Figure 2 illustrates definition of XML Schema for representing task information.

Task ID	Task Code Number	Task Name	Prerequisite Task

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="tasklist" type="tasklistType"/>
   <xs:complexType name="tasklistType">
      <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="taskitem"/>
      </xs:sequence>
   </xs:complexType>
   <xs:element name="taskitem" type="taskitemType"/>
   <xs:complexType name="taskitemType">
   <xs:sequence>
      <xs:group ref="item" />
      <xs:element maxOccurs="unbounded" minOccurs="0" ref="prerequisitetask" />
   </xs:sequence>
   </xs:complexType>
   <xs:group name="item">
      <xs:sequence>
         <xs:element ref="taskcode" />
         <xs:element ref="taskname" />
      </xs:sequence>
   </xs:group>
   <xs:element name="taskcode" type="taskcodeType" />
   <xs:simpleType name="taskcodeType">
      <xs:restriction base="xs:string">
         <xs:pattern value="[A-Za-z0-9]{3}" />
      </xs:restriction>
   </xs:simpleType>
   <xs:element name="taskname" type="xs:string" />
   <xs:element name="prerequisitetask" type="prerequisitetaskType" />
   <xs:complexType name="prerequisitetaskType">
      <xs:sequence>
      <xs:group ref="item" />
      </xs:sequence>
   </xs:complexType>
</xs:schema>
```

Figure 2. The XML Schema for task information

As shown in Figure 2, in order to express task information, we classified task information into task ID, task code number, task name, and prerequisite task. Task code number is assigned to each task for easy identification in safety management logic. The code number list should be managed in a separate file and adjusted according to individual projects and shipyard situations. Task name represents a textual name for each task. Prerequisite task is a preparatory work performed before main task.

 ISBN: 978-1-61208-448-0

Based on the classification, in order to represent a list of task information, we declared <tasklist> element as the topmost element in the XML Schema. The element <tasklist> contains the <taskitem> element to represent each task as shown in Figure 2. The element <taskitem> comprises the <item> element and the <prerequisitetask> element. The element <item> is used to show the task name and task code number. The <taskcode> element reflects the task code number of each task and shall be supplied in the format of 3 alphanumeric characters. The <prerequisite task> element shall be used to represent the task which should be performed before main task of the <task> element. Since one or more prerequisite tasks shall be existed, we defined the value of 'maxOccurs' value as 'unbounded'. The <prerequisitetask> element shall be represented using the <item> element.

Then, although hazard information is closely related with other information, we defined XML Schema elements for hazards to contain information for the hazard itself to exclude overlap of information. Figure 3 shows the XML schema for representing hazard information.

First of all, we classified hazard information into hazard ID, hazard code number, hazard name, hazard type, causes of incident, and hazard zone.

Hazard code number is assigned to each hazard for easy identification in safety management logic. The code number list should be managed in a separate file and adjusted according to individual projects and shipyard situations. Hazard name represents a textual name for each hazard. Hazard type represents hazard factors identified from each perspective. Causes of incident represent causes of an incident by the hazard. Hazard zone is required for representing the specific zone that can be affected by the hazard.

Based on the classification, in order to represent a list of hazard information, we declared <hazardlist> element as the topmost element in the XML Schema. The element < hazardlist> contains the <hazard> element to represent each hazard as shown in Figure 3. The element <hazard> contains the <hazarditem> element. The <hazard> element contains <hazardcode>, <hazardname>, and <hazardzone>. It also comprises the 'cause' attribute and the 'seriousness' attribute. The <hazardcode> element reflects the hazard code number of each hazard and shall be supplied in the format of 3 alphanumeric characters. The <hazardname> element reflects the hazard name. In case the hazard occupies the fixed spot, the <hazardzone> element reflects the spot of hazard. The 'cause' attribute reflects causes of incident and can be one value of "Falls from height", "Slips", "Trips", "Hit something fixed/stationary", "Hit by moving/falling object", "Struck by something", and "Collapse".

Next, to represent relationships between a task and hazards, task code number, hazard code number, prevention measures and seriousness can be used. The hazard code number is the hazard which can be occurred by the task. The prevention measures reflect measures that prevent incidents that arise from hazards. The seriousness represents degrees of the seriousness of the incident. Figure 4 shows the XML Schema for representing relationships between a task and hazards information.

Hazard ID	Hazard Code Number	Hazard Name	Hazard Type	Causes of Incident	Hazard Zone

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="hazardlist" type="hazardlistType"/>
  <xs:complexType name="hazardlistType">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="hazard"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="hazard" type="hazardType"/>
  <xs:complexType name="hazardType">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" ref="hazardcode" />
      <xs:element ref="hazardname" />
      <xs:element ref="hazardtype" />
      <xs:element minOccurs="0" ref="hazardzone" />
    </xs:sequence>
    <xs:attribute ref="cause"/>
  </xs:complexType>
  <xs:element name="hazardcode" type="hazardcodeType" />
  <xs:simpleType name="hazardcodeType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Za-z0-9]{3}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="hazardname" type="xs:string" />
  <xs:element name="hazardtype" type="hazardtypeType" />
  <xs:simpleType name="hazardtypeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Material"/>
      <xs:enumeration value="Machine"/>
      <xs:enumeration value="Task Quality"/>
      <xs:enumeration value="Management"/>
      <xs:enumeration value="Human action"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="hazardzone" type="hazardzoneType" />
  <xs:complexType name="hazardzoneType">
    <xs:sequence>
      <xs:element ref="referencepoint" />
      <xs:element ref="horizontalwidth" />
      <xs:element ref="verticallength" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="referencepoint" type="referencepointType" />
  <xs:complexType name="referencepointType">
    <xs:sequence>
      <xs:element name="x" type="xs:long" />
      <xs:element name="y" type="xs:long" />
      <xs:element name="z" type="xs:long" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="horizontalwidth" type="xs:long" />
  <xs:element name="verticallength" type="xs:long" />
  <xs:attribute name="cause" type="causeType" />
  <xs:simpleType name="causeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Falls from height"/>
      <xs:enumeration value="Slips"/>
      <xs:enumeration value="Trips"/>
      <xs:enumeration value="Hit something fixed/stationary"/>
      <xs:enumeration value="Hit by moving/falling object"/>
      <xs:enumeration value="Struck by something"/>
      <xs:enumeration value="Collapse"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Figure 3. The XML Schema for hazard information

TaskHazard ID	Task Code Number	Hazard Code Number	Prevention Measure	Seriousness

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="taskhazardlist" type="taskhazardlistType"/>
 <xs:complexType name="taskhazardlistType">
   <xs:sequence>
     <xs:element maxOccurs="unbounded" ref="taskhazard"/>
   </xs:sequence>
 </xs:complexType>
 <xs:element name="taskhazard" type="taskhazardType"/>
   <xs:complexType name="taskhazardType">
     <xs:sequence>
       <xs:element minOccurs="1" maxOccurs="1" ref="taskcode"/>
       <xs:group minOccurs="1" maxOccurs="unbounded" ref="hazardprevention" />
     </xs:sequence>
     <xs:attribute ref="seriousness"/>
   </xs:complexType>
   <xs:group name="hazardprevention">
     <xs:sequence>
       <xs:element ref="hazardcode"/>
       <xs:element ref="preventionmeasure"/>
     </xs:sequence>
   </xs:group>
   <xs:element name="preventionmeasure" type="preventionmeasureType"/>
   <xs:complexType name="preventionmeasureType">
     <xs:sequence>
       <xs:element name="actionitem" type="xs:string" minOccurs="1" maxOccurs="unbounded" />
     </xs:sequence>
   </xs:complexType>
   <xs:attribute name="seriousness" type="seriousnessType" />
   <xs:simpleType name="seriousnessType">
     <xs:restriction base="xs:positiveInteger">
       <xs:enumeration value="1"/>
       <xs:enumeration value="2"/>
       <xs:enumeration value="3"/>
       <xs:enumeration value="4"/>
     </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

Figure 4. The XML Schema for relationships between a task and hazards

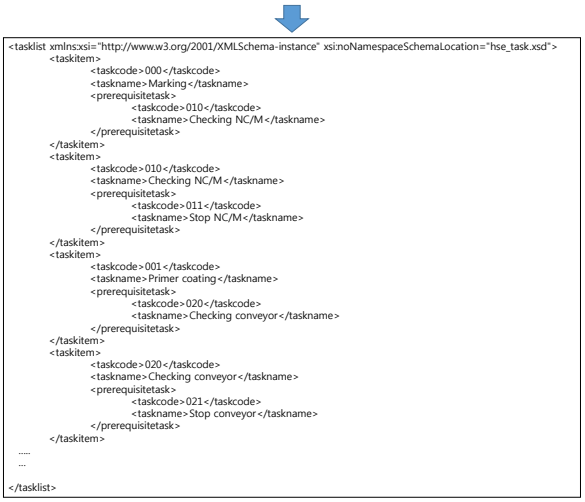
First, in order to represent a list of mapping information between a task and hazards, we declared <taskhazardlist> element as the topmost element in the XML Schema. The element < taskhazardlist> contains the <taskhazard> element to represent each mapping information as shown in Figure 4. The <taskhazard> element contains the <taskcode> element and the group element <hazardprevention>. The group element <hazardprevention> comprises the <hazardcode> element and the <preventionmeasure> element. The <preventionmeasure> element contains the <actionitem> element that reflects an action item for preventing each hazard. The 'seriousness' attribute can be a number from "1" to "4".

IV. EXAMPLE

This section describes an example that a XML file is defined based on the XML Schema for the safety management system for shipyards.

Figure 5 shows an example of XML representing some tasks of major work processes at shipyards using XML Schema defined in Section III.

Task ID	Task Code Number	Task Name	Prerequisite Task
	000	Marking	Checking NC/M
	001	Primer coating	Checking conveyor
	...	...	...
	010	Checking NC/M	Confirm that use of the NC/M was prohibited
	020	Checking conveyor	Confirm that use of the conveyor was prohibited
	...	....	



```
<tasklist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="hse_task.xsd">
    <taskitem>
        <taskcode>000</taskcode>
        <taskname>Marking</taskname>
        <prerequisitetask>
            <taskcode>010</taskcode>
            <taskname>Checking NC/M</taskname>
        </prerequisitetask>
    </taskitem>
    <taskitem>
        <taskcode>010</taskcode>
        <taskname>Checking NC/M</taskname>
        <prerequisitetask>
            <taskcode>011</taskcode>
            <taskname>Stop NC/M</taskname>
        </prerequisitetask>
    </taskitem>
    <taskitem>
        <taskcode>001</taskcode>
        <taskname>Primer coating</taskname>
        <prerequisitetask>
            <taskcode>020</taskcode>
            <taskname>Checking conveyor</taskname>
        </prerequisitetask>
    </taskitem>
    <taskitem>
        <taskcode>020</taskcode>
        <taskname>Checking conveyor</taskname>
        <prerequisitetask>
            <taskcode>021</taskcode>
            <taskname>Stop conveyor</taskname>
        </prerequisitetask>
    </taskitem>
  .....
  ...

</tasklist>
```

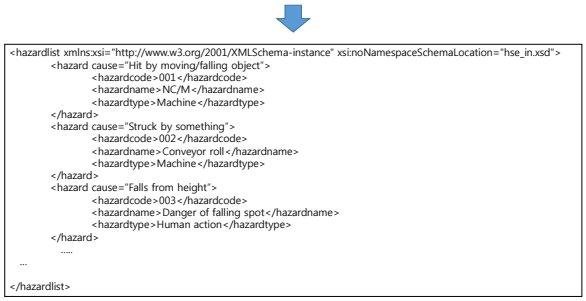
Figure 5. The XML file for task information

It represents XML defined using the <taskitem> element for each task listed in the table in Figure 5.

Figure 6 shows an example of XML representing hazard factors that can be occurred in work processes of shipyards based on the XML Schema defined in Section III.

It represents XML defined using the <hazard> element for each hazard listed in the table in Figure 6.

Hazard ID	Hazard Code Number	Hazard Name	Hazard Type	Causes of Incident	Hazard Zone
	001	NC/M	Machine	Hit by moving/falling object	
	002	Conveyor roll	Machine	Struck by something	
	003	Danger of falling spot	Human action	Falls from height	
	...				



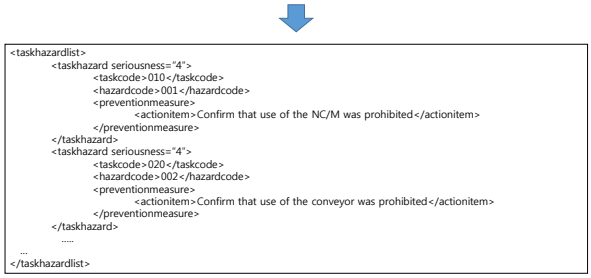
```
<hazardlist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="hse_in.xsd">
    <hazard cause="Hit by moving/falling object">
        <hazardcode>001</hazardcode>
        <hazardname>NC/M</hazardname>
        <hazardtype>Machine</hazardtype>
    </hazard>
    <hazard cause="Struck by something">
        <hazardcode>002</hazardcode>
        <hazardname>Conveyor roll</hazardname>
        <hazardtype>Machine</hazardtype>
    </hazard>
    <hazard cause="Falls from height">
        <hazardcode>003</hazardcode>
        <hazardname>Danger of falling spot</hazardname>
        <hazardtype>Human action</hazardtype>
    </hazard>
      .....
  ...

</hazardlist>
```

Figure 6. The XML file for hazard information

Figure 7 illustrates an example of XML representing the relationship between tasks and related hazard factors based on the XML Schema defined in Section III.

TaskHazard ID	Task Code Number	Hazard Code Number	Prevention Measure	Seriousness
	010	001	Confirm that use of the NC/M was prohibited	4
	020	002	Confirm that use of the conveyor was prohibited	4



```
<taskhazardlist>
    <taskhazard seriousness="4">
        <taskcode>010</taskcode>
        <hazardcode>001</hazardcode>
        <preventionmeasure>
            <actionitem>Confirm that use of the NC/M was prohibited</actionitem>
        </preventionmeasure>
    </taskhazard>
    <taskhazard seriousness="4">
        <taskcode>020</taskcode>
        <hazardcode>002</hazardcode>
        <preventionmeasure>
            <actionitem>Confirm that use of the conveyor was prohibited</actionitem>
        </preventionmeasure>
    </taskhazard>
      .....
  ...
</taskhazardlist>
```

Figure 7. The XML file for relationships between a task and hazards

The first value line of the table in Figure 7 shows the relationship between the task item which has 010 as the 'Task Code Number' and the hazard item which has 001 as the 'Hazard Code Number'. Similarly, the second line of the table in Figure 7 shows the relationship between the task item which has 020 as the 'Task Code Number' and the hazard item which has 002 as the 'Hazard Code Number'.

V. CONCLUSIONS

In order to ensure that the key information, such as task and hazard information for building safety management system is not inherent within the system but explicitly expressed in standardized format XML, we defined XML Schema to express such data in standardized XML formats. Based on it, diverse data of existing systems can be integrated and interoperated in standardized format for the safety management system. In the future, it is essential to conduct studies on how to integrate with a variety of existing systems on the basis of defined XML files. It is also required to develop a framework for the safety management system that can be integrated with existing process management systems.

ACKNOWLEDGMENT

 ISBN: 978-1-61208-448-0

This work was supported by the IT R&D program of MSIP/KEIT. [Development of Smart HSE System for Shipbuilding and Plant]

REFERENCES

[1] M. Golparvar-Fard, F. Peña-Mora, C. A. Arboleda, and S. H. Lee, "Visualization of construction progress monitoring with 4D simulation model overlaid on time-lapsed photographs", Journal of Computing in Civil Engineering vol. 23, no. 4, November/December 2009, pp. 391-404.

[2] M. Kiviniemi, K. Sulankivi, K. Kahkonen, T. Makela, and M. L. Merivirta, "BIM-based Safety Management and Communication for Building Construction", VTT Technical Research Centre of Finland, 2011, pp.1-118.

[3] B. Hadikusumo and S. Rowlinson, "Integration of virtually real construction model and design-for-safety-process database", Automation in Construction vol. 11, no. 5, 2002, pp. 501–509.

[4] Y. Mizuno, H. Kato, and S. Nishida, "Outdoor Augmented Reality for Direct Display of Hazard Information", IEEE SICE 2004 Annual Conference, vol. 1, 2004, pp. 831–836.

[5] C. H. Caldas, D. G. Torrent, and C. T. Haas, "Using global positioning system to improve materials-locating processes in industrial projects", Journal o f Construction Engineering and Management, vol. 132, no. 7, 2006, pp. 741–749.

[6] J. Song, C. T. Haas, and C. H. Caldas, "Tracking the location material on construction job sites", Journal of Construction Engineering and Management, vol. 132, no. 9, 2006, pp. 911–918.

[7] J. Gong and C. H. Caldas, "Data processing for real-time construction site spatial modeling", Automation in Construction, vol. 17, issue 5, 2008, pp. 526–535.

[8] J. Teizer, C. H. Caldas, and C. T. Haas, "Real-time three-dimensional occupancy grid modeling for the detection and tracking of construction resources", Journal of Construction Engineering and Management, vol. 133, no. 11, 2007, pp. 880–888.

[9] T. Aksorn and B. H. W. Hadikusumo, "Critical success factors influencing safety program performance in Thai construction projects". Safety Science, vol. 46, no. 4, 2008, pp. 709–727.

[10] O. N. Aneziris, et al., "Quantified risk assessment for fall from height", Safety Science, vo. 46, no. 2, 2008, pp. 198–220.

[11] H. Li, Z. Ma, Q. Shen, and S. Kong, "Virtual experiment of innovative construction operations", Automation in Construction, vol. 12, no. 5, 2003, pp. 561–575.

[12] C. S. Park and H. J. Kim, "A framework for construction safety management and visualization system", Automation in Construction, Vol.33, August 2013, pp. 95–103.

[13] K. Sulankivi, K. Kähkönen, T. Mäkelä, and M. Kiviniemi, "4D-BIM for construction safety planning", CIB 2010 World Congress proceedings, 2010, pp. 117–128.

[14] H. Yang, D. A. S. Chew, W. Wu, Z. P. Zhou, and Q. Li, "Design and implementation of an identification system in construction site safety for proactive accident prevention", Accident Analysis and Prevention, vol. 48, 2012, pp. 193-203.

[15] U. K. Lee, J. H. Kim, H. Cho, and K. I. Kang, "Development of a mobile safety monitoring system for construction sites", Automation in Construction, vol. 18, issue 3, May 2009, pp. 258–264.

[16] I. W. H. Fung, V. W. Y. Tam, T. Y. Lo, and L. L. H. Lu, "Developing a Risk Assessment Model for construction safety", International Journal of Project Management, vol. 28, issue 6, August 2010, pp. 593–600.

 ISBN: 978-1-61208-448-0