Interface Life Cycle Management within Enterprise Architecture

Lama Balloul[1], Jorge Marx Gómez[1], Andreas Winter[2]
[1]Department for Computer Science, Business Informatics,
[2]Department for Computer Science, Software Engineering,
Carl von Ossietzky University Oldenburg, 26129 Oldenburg,
{lama.balloul, jorge.marx.gomez}@uni-oldenburg.de
winter@se.uni-oldenburg.de

Abstract: Enterprise architectures provide reasonable information to guide evolution of complex technical systems within their organizational embeddings. In current approaches, viewing the impact of system changes is focused on analyzing components and neglects detailed views of interfaces connecting components. This paper presents a new approach, which considers interfaces as first-class objects to support decision making on evolving enterprise architectures and deal with change requests efficiently. Documentation using interface profiles is presented which connects to the enterprise data models and interaction mechanisms and allows tracing the impact of change syntactically and functionally. The approach is evaluated by using various metrics to estimate costs and risk of realizing requested changes within the enterprise architecture of a photo finishing company; CEWE COLOR AG & Co. OHG.

1 Motivation

Enterprises are driven by business both internally and externally, therefore IT systems are in continuous evolution in order to correspond to business requirements. Traditional maintenance activities and evolution are combined with high efforts and significant time [EFK+03, DLG+06]. Maintenance efforts increase in the presence of heterogeneous IT landscapes and out-of-date documentation [EFK+03]. The choice for taking CEWE COLOR AG & Co. OHG, a photo finishing company as a case study for this paper, is because it has a heterogeneous IT Landscape consisting of 16 Software systems, where half of them are in-house developed systems[1].
Hence, there are many interfaces in the line of duty in order to connect the software systems that are divided into three major parts; Online IT, Production IT and Commercial IT. Online IT is responsible for providing web services for end consumers and trade partners, presenting products of the company on the website, taking orders online and passing them to the production and commercial IT. Commercial IT receives orders from the trade partners, maintains their master data, products and pricing and provides both online and production with the required data. In addition, invoices are generated, converted to the proper format required by trade partners and electronically submitted by commercial IT.

[1]This Information has been collected from CEWE COLOR AG & Co. OHG

To appear in: Ursula Goltz, Marcus Magnor, Hans-Jürgen Appelrath, Herbert K. Matthies, Wolf-Tilo Balke, Lars Wolf, Eds., Beitragsband der 42. Jahrestagung der Gesellschaft für Informatik e.V. (GI), GI-Edition - Lecture Notes in Informatics (LNI), Bonner Köllen Verlag, 2012.

Production receives orders from both other sections and prices of products from the commercial IT. At the end of production, the orders are shipped and feedback is sent back so post-production processes can be initiated.

This IT landscape experiences continuous change to fulfill customers' needs. Employees invest a lot of time and effort to cope with changes that affect the company, namely project and daily changes. Project changes are in most cases well structured and managed with enough time to execute the required alteration and test them before going live. A recent example of project change is establishing a new Customer Relationship Management (CRM) system in the IT landscape. Daily changes, on the other hand, occur more frequently and have short time frames to be executed and tested. In some cases, these could coincide with high rate of mistakes even with skillful employees because they tend to prioritize rapid task completion over task correctness [Pet09a]. Without the help of a tool to determine the location of changes and with lack of adequate documentation, complications are common which results in tasks taking more time than expected.

Detecting the impact of change has been presented and discussed by different approaches toward enterprise architecture. Since the business is the source of change requests and IT is the place where these changes should be mapped, the focus was on aligning the business with IT which is managed by enterprise architecture [PS05]. Enterprise architectures provide comprehensive information to represent complex enterprises, in order to manage their evolution ensuing from technological alteration and increased complexity of information environments [Lan05, Kho07].
However, the matter is to decide if the changes should be applied and mapped onto the components within a software system, or on the interfaces that connect systems and components. So far, much effort has been made to study impact of changes at the code level of software systems. For programmers, this is helpful to capture the influenced details, but it is hard for high level decision makers to understand the impact of requirement changes and make an appropriate decision [LLYL08]. Hence, several studies have been directed to a higher level and focused on analyzing components. Detailed views of interfaces connecting components have been neglected even it is clear that the organizational effectiveness is driven by the relationships between components rather than by the detailed specification of each individual component and the local optimization [Lan05].

In this paper Section 1 discusses the approaches of enterprise architecture and their evolution in the following section 2 defines the problem by discussing the reasons for moving from management and analysis of components to interfaces. Section 3 defines the interface and its life cycle. Section 4 gives an overview of the taxonomy of changes' requests and their influence on interfaces. Section 5 presents initial results, outlines the metrics of interfaces and illustrates their usage with the help of an industrial example.

2 Related work: Enterprise approaches

Traditionally, enterprise architectures focus on IT related issues and artifacts to support better IT operations [WF07, Dav11]. According to Gartner Group [TG12], there are two approaches modeling enterprise architectures which differentiate from each other in the rate of change and the centralization of decision making. The first approach, called the *traditional approach*, tends to work well in cases where the architecture is driven by strategy i.e. when the organization has defined business strategy and manageable rate of changes in addition to a centralized decision making. This approach works poorly if it is expected to be flexible with emergent domains because it does not behave in a predictable way. Therefore, the *federal approach* is more suitable in cases when the decision making process is not centralized e.g. in large complex organization where the pace of change is very high.

However, in order to take advantage of management activities like change impact analysis, risk analysis and compliance, enterprise architecture should include business related artifacts [BW05, WF07]. As a response, enterprise architectures have become more business-oriented and new approaches emerged; the *managed diversity* and *middle-out* [Sch10]. Managed diversity tries to provide a balance between the need for standards and need for options. It provides a limited set of standards with a diversity of options for each component or service thereby reducing costs and complexity. The middle-out approach, on the other hand, could be costly but it is highly suited to very large organizations that experience a high rate of change. Middle-out has aroused criticism and been interpreted differently. The core idea of middle-out supposed to identify the central parts of an organization, that have the biggest impact on the ability to change and manages the connections and key dependencies between them [TG12, Pet09b].

The mentioned parts should firstly be clearly defined because they can be confusing. They could represent people, strategy, infrastructure, maybe the components or any part from the Mckinsey 7's framework [WPP80]. They could also represent the generic systems and federal components [Gal10]. Saggezza [Sag11] called this approach 'managed dependencies' because it focuses on interoperability and dependency management and on coordinating the behaviors among systems rather than optimizing their internal mechanisms. This approach is characterized by Gartner Group as "architect the lines, not the boxes" since it models the relationships as interactions using small set of interfaces and defines standard identity, format and protocol for each interface [MC07], therefore it is also known as *IFaP* (Identifier standard, Format standard and Protocol standard). Grigoriu [Gri10] and Rollings [Rol09] criticized this approach considering it nothing other than an SOA (Service Oriented Architecture). As a conclusion, middle-out is the precedent approach which changes the direction of interest from the parts to the relationships, yet the definition of the parts and relationships is ambiguous.

3 Challenge: Components vs. Interfaces

A driving force behind establishing an enterprise architecture is have a means to address continual internal and external changes of business strategy and the evolution of technology. Change impact analysis identifies the potential consequences of a change or estimate what needs to be modified to accomplish a change [BA96]. Much effort has been made to view the impact of change by analyzing components. In other words, the Change Impact Analysis (CIA) depicts how components are affected as a result of applying a change on one component but this does not mean automatically mean that all interfaces provided by a component that has changed should be altered. To specifically define the affected interfaces further investigation is required.
Each component comprises a set of interfaces and the application logic [AH01]. Each interface has method(s), input and output and a defined function to be performed in addition to transmitting data. Interfaces can be divided into two categories; those that provide services to other components or external environments, and those that require (used) services from other components or external environments [FM06]. A used interface is the same interface provided by another component. However, there are around twelve kinds of relationships between components, four of them, which are the most important, have been explained in [dBBG$^+$05]. Within the company, two types are used *Access* and *Use*.
Access Relationship means that one component accesses data objects provided by the other. For example, the orders from a consumer are saved in the database of OPS and are retrieved by other components from the rest of the company that need this information.
Use Relationship on the other hand, means that one component provides the information to the environment that depends on it, e.g. ICOS which provides master data to other components. The difference between these two kinds of relationships is, in the first case the interfaces are managed by the one who needs the information while in the second case it is managed by the one who is providing it.

In order to illustrate our basic idea, we consider the component ICOS[2] System-Trade Partner (TP) which is responsible for fostering the master data of trade partners and provides the data via interfaces to other components, see Figure 1. The master data includes the name of the trade partner, his phone number, his different subsidiaries and their addresses, his discount type, country, etc. SAP-ERP[3], as a system, is responsible for generating invoices; OPS[4] is responsible for offering the subsidiary data and DECENT[5] manages the pricing. The interfaces (TP-Master Data) differentiate in the form of the output and depend on the required data from other components, so the amount of provided data is often different. The discounts for the trade partners are transmitted separately to SAP-ERP SD. Applying a change on the ICOS-Trade Partner component, does not necessitate changing all interfaces which are provided by it. Changing the discount should by no means only require a change on the TP-Discount interface. In order to discover the affected interfaces and exclude the non affected interfaces, further analysis and study on the logic, function-

[2]Integrated CEWE COLOR Organization
[3]Enterprise-Resource-Planning
[4]Online Photo Service
[5]DECentral New Technology

ality and special cases of each interface should be done which means further effort and time, resulting in higher costs. However, if the initial focus was on the interfaces, the costs for impact of change analysis on components would be saved because each interface has a source and destination documented and can be easily retrieved.

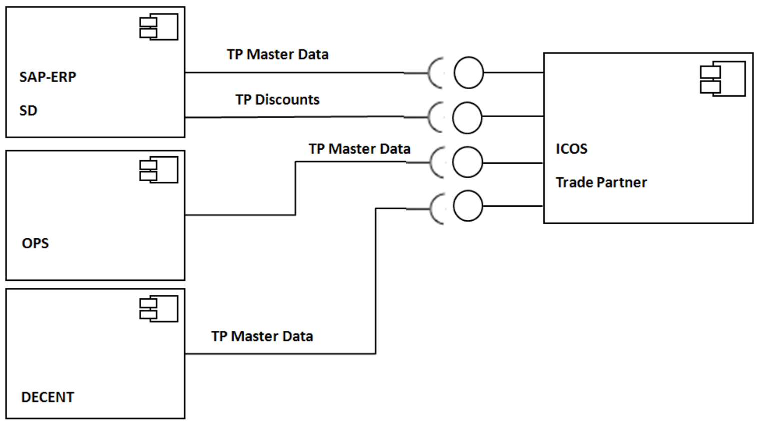

Figure 1: Interfaces within CEWE COLOR AG & Co. OHG.

In addition to the polygamy of component interfaces, component composition is another motivation to focus on interfaces rather than components. Component composition is used to build complex components from simpler ones [GS05]. The composition highly depends on the description of interfaces to know how a new complex component can be built in an effective way. The granularity of a component definition differs from one person to another and from a company to another. Therefore, degree of granularity should be settled from the beginning since interfaces follow the complexity of components.
For these reasons, we propose in this paper a new approach considering the interfaces instead of the components. Hereby, we adopt the statement of Gartner Group [TG12] ”architect the lines, not the boxes”. By applying the impact of change principle on the interfaces repository of an enterprise, represented by an interface profile -discussed in details in section 4.2-, we can determine the interfaces that are affected by a change request. By identifying the impacted interfaces, we can easily highlight the providing components and the requiring ones by parsing the modified version of documentation (interface profile).

4 Interface life cycle management

An interface is similar to an other IT or business element. It is like a product or a system. The lifecycle of an interface begins with a requirements definition moving to the design and then to the implementation combined with testing and configuration. The interface spends most of its life in the run-time phase. If it requires maintenance or customization to reflect business or technology changes then its lifecycle moves back to previous phases. When the interface does not corresponds with the changes, it has to be deactivated [Bal11].

4.1 Taxonomy of changes

Requirement changes usually driven by market, or by the customer whose satisfaction is boosted by companies. Additional requests could also be made internally to improve statistics for the managers or to reflect their business decisions. The following points outline the steps which could be taken in order to accommodate a change request:

1. Add a new interface: when a new function is required or a new connection should be established to a system or a component. Outsourcing is also a reason for adding new interfaces. We can also consider adding a new interface to supply the need for replacing an existed interface.
2. Deactivate an interface: either because its absolute technique or its functionality has been included or replaced by another interface.
3. Modify an interface: modifications are classified in three groups. Several changes on more than one level are possible.
    - Business level: the function of the interface. We use here tags (keywords) to specify the task done by an interface, e.g. convert, transfer, push, write, trade partner, order, customer, etc.
    - Logical level: this includes
        - Add the source and destination in the case of new interface
        - Change the source/destination in the case of outsourcing or assignment of functionality
        - add/delete one or more of the parameters (input, output)
        - Change the conditions e.g. The trigger of an interface (timely, manually, by event) or the sequence of occurrence
    - Technical level:
        - Used protocol/port in the case the customer changed its protocol
        - Representation of the input/output (char, number,etc., How many characters are used to represent a field)
        - The server on which the interface runs
        - The database on which the interface has access

These changes can be proofed syntactically after applying a mapping between the enterprise data model and the interfaces that are documented by an interface profile. The relation between them is (n) to (m). Each interface transfers (n) parameters which are represented in enterprise data model and each field in the model can be transferred by several interfaces. The mapping helps us to reflect any adjustment on the representation of a piece of data directly to the influenced interfaces. However, to check the logical modification we need to check the function of interface and the rules of interactions mechanisms i.e. dependencies. An example of dependencies of a dependency is trade partner billing. To create an invoice, trade partner master data, sales discounts and orders should be passed. The sequence rule is defined by transferring first trade partner data, then sales discounts, and lastly the orders. If trade partner data was not settled, no sales discount could be assigned and without the sales discount, faulty invoices could be generated. So any type of changes on one of these interfaces must consider dependency rules to avoid bad cases.

4.2 Interface profile

Interface profile provides a management summary of documentation and analysis for interfaces and includes the most important information for decision makers. Later, this profile is stored in an enterprise repository. However, the profile is not a substitute for the documentation. A structured and detailed template of documentation and the analysis background are generated but not presented here due to the limitations of space. The detailed documentation is always required to enable the implementation, modification, maintenance and testing [Pau10]. Unfortunately in many cases, this document is generated when the interface is first designed and implemented but not later modified, therefore it does not reflect the reality. In worst case, it could happen that the documentation does not exist at all. The reason for absence of or out-of-date documentation is high time consumption [Sin98, LSF03]. Therefore, an interface profile overlaps this problem as it is easier to maintain and to be kept updated. Nevertheless, the documentation of interfaces or even writing a profile for them and storing them in a repository is nothing new but the novel piece of this work is the analysis beyond this profile and connecting it with the data model and dependency model so that an assessment on the impact of changes on interfaces is easier and more accurate.

Example: Change request: Include CRM system to take care of orders from end consumer and trade partners and for marketing.
Reason: As yet, to get an overview of all orders, each time data has to be collected from several points in enterprise landscape.
Adding CRM is a project change. CRM is supposed to undertake the tasks which has been supported by **master data management system** and **online system**. To be able CRM to support the customers, it needs partner data, end consumer data, orders, prices from other systems (components). This data should be transmitted in an appropriate format to CRM via different interfaces. One of the new interfaces which has to connect ICOS with CRM is documented using interface profile, depicted by Table 1. This profile is stored in a database composing an enterprise repository for available interfaces and connecting to both the enterprise data model to retrieve input/output and the dependency model.

Responsible Person	John Smith
Source System	ICOS
Target System	SAP CRM
Functionality	transfers only new and changed trade partner data
Frequency of Usage	once a day, morning
Relationship Type	use
Business Process	Post-production process
Criticality (1 very critical - 4 not critical)	2 it is allowed to be a couple of hours out of order
Consequences of Failure	obsolete TP data in SAP CRM when adding a new one no orders can be assigned (dependency)
Effort (H) Implementation Effort (H) Changes Effort (H) Fix Error	15 man-days (120 Hour)
Input	Customer number, Trade partner number, company name, street, ZIP Code, City, ..
output	Trade partner number (extern), TPnr (intern), ..

Table 1: Interface Profile

4.3 Proof of concept

In order to validate the benefits of interface profile, a case study from CEWE COLOR AG & Co. OHG is presented. Change request: The delivery note number had to be unique across all branches of the company throughout the world so that it is easier for the trade partners.
Required steps to receive the request: the representation of the number had to be changed from eight characters to ten characters. Two extra characters had to be added to depict the laboratory. Some of the external interfaces were unfortunately not customized to handle the new representation. However, this challenge had been later solved through a collaboration with the customers. Some of the consequences the company faces:

1. in some cases, interfaces threw an internal error and no data could be sent.
2. the trade partner received the data but was unable to process it.
3. since the expectation was only for eight characters, the receiving system shortened the received data, which may produce invalid data.

Using interface profile, even the small number of bad cases are prevented. This is because the representation of delivery note number is modified in the enterprise data model. This modification throws a trigger to all interfaces which have this field in their input or output. Affected interfaces are retrieved from the repository with their source and destination to detect the influenced components/systems. The indirectly affected interfaces are retrieved by checking the dependency model. In order to determine the elaborated and risky interfaces, they should be first evaluated according to some metrics which are presented in the next section.

4.4 Evaluation of interfaces

To evaluate which interfaces have high costs and high risks to modify, some metrics are considered with their values and how the value may be interpreted. Some of the metrics can take an absolute value while the others depend on the context of the company and its branch. These metrics are: business value, frequency of usage, rate of usage, documentation availability, number of failures, number of modifications, duration of being out of order (in hours), effort until implementation. The values of the metrics should be set from the beginning particularly those which are context dependent.

Business value: this value can differentiate from one company to another. However, it is usually noticed that the interfaces, which are in the financial section [Bal12], are rated as very important, like the interfaces for sending invoices or salaries. The Business value derives its worth from functionality of the interface, its context and the consequences in the case of being out of order. For example, business factors play a big role in determining the value of an interface. The interfaces which transmit data to a strategic trade partner, partner with high turnover, or a new partner have more priority than others. Also the interfaces which are involved in high projects have higher priority than existing interfaces. The functionality of the interface plays a role in specifying its importance.
Frequency of usage: this value is context dependent and varies according to the business of the company. Here we give value 5 to an interface which is executed several times a day. Value 4 = once a day. Value 3 = once a week. Value 2 = once a month. Value 1 = once a year or manually. When the interface is executed by a trigger then it depends on the frequency of trigger occurrence. The more often the execution of an interface is, the faster the reaction is expected. The interface profile in Table 1 gives an example about an interface that is executed once per day. If it throws an error; no data or faulty data is transmitted, it has tolerance of some hours. The failure should be rectified before next time of execution (next day morning).
Rate of usage: since some interfaces transmit only new and changed data, the volume of data varies from one interface to another, e.g. interface transmitting trade partner data has higher rate of usage (data volume) than the one transmitting sales discounts, because trade partner data has higher level of change than sales discount. Partner's name, firm name, street, telephone number, city, ZIP code, branch are all belong to the master data of trade partner so the probability of changing partner's master data is higher than changing the sales discount which makes the former more critical with higher rate of errors.
Documentation: the value of documentation exceeds one's expectations especially by continuous change. If the person with knowledge of the interface left the company, it would be very expensive to reconstruct the documentation or apply a change on the interfaces. When it takes longer to solve an issue, the costs increase to both the customer and

the business. Good interface documentation includes the initial important information in addition to each change which influence the way in which interface works. In other words, good documentation should be up to date, provide both high-level overview plus low-level information. Value 2 is when the document contains only the initial version of the design (not up to date) or when it is unstructured with too many details which make it difficult to read and understand the document. Value 3 is when the document too short and does not provide enough detailed information to implement or make a change on the interface. Value 4 is when the documentation does not exist at all.
Number of failures: a high value for this metric indicates an interface that requires a large amount of effort and could threaten all depending tasks and interfaces. A failure is counted each time the interface is out of order or faulty when it is supposed to work correctly.
Number of modifications: when several modifications are necessary, the cost of the affiliated interface increase. Nevertheless, some of these costs are unavoidable especially when the customer's satisfaction is involved. Though a balance between these costs and importance to customers should be clearly defined.
Duration of being out of order: this value should be recorded only when it exceeds the tolerance of an interface. Our example shows a tolerance of some hours until the next execution. The longer this time is, the higher the associated risk.
Effort until implementation: this is also conditional and should be first determined by the company which would like to use this approach because the effort required for each implementation varies. This value can be compared with the average effort for the evaluated interface.
When the sum of the values for all metrics is small, the interface is neither risky nor costly. The first three metrics specify the importance while the rest define the costs of an interface.

These metrics have been discussed with CEWE COLOR AG & Co. OHG to determine their potential advantages. Evaluation of interfaces help the company to make a comparison between trade partner's revenue and the costs of the interfaces to fulfill requirements. On the basis of the comparison, managers could make their tactical and strategic decisions. IT managers would also get an objective depiction of the interfaces, their costs and risks avoiding subjective views of the employees.

5 Conclusion and Outlook

An important problem that demands a prompt solution with enterprise evolution is how to deal with change requests in an effective way. In this paper, the weak points and high costs of detecting impact of change by analyzing components have been argued and a new approach proposing analyzing of interfaces has been explored. Interface profile, as a solution for documentation problems and some changes requests are presented. Interface profile has been used successfully within CEWE COLOR AG & Co. OHG. Evaluation of interfaces will be evaluated in the same company and later in a bank in Switzerland. We aim to generalize the approach by surveying companies in different domains.

References

[AH01] Luca de Alfaro and Thomas A. Henzinger. Interface Theories for Component-Based Design. In *Proceedings of the First International Workshop on Embedded Software*, EMSOFT '01, pages 148–165, London, UK, UK, 2001. Springer-Verlag.

[BA96] Shawn A. Bohner and R.S. Arnold. *Software Change Impact Analysis*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1996.

[Bal11] Lama Balloul. Management of the Interfaces during their Life Cycle in a System Landscape. In Tom Mens, Yiannis Kanellopoulos, and Andreas Winter, editors, *2011 15th European Conference on Software Maintenance and Reengineering*, pages 385–388. IEEE Computer Society, 2011.

[Bal12] Lama Balloul. Interface Management: An approach to support IT-Managers in decision making. In *2012 International Conference on Information Resources Management (Conf-IRM 2012)*, 2012.

[BW05] Christian Braun and Robert Winter. A Comprehensive Enterprise Architecture Metamodel and Its Implementation Using a Metamodeling Platform. In *in: Enterprise Modelling and Information Systems Architectures, Proc. of the Workshop in*, pages 64–79. EMISA, 2005.

[Dav11] Rob Davis. Processes in Practice: Putting the E back into Enterprise Architecture. Website, July 2011. http://www.bptrends.com/publicationfiles/ 07-05-COL-rise

[dBBG$^+$05] Frank S. de Boer, Marcello M. Bonsangue, Luuk Groenewegen, Andries Stam, S. Stevens, and Leendert W. N. van der Torre. Change Impact Analysis of Enterprise Architectures. In *IRI'05*, pages 177–181, August 2005.

[DLG$^+$06] Jean-Luc David, Tony Loton, Erik Gunvaldson, Christopher Bowen, Noah Coad, and Darren Jefford. *Professional Visual Studio 2005 Team System.* wilez Publishing, 2006.

[EFK$^+$03] Kagan Erdil, Emily Finn, Kevin Keating, Jay Meattle, Sunyoung Park, and Deborah Yoon. Software Maintenance As Part of the Software Life Cycle. *Comp180 Software Engineering Project*, 2003.

[FM06] Tie Feng and Jonathan I. Maletic. Applying Dynamic Change Impact Analysis in Component-based Architecture Design. *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, International Conference on & Self-Assembling Wireless Networks, International Workshop on*, 0:43–48, 2006.

[Gal10] Nick Gall. The Role of Enterprise Architecture in Shaping BPM and SOA. Website, 2010. http://www.statcan.gc.ca/conferences/ it-ti2007/pdf/gall10-eng.pdf. Last visited: 03.05.2012.

[Gri10] Adrian Grigoriu. Enterprise Architecture Matters: Gartners EA hypecycle critique. Website, August 2010. http://www.ebizq.net/blogsea_matters201008 gartners-ea-hypecycle-critique.php. Last visited: 25.04.2012.

[GS05] Gregor Gösslerand and Joseph Sifakis. Composition for component-based modeling. *Science of Computer Programming*, 55(1-3):161–183, March 2005.

[Kho07] G. R Khoury. *A unified approach to enterprise architecture modelling*. PhD thesis, University of Technology, Sydney, 2007.

[Lan05] Marc Lankhorst. *Enterprise architecture at work: Modelling, Communication and Analysis*. Springer, Berlin [u.a.], 2005.

[LLYL08] Yin Li, Juan Li, Ye Yang, and Mingshu Li. Requirement-centric traceability for change impact analysis: a case study. In *Proceedings of the Software process, 2008 international conference on Making globally distributed software development a success story*, ICSP'08, pages 100–111, Berlin, Heidelberg, 2008. Springer-Verlag.

[LSF03] Timothy Lethbridge, Janice Singer, and Andrew Forward. How Software Engineers Use Documentation: The State of the Practice. *IEEE Software*, 20(6):35–39, 2003.

[MC07] R. Miller and R. Cherinka. Engineering a Complex Information Enterprise: A Case Study Architecting the Department of Defense Hourglass. In *Enterprise Information Systems and Web Technologies*, pages 183–190, 2007.

[Pau10] Debra Paul. *Business Analysis*, chapter 10. Documenting and Managing Requirements, pages 168–175. British Informatics Society Limited, second edition, 2010.

[Pet09a] Antoaneta P. Petkova. A theory of entrepreneurial learning from performance errors. *The International Entrepreneurship and Management Journal*, 5(4):345367, 2009.

[Pet09b] Christy Pettey. Gartner Identifies New Approach for Enterprise Architecture. Website, August 2009. http://www.gartner.com/it/ page.jsp?id=1124112. Last visited: 25.04.2012.

[PS05] Carla Marques Pereira and Pedro Sousa. Enterprise architecture: business and IT alignment. In *Proceedings of the 2005 ACM symposium on Applied computing*, SAC '05, pages 1344–1345, New York, NY, USA, 2005. ACM.

[Rol09] Mike Rollings. Gartner wakes out of an EA induced coma. Website, August 2009. http://eapblog.burtongroup.com/executive_advisory_progra/ 2009/08/gartner-wakes-out-of-an-ea-induced-coma.html. Last visited: 03.05.2012.

[Sag11] Saggezza. Saggezzas IT architecture services. Website, 2011. http://www.saggezza.comservices architecture. Last visited: 03.05.2012.

[Sch10] Jaap Schekkerman. STREAM: A Successful and Pragmatic 'Managed Diversity' Enterprise Architecture Approach. Website, 2010. http://www.enterprise-architecture.info/Images/STREAM/White

[Sin98] Janice Singer. Practices of Software Maintenance. In *ICSM*, pages 139–145, 1998.

[TG12] Ben Tudor and Laurence Goasduff. Gartner Predicts 95 Per Cent of Organisations Will Support Multiple Approaches to Enterprise Architecture by 2015. Website, April 2012. http://www.gartner.com/it/ page.jsp?id=1358913. Last visited: 19.04.2012.

[WF07] Robert Winter and Ronny Fischer. Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. *Journal of Enterprise Architecture*, 3(2):7–18, May 2007.

[WPP80] Robert H. Jr. Waterman, Thomas J. Peters, and Julien R. Phillips. Structure is not Organization. *Business Horizons*, 23(3):14–26, June 1980.

Acknowledgment

The authors would like to thank CEWE COLOR AG & Co. OHG for the valuable information, significant time and kindness especially the head of commercial IT.