Change Impact Analysis of Enterprise Architectures

F.S. de Boer[1,2] M.M. Bonsangue[2*†] L.P.J. Groenewegen[2]
A.W. Stam[2,3] S. Stevens[4] L. van der Torre[1,5]

[1]*CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands*
[2]*LIACS, Leiden University, The Netherlands*
[3]*Ordina SI&D Technology Consulting, Amersfoort, The Netherlands*
[4]*Innoveer Solutions, United Kingdom*
[5]*Delft University of Technology, The Netherlands*

Abstract

An enterprise architecture is a high-level description intended to capture the vision of an enterprise integrating all its dimensions: organization structure, business processes, and infrastructure. Every single part of an enterprise is subject to change, and each change may have significant consequences within all domains of the enterprise. A lot of effort is therefore devoted to maintaining the integrity of an architectural description.

In this paper we address the problem of mastering the ripple effects of a proposed change. This allows architects to assess the consequences of a particular change to the enterprise, in order to identify potential impacts of a change before it actually takes place.

1. Introduction

In a large modern enterprise, a rigorously defined framework is necessary to be able to capture a vision of the "entire system" in all its dimensions and complexity; an enterprise consists of many elements, resources, business processes and infrastructures. All these parts have a certain role and their relationships can be rather complex but are fundamental to achieve the goal of the enterprise. To manage this intrinsic complexity of an enterprise, architectural description is necessary. The added values of an architecture are to create insight, to aid communication between stakeholders, and to analyze the enterprise. One of the most important kinds of analysis of an enterprise is the assessment of the impact of changes.

*The research of Dr. Bonsangue has been made possible by a fellowship of the Royal Netherlands Academy of Arts and Sciences

†Corresponding author. Email: `marcello@liacs.nl`

Changes in an enterprise's strategy and business goal may have significant consequences within all domains of the enterprise, such as the organization structure, business processes, software systems, data management and technical infrastructure. The study of the ripple effect that a change may cause in an organization is called *change impact analysis* [1]. The goal of a change impact analysis is to see what would happen if a change occurs, before the change really takes place. This information can then be used to help in making a decision on the necessity of a change.

An example of a question where such an input is necessary is "what is the impact on my business processes if we modify our infrastructure by removing a server"? If an undesired impact occurs as a ripple effect of the proposed change, actions can be taken to prevent it.

Architectural description languages facilitate impact of change analysis because they allow 1) to extract information from the model to see where the first change will appear, and 2) to calculate change impact on the other parts of the model for the change proposals. For an enterprise, however, this means that an architectural description language has to offer an integrated view of the entire enterprise: only if such a description integrates the several perspectives of the enterprise, we can, for example, analyze the effect at a business level of a change that takes place at a technical level.

In this paper we propose a framework for mastering the ripple effects of the proposed change within ArchiMate models. ArchiMate [7, 6] is an enterprise modeling language that focuses on the description of several domains of an enterprise as well as the relevant relations between the domains. Within the ArchiMate language there are many concepts and relationships, each with its own intended semantics. Building on the semantics of these relationships, we define heuristic rules for calculating the direct impact

of a changed entity in an ArchiMate model on the ones related to it. Furthermore, we distinguish between different kinds of change, so to achieve a more fine-grained impact analysis. The effect of a change is then propagated in the model using the graph induced by the ArchiMate relationships, resulting in a labeling of all entities with the change they may need to undergo in order to maintain the integrity of the design.

Clearly this process can be easily mechanized, with as result a powerful tool for planning changes, making changes, accommodating certain types of changes, and tracing through the effects of changes.

The layout of this paper is as follows. In Section 2 we introduce the ArchiMate modeling language for enterprise architecture and a running example to explain our definitions. In Section 3 we explain the framework for calculating the ripple effect of a change in an enterprise description. In Section 4 we draw some conclusions from our work, mention related work and discuss possible future activities.

2 ArchiMate and ArchiCable

ArchiMate is an enterprise architecture modeling language [7, 6]. Based on a meta-model, it provides several concepts for architectural design at a very general level, covering business, applications, and infrastructure. The ArchiMate language resembles the business language Testbed [5] but it has also a UML-flavor, introducing concepts like interfaces, services, roles and collaborations.

Throughout this paper we will consider, as a running example, the enterprise architecture of a small TV cable company, called *ArchiCable*, modelled using the ArchiMate language. To describe this architecture we will need some ArchiMate concepts and relationships. It is not our intention to show the complete use of all ArchiMate concepts. Rather, we consider few basic ones in this example to be used for calculating the impact of change in the next section. More specifically, we will use structural concepts (role, component and data object) and behavioral ones (trigger, process and service). Concepts are connected by means of use, assign, realize, access, and triggering relationships.

A role is the representation of a collection of responsibilities that may be fulfilled by some entity capable of performing behavior. The assignment relation links processes with the roles that perform them. Further, roles can use services provided by others, where a service is a description of a functionality realized by some entities that make it available to the environment. Services are logically realized by a process and physically realized by a component. A component is a software entity that can be a reusable part of one or more applications, but also a complete software application, or information system. The passive counterpart of the component is the data object, used for the representation of data-type. Data objects can be observably accessed (i.e. modified, created, or deleted) by entities realizing services. Finally, the triggering relation between processes describes the temporal relation between them.

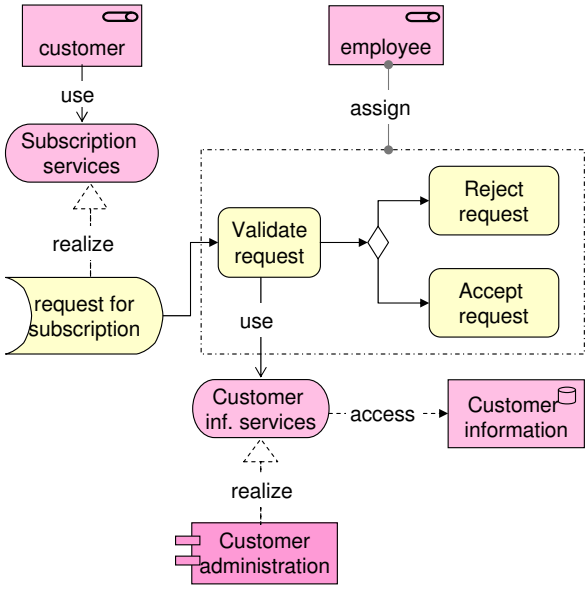

Figure 1. Subscription in ArchiCable

In Figure 1, we consider the ArchiMate description of the ArchiCable relative to an existing "customer" who may subscribe for some extra TV channels. Subscription is requested through the "subscription services". Such a request triggers the process "request subscription", that will initiate a process performed by an "employee" of the company. The "employee" first "validates the request" by using the "customer administration" application of the company to check the administrative situation of the customer, as recorded in the "customer information" database. If no problem arises, the employee will accept the request for subscription, otherwise the request will be rejected.

The proposed architectural description distinguishes two main layers: the *business layer* offers services to external customers that are realized in the organization by business processes (performed by business actors or roles), and the *application layer* supports the business layer with application services that are realized by software applications. More generally, ArchiMate allows for an integrated description of all dimensions of an enterprise, such as business, applications and technology, where the *technology layer* offers infrastructure services (e.g. processing, storage and communication services) needed to run applications, realized by computers and communication devices.

3. Calculating concept dependencies

An enterprise architecture represented by ArchiMate consists of a set of concepts and of some relationships connecting them. In this section we consider the impact of a change that takes place in a single concept. There are different kinds of changes that can be considered. In this paper we calculate the possible ripple effects of a change when the proposed change consists of removing, extending or modifying an existing concept of an ArchiMate diagram. By an *extension* here we mean the substitution of an entity with another one that preserves the information, behavior and structure of the initial entity. For example, the extension of a component can be another component providing more services. In contrast to extension, *modification* means the substitution of an entity with another one that (partially) destroys the initial information, behavior and structure. For example, a component can be modified by changing the services it provides.

To calculate the ripple effect of a change, we proceed as follows: we repeatedly change the labeling of the concepts of an ArchiMate model, until no change in the labeling is possible. The changes in the labeling are performed by calculating the immediate impact from one concept on another one, based on both the semantics of the relation that relates the concepts, and the kind of change we consider. In general, if two concepts are related, then the change on one of them does not need to have an impact on the other one: the presence of an impact mainly depends on the semantics of the relation between the two concepts. Relations thus provide the semantic context of each concept.

Within ArchiMate, twelve different relationships are distinguished. In the next paragraphs, we examine the impact of a change for the most important relationships, with respect to each different kind of change.

Access The access relationship models the access of a behavioral concept A to a data objects B. For example, consider the services "Customer information services" that access the data object "Customer information", for retrieving data about customers of the enterprise.

If the behavioral object A is deleted then it will have no immediate effect on the data object B, because B does not depend on A. However, an extension (or modification) of the process A may involve a new (or different, respectively) way of accessing the data in B. In order to maintain the integrity of the model, B has to be extended (or modified).

Conversely, if the data object B is deleted, then the process A cannot access B anymore. This in itself does not mean that the process A has to be changed, but rather that its access relationship remains dangling, a signal for the architecture maintainer to adjust the model by relinking the access relationship to another (or novel) appropriate data object.

If the data object B is extended then it still preserves the original structure of the data, and hence can be accessed by the process A without modification. However, if the data object B is modified then its data structure can be partially destroyed, and hence a modification of the way the process A is accessing it may be required.

Assign The assignment relationship links a unit of behavior A with an active element B that performs it. If we delete or modify A, this in principle does not have any impact on B, except for the case no unit of behavior is assigned to it anymore. In this case, the absence of an entity capable to perform the process B should be signalled. If we extend A, then it will still be able to execute the behavior B, i.e. the change on A does not have impact on B.

Conversely, a change in B leads to a respective change in A. For example, an extension of "validate request" with a check on the legal situation of a customer may require a new responsibility of the "employee" as to be able to use confidential data stored in a law court.

Use The use relationship describes the relation between a service A that is offered to the environment and an entity B in that environment that depends on the functionalities declared in A. If we delete A, then B will be affected because it cannot use the functionality of B anymore. This dangling dependency should be signalled. If A is extended then it will have no impact on B, as it can still use the older functionalities. However if A is modified, B may need to be modified as well, because the older functionalities may no longer be declared in A.

Conversely, deleting B will have no effect on A, but a modification or extension of B may require an appropriate modification or extension of A. For example, if the "customer information services" are deleted (because the application "customer administration" is removed, for example) then the process "validate request" will have a dangling dependency that needs to be signalled to the architect, who can either remove/modify the process or relink the use relationships to another appropriate service.

Realize The realization relationship links a logical entity A with a more concrete entity B that realizes it. The logical entity A is meant to declare to the environment some of the services realized by B, the concrete entity. If we delete B, the logical entity A does not have a reason to exist anymore and will be deleted as well, unless it is also by another concrete entity as well. If we extend B this does not have impact on A because the structure of B is preserved, whereas if we modify B this may require a modification of A as well.

Conversely, if we delete A this will have no effect on B, but if we modify or extend A this may require a respective change in B. For example, if the "subscription services" are modified by allowing only on-line subscription, the realizing trigger "request for subscription" has to be modified as well, by using, for example, new application software.

Trigger The trigger relationship describes the temporal or causal relationship between processes. A process A triggers another process B only after A has ended. Please note that triggering involves no transfer of data (for this purpose, the flow relationship should be used). This means that the two processes are independent of each other, so a change in one process will have no impact on the other. However, if the deletion of process A causes the situation that no trigger for process B is present anymore, this should be signalled to the user.

3.1. Some examples of impact analysis

Consider the situation that no support will given anymore for the current version of the application represented by the component "Customer information administration". Assume furthermore that an upgraded version exists, and that this version is guaranteed to be an extension of the older application. The company ArchiCable decides to upgrade this application, because an extension of the component will have no impact on the "customer information services".

Note that if the component is modified because it is substituted with a completely different one, then its impact may be rather large. In that case, for example, the "customer information services", the data object "customer information", and even the process "validate request" and the role of the "employee" executing it may need to be modified.

Finally, consider the following situation: although customers of ArchiCable show their enthusiasm about the fact that they can subscribe to extra channels, a new law imposes the company to offer all its channels in a single package only. As a consequence, ArchiCable decides to remove the "request for subscription" process. The impact will cause the removal of the "subscription services" (but the "customer" is not removed!) and results in a dangling trigger to the process "validate request", signalling the architect that the rest of the process cannot start and hence can be deleted too. This entire change may possibly even result in the reallocation of "employee".

4. Conclusion

In this paper we presented a novel analysis technique for enterprise architectures that involves analyzing change impact from a semantic perspective. This in contrast to the more usual syntactic perspective that calculates a dependency graph [1] between concepts of a model purely by composing the relationships between the concepts themselves. Our approach is inspired by the work of Kung et al. [3], who describes various sorts of relationships between classes in an object relation diagram, classifies types of changes that can occur in an object-oriented program, and presents a technique for determining change impact using the transitive closure of these relationships. Our technique for change impact analysis uses the semantic classification of the different relationships already present in the modeling language ArchiMate, and applies different kinds of atomic changes that could affect concepts in an enterprise.

In the recent years, a vast number of tools and techniques have been developed for change impact analysis depending on the definition of a software program, i.e. the code. Examples are forward static slicing techniques [10], Law and Rothermel's dynamic impact analysis based on whole-path profiling [8], and Zeller delta debugging [11]. Our change impact analysis technique is applied to *models instead of* code: change impact analysis for an enterprise is merely an analysis on the level of its conceptual structure.

We used a running example to demonstrate the possibility of implementing the proposed approach in an interactive tool. Admittedly, our running example is very simple, and in this case the effect of a change could have been easily extracted from the ArchiMate model. In reality, however, the model of an enterprise will be much larger, requiring techniques for selecting and visualizing the elements that are relevant for a particular stakeholder [7]. As a consequence, the identification of the concepts affected by a change becomes impossible without algorithmic techniques like the one we propose in this paper.

The outcome of an impact of change analysis can be used as a measure for the effort of a change: the more the change causes other rippling changes, in general, the higher the cost is. In the near future, we plan to study a suitable set of enterprise change impact analysis metrics. Despite the importance of change impact analysis on enterprise architectures, we are aware of works in this direction only for object-oriented modeling languages [4].

An interesting modification of our approach is to introduce more interaction when a change action causes a ripple effect, trying to understand the nature of the change by a refinement of the specification of the exact modifications within the model. This may lead to a better understanding of both the architectural design and the nature of the change.

Another future research possibility is the combination of change impact analysis with temporal logics [9] for a temporal reasoning about the impact of a change on processes. For this goal the logical viewpoint on architectures, as presented in [2], can be useful.

References

[1] R. S. Arnold and S. A. Bohner. An Introduction to Software Change Impact Analysis. In *Software Change Impact Analysis*, IEEE Computer Society Press 1996.

[2] F.S. de Boer, M.M. Bonsangue, J. Jacob, A. Stam, and L. van der Torre. A Logical Viewpoint on Architectures. In *Proceedings of the 8th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2004)*, IEEE Computer Society Press, 2004.

[3] D.C. Kung, J. Gao, P. Hsia, F. Wen, Y. Toyoshima, and C. Chen. Change impact identification in object oriented software maintenance. In *Proceedings of the International Conference on Software Maintenance*, 1994.

[4] Michelle L. Lee. Change Impact Analysis of Object-Oriented Software. PhD Thesis, 1998

[5] H. Eertink, W. Janssen, P. Oude Luttighuis, W. Teeuw, and C. Vissers. A business process design language. In *Proceedings of the 1st World Congress on Formal Methods*, 1999.

[6] H. Jonkers, M. Lankhorst, R. van Buuren, S. Hoppenbrouwers, M.M. Bonsangue, and L. van der Torre. Concepts for modeling enterprise architectures. *International Journal of Cooperative Information Systems*, 2004.

[7] M.M. Lankhorst (ed.), *Enterprise Architecture at Work: Modelling, Communication and Analysis*, Springer-Verlag, May 2005.

[8] J. Law, and G. Rothermel. Whole program path-based dynamic impact analysis. In *Proceedings of the International Conference on Software Engineering*, 2003.

[9] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.

[10] F. Tip. A survey of program slicing techniques. *J. of Programming Languages* 3(3):121–189, 1995.

[11] A. Zeller. Isolating Cause-Effect Chains from Computer Programs. In *Proceedings 10th International Symposium on the Foundations of Software Engineering (FSE-10)*, 2002.