Improving architectures by Simulation and Animation

A. Stam*
Ordina Institute
Groningenweg 6
2803 PV Gouda
The Netherlands
A.Stam@institute.nl

H.A. Proper
University of Nijmegen
Sub-faculty of Informatics
IRIS Group
Toernooiveld 1
6525 ED Nijmegen
The Netherlands
E.Proper@acm.org

23rd June 2004

PUBLISHED AS:

A. Stam and H.A. Proper. Improving Architectures by Simulation and Animation. In D.B.B. Rijsenbrij, editor, *Proceedings of the Second National Architecture Congres*, Amsterdam, The Netherlands, EU, November 2000.

Abstract

In this paper we present our ideas about using simulation and animation techniques to improve the understanding and the quality of architectures. We discuss the meaning of validation and verification for architectures and the value of simulation and animation in this context. We give an outline of what is needed in order to simulate systems based on their architecture and what can be accomplished by doing so. A small example is given to illustrate our ideas.

1 Introduction

Many organisations are confronted with the problem of shaping their organisational structures and ICT systems in such a way that they can be adapted quickly enough to cope with rapid contextual and socio-economical changes which influence their (way of doing) business. Flexibility and changeability become core issues and are relevant for both business and ICT [HV93]. Mechanisms are needed to swiftly assess the impact of adaptations and steer and manage changes in the organisational system.

Therefore, organisations try to capture (part of) their structures in a high-level description, which we call an *architecture*, which serves as a means to gain insight in the organisation of their activities and structures, as a prospect to steer the direction of ICT developments and as a basis for important decisions in several areas.

As such, a good architecture serves multiple goals at the same time and therefore must have at least the following qualities:

- *Clarity*: it should clarify the various structures of the organisation.
- *Manageability*: it should be able to be used as a steering mechanism for changes in these structures.
- *Flexibility*: it should enable change in these structures as quickly and simply as possible.
- *Realizability*: it should contain realistic models of structures that are to be realized in future, rather than idealistic models.

These qualities are key properties for an architecture, but they are difficult to obtain for various reasons. First, we have to deal with the unpredictable future for which, unfortunately enough, validation is impossible; *an archictecture must enable the exploitation of unforeseen opportunities* [Tap96]. Second, no proven methods exist for quantifying and measuring these properties; validation is often based on

*Also: University of Leiden, Leiden Institute for Advanced Computer Science

statistics and even more often on experience or intuition. Formal methods are, at least up to now, deficient in these situations.

In many fields of engineering, the same problem is recognized. When designing an airplane, for example, how do we know if it actually will be satisfactory and can be realized and maintained? When designing a new building, how can we be sure that it will not collapse when we actually construct it? When designing a new rollercoaster, how can we be sure that visitors will like this new attraction? A tool often used in these fields is *simulation*, i.e. experimentation with executable models and designs of real systems. Simulation of these systems based on their designs is often combined with *animation* techniques. The combination of simulation and animation often yield valuable insight into the qualities of a model.

We think that techniques of simulation and animation are applicable in the field of architectures as well, as an addition to existing validation and verification techniques. They can be used to get a better feeling for many qualities of an architecture or the system it describes[1], by working with scenarios and use cases or making changes in configurations and viewing their impact. In this paper we give a short overview of our ideas.

1.1 Organisation of this paper

This paper is organized as follows. In Sections 2 and 3 we give definitions for the terms validation, verification, simulation and animation. In Sections 4 and 5 we discuss what is needed in order to simulate a system based on its architecture, and what can be learnt from doing this. A small example to illustrate our ideas is given in Section 6. Finally, we draw conclusions in Section 7.

1.2 Terminology

We take the view that an architecture is in essence an abstract representation of a system. This system need not exist in reality (yet). An architecture consists of information about the system in the form of various models, each covering a specific aspect of the system. Architectures can contain models that describe the current state of a system, as well as models that describe a planned future state of this system. The models of an architecture should be fully integrated and connected to each other.

We make a distinction between the following uses of the word architecture:

- With *architecture* we mean the process of designing an architecture.
- With *the architecture* of a system we mean the entire "integrated whole" of abstract models and views on this system.
- All specific models contained in the architecture of a system have names like organisational architecture, software architecture, operational architecture, etc.

2 Validation and verification

In some literature a distinction is made between *validation*, which ensures that we *build the right system*, and *verification*, which ensures that we *build the system right*. *"Some authors distinguish between*

validation – an assessment of how the product we are building responds to the needs of the customer – and verification – an assessment of the internal correctness of the process (e.g., how a software architecture is correct with respect to the stated requirements). This distinction is motivated by the fact that requirements might not capture exactly what the customer wishes, so that a verified system might still be unacceptable." [GJM91]

In the light of architecture, both validation and verification are important. An architecture that does not respond to the needs of the customer is worthless; the very essence of designing architectures is to have a mechanism to increase this response! Therefore, validation is essential. Verification, on the other hand, is also of considerable importance, especially for keeping an architecture internally consistent. Architectures cover a wide spectrum of aspects; integration of these aspects is certainly not trivial.

[1] Note that there is a difference between desirable properties for an *architecture* (mentioned in Section 1) and those desirable for the *system* that is described by the architecture. For the system itself, properties like efficiency, reliability, portability, etc. are also important. For an overview of important quality criteria for (software) systems, see for example [FV99, ISO96].

The diversity of models within an architecture have to be verified against each other, because internal inconsistencies can lead to wrong decisions.

Guaranteeing qualities like clarity, realizability, manageability and flexibility is an assessment for which both validation and verification are important. An architectural model which describes a *future* system must be *valid* with regard to the needs of the customer, while an architectural model which describes an abstraction of an *existing* system must be *verified* against this system.[2]

3 Simulation and animation

Simulation and animation techniques can be very helpful for finding a valid solution for a problem. Instead of trying to *compute* a valid solution (which is impossible or infeasible for many problems), with simulation and animation we can try several candidate solutions and analyse their benefits and shortcomings.

3.1 Simulation

Simulation can be described as *the process of developing a simulation model of a real system and experimenting with this model in order to gain insight in its behaviour under various circumstances.* A *simulation model* consists of a description of the entities of a system and their relations such that it is possible to simulate the behaviour of this system over time.

A simulation process comprises three main phases:

- In the *induction* phase, information is collected about the system we want to simulate, and a simulation model is built.
- During the *deduction* phase, several experiments are done with the simulation model in order to achieve an optimal solution according to certain criteria.
- The *validation* phase runs in parallel with the two other phases. In this phase the validity of the simulation model (its correspondence to reality) is guaranteed.

In order to simulate a system, three things must hold. First, the system to be simulated must be demarcated clearly. It must be clear which part of reality belongs to the system, and which part forms its context. Second, it is necessary to have a predefined goal, which in many cases has to do with finding an optimal solution to a problem. Without a goal, it will be very complicated to choose the right entities, relations and parameters for the simulation model. Third, parameters must be *quantifiable* in order to be able to simulate something.

3.2 Animation

Animation comes from the latin word *animare*, which means "to give life to". As one picture says more than a thousand words, one animation says more than a thousand pictures. We have to *see* what is happening with a system and how it behaves under various circumstances in order to get a feeling for its complexity and to gain insight into its functioning. *Visual* simplicity and clarity, especially at a high level of abstraction, are important properties for relatively complex and abstract artefacts; these kind of properties can be validated with techniques like animation.

4 What is needed for simulation

With simulation, we try to abstract from reality in order to get a relatively simple model with which we can experiment. During the so-called *induction* phase in a simulation process, a simulation model is built, which is used to simulate a system. We try to ensure that the simulated behaviour mimics real behaviour as closely as possible within the limits of the abstract simulation model.

An architecture is also an abstraction from reality. It therefore does not differ much from a simulation model. The only thing needed is a *framework* in which we can represent all architectural information. Many frameworks for architectural description can be found in literature, for example in [SZ92, IEE00]

[2] The examples above are perhaps an indication for the fact that the original distinction between validation and verification is not really useful in the context of architectures. Maybe it would be better to use the term validation for all activities which deal with guaranteeing the semantic correspondence between models.

and they usually support a set of models in which all architectural information can be specified. Most frameworks support at least the following dimensions:

- Different kinds of *entities* that can be used, together with their entity-specific *properties*. Examples of these are "activity", "function", "actor", "trigger", etc.
- Different *contexts* in which these concepts can play a role and which can be used as different views on the same system. Examples are: "business context", "ICT context", "customer context", etc.
- Different *levels of abstraction* for modeling these entities in a specific context. Examples are: "enterprise level" and "department level" in an organisational context, "enterprise application level" and "executable component level" in an ICT context.

An architectural framework allows for the representation of an architecture within a set of models. However, an architecture specified within a framework is certainly not the same as a simulation model. A simulation model contains information which would not be specified in an architecture normally. Simulation requires additional information about the initial state of a system and about external triggers which cause activity within the system. It also requires detailed quantifiable information about the entities of the architecture, for example exact timing information for activities, or the maximum number of projects an employee can handle within a given time slot. This kind of information, which is not regarded as part of an architecture, can be classified as follows:

- additional *quantifiable properties* of entities within the architecture;
- a high-level description of the *behaviour of the context* that interacts with the system specified in the architecture;
- information about the *initial state* of the system and its context, in terms of entities of their high-level descriptions.

What we regard as "the system" and "its context" is dependent on what we want to simulate. Consider an architecture for an entire enterprise: a description of its business processes, its organisation, its ICT systems, etc. A simulation of the entire company could become very demanding, while a simulation of only one system or one department or one business process could be already quite informative. In the last case, we regard only the specific part we have chosen as "the system to be simulated", while its "context" is defined by only those parts which directly interact with the chosen part.

When we have chosen an architecture, represented it within a framework, enriched it with quantitative properties, described its context and specified the initial state of both the system and its context, we can start with a simulation.

5 What can be learned from simulation

The goal for simulating (part of) an architecture will always have to do something with optimization. Some examples of a simulation goal (in terms of a question) are:

- Which software system part forms the bottleneck for my production process?
- Which parts of my analysis software should I run on the mainframe?
- What would be the impact of replicating a software component on all my clients instead of running it on two servers? Should I buy a third server instead?
- Should I automatize this manual task or not?
- When changing this interface, which impact does it have?
- Which part of my software or hardware has to be expanded or replaced in order to double the size of my call centre?

All these goals are in fact optimization goals. Simulation will in no case yield *the* answer to these questions, partly because not all aspects of a problem can be modeled in general (for example political or emotional aspects), partly because we work with an abstraction of reality, a high-level architecture, instead of reality itself. Nevertheless, in many cases it will yield new information and insights.

Simulation can also be used for "measuring" the quality of an architecture regarding the properties we mentioned in Section 1: clarity, manageability, flexibility and realizability. "given the following change in the system, which architecture will perform best after the change? Which architecture needs minimal effort to implement this change?" This way, simulation adds value to the design of architectures.

As a third application, simulation can be used as a *verification mechanism*. An internal inconsistency in the models could lead to strange simulation results or could be detected before simulation is even possible.

Some parts of a simulation can be visualized and *animated*, like:

- The execution of a single business process
- Amount of database transactions; bottlenecks in batch systems
- Several flows, like information flows or control flows
- Activities performed by an employee in a simulated environment
- Inbound and outbound traffic for a specific department

Animation does not yield new results; instead, it yields an *insight* into the way a system works, which could be harder to gain from only a bunch of numbers (the result of a simulation). It could also be used as a *validation tool*: "Is it the architecture we had in mind? Does it behave like we want it to?"

6 Example

Consider a small company, named MatchIt, with 21 employees, providing a "matching" service to customers: Customers can sell or buy things to or from other customers, while MatchIt takes care of bringing the right people in contact with each other, in order to match supply and demand. Currently, there is one manager and there are 20 employees. The employees enter information about customers who sell products and customers who demand products, and try to find matches between them. Once a good match is found, employees inform the customers involved and supply contact information to them. Customers request mostly via telephone, but also via email. MatchIt has developed an ICT system to support its activities, which consists of a large database, environments for entering information about supplies and demands, and a specific analysis tool that supports the search for good matches. Currently, there are two establishments: one in Utrecht and one in Hilversum.

6.1 An architectural framework

In order to describe the architecture of MatchIt, we need a framework to represent the architectural information. We choose to make an architecture for the entire company and use the following framework for this:

- An *Organisational architecture* describes the different functions that are distinguished within MatchIt and gives an overview of the human resources.
- A *Service architecture* describes the services that the company offers and their underlying business processes.
- An *Application architecture* describes application components that are recognized in the ICT system of MatchIt.
- A *Hardware architecture* describes the available hardware and their relations, including physical information about their location.
- An *Operational architecture* combines the architectures above.

The operational architecture combines several parts of the other architectural models. Employees specified in the organisational architecture carry out services specified in the service architecture. In order to do so, they need certain applications specified in the application architecture. These applications run on hardware, specified in the hardware architecture. Thus, all models are related and their relations are specified in the operational architecture.

6.2 Creating a simulation model

As an example, Figure 1 contains the current hardware architecture of MatchIt, combined with the application architecture and employees from the organisational architecture. Currently, each employee performs a single task: either management, or matching, or entering supplies, or entering demands. Eight employees work in Hilversum, thirteen work in Utrecht. The hardware consists of two servers and several clients. Clients run software specific to the activities of one employee, while the servers contain

the common database and analysis software (which is too demanding for simple client systems). The main database is situated in Utrecht, where all matching is done, while in Hilversum there is a local database; its contents are sent to the main database in one batch once per day.

A complete simulation model for MatchIt would include much quantitative information about the entities within the system and a description of the behaviour of the context of the system (in this case, the behaviour of customers). Consider the following examples:

- MatchIt gets 200 telephone calls for supplies and 250 for demands in Hilversum each day (probably with a certain distribution), and 150 respectively 200 in Utrecht.
- Matches between supply and demand are found 50 times a day on average.
- A supply that matches a given demand is often found within two days.
- Entering information about a new supply takes between 2 and 10 minutes.
- It takes approximately one hour to copy the contents of the local database in Hilversum to the main database in Utrecht.
- Etcetera.

The simulation model can be made as rich as one wants. Information about the size of the database, information about telephone costs, etc. could also be included in the model.

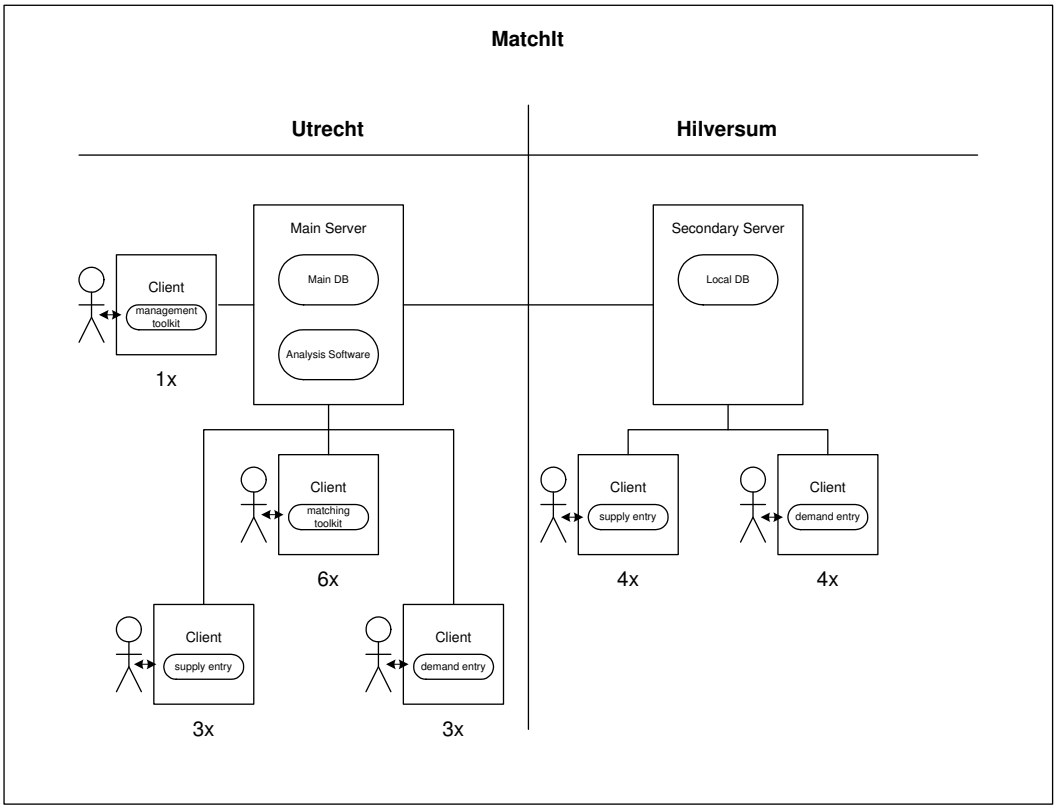

Figure 1: An overview of the example: hardware, application software and employees

6.3 Simulating the system

Suppose MatchIt wants to improve its organisation as follows.

1. The manager decides on the basis of recent inquiries that MatchIt should improve its customer intimacy. Instead of doing the same activity again and again, employees must help the same customer again and again. This means that when an employee gets a telephone from a customer supplying or demanding something, this same employee has to find a match for this customer.

2. In addition, the time between request and match must decrease from two days on average to four hours on average. The manager does not know how this can be achieved, but thinks that optimization of the internal workflow could help.

What is the impact of these changes? In our example, it is obvious that this means considerable reconfiguration of the entire architecture: the application component for matching must run on all clients now, analysis software must be deployed on the server in Hilversum, the main database has to be replicated in Hilversum once per four hours at least, etcetera. However, no one can foretell if the change for increasing customer intimacy will actually be beneficial to the requirement of decreasing the time between request and match. While it is relatively obvious to see which influence the first change has on the *structure* of the system, it is hard to tell how it will influence the *behaviour* of the system.

This is typically a situation in which it could be useful to simulate the system before and after the first change and look if it will actually perform better, worse, or equal. By using animation techniques as well, we can visualize the influence that changes in the configurations have on individual employees, on system performance, on information flows, etc. This visualization is especially important for direct feedback (instead of number crunching) as to better and more easily understand the dynamics of the entire system, as well as the quality of its architecture.

7 Conclusions and future work

In our opinion, clarity, manageability, flexibility and realizability are important properties of architectures for which formal methods are deficient up to now. We propose to use simulation and animation techniques in order to improve the understanding of an architecture and to validate these kind of qualities. Currently, there are systems for simulation of some architecture parts, like Testbed from the Telematics Institute [FJ98] for business processes. However, up to now there is no system which supports simulation and animation for entire architectures, including software and hardware. We think that simulation and animation can be used for the improvement of architectures in different ways:

- *conception*: gaining insight into an architecture;
- *anticipation*: estimating the impact of changes in the system;
- *validation*: comparing the system with the picture one has in mind;
- *verification*: verifying internal consistency or consistency with existing systems;
- *comparison*: comparing different architectures on the basis of specific qualities.

We would like to invite the reader to comment on the ideas presented in this paper. Together with CWI in Amsterdam, Leiden Institute for Advanced Computer Science (LIACS) and Nijmegen University, Ordina Institute is currently involved in the creation of a consortium with industrial partners in order to develop frameworks, integrate theories and develop a demonstrator for architecture based simulation and animation. Participation in this consortium is still possible for both universities and industry. If the reader is interested in participation, please contact the authors of this paper.

References

[FJ98] H.M. Franken and W. Janssen. Get a grip on changing business processes: Results from the Testbed-project. *Knowledge and process management*, 5(4):208–215, December 1998.

[FV99] M. Franckson and T.F. Verhoef, editors. *Specifying Deliverables*. Information Services Procurement Library. ten Hagen & Stam, Den Haag, The Netherlands, 1999. ISBN 9076304823

[GJM91] C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall International, Englewood Cliffs, New Jersey, 1991. ISBN 0138204322

[HV93] J.C. Henderson and N. Venkatraman. Strategic alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal*, 32(1):4–16, 1993.

[IEE00] Recommended Practice for Architectural Description of Software Intensive Systems. Technical Report IEEE P1471-2000, IEEE Standards Department, The Architecture Working Group of the Software Engineering Committee, September 2000. ISBN 0-738-12518-0
`http://www.ieee.org`

[ISO96] ISO. *Kwaliteit van softwareprodukten*. ten Hagen & Stam, Den Haag, The Netherlands, 1996. In Dutch. ISBN 9026724306

[SZ92] J.F. Sowa and J.A. Zachman. Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, 31(3):590–616, 1992.

[Tap96] D. Tapscott. *Digital Economy - Promise and peril in the age of networked intelligence*. McGraw-Hill, New York, New York, USA, 1996. ISBN 0070633428