Playing archimate models

Masters thesis

Jos Groenewegen

Preface

Being a student and walking the whole path through university is a special time. It is the time in which one grows from being an old adolescent to being a productive member of society. During my time at university I have had a great deal of opportunities to learn and grow. This learning was embedded in the curriculum by inspiring teachers and the things they had to teach.

Aside from the curricular activities there have also been extracurricular activities, each valuable in their own way. My work at university and the committees and boards I have been a part of.

All of these have helped shape me from who I was to who I am today. There are far too many names to mention everyone and as such I am not going to try. All those who I had such a great experience with over the last years know who they are.

I would like to extend special thanks however to Stijn Hoppenbrouwers, an excellent teacher and a great supervisor who has helped, and still helps, to make me a better person and better information scientist to this day.

Abstract

This thesis looks at the application of a developed game for the validation of archimate models in an architectural sense. The development process of the game is reviewed. The information transformed from the archimate model is highlighted, and an evaluation through application of the game is highlighted.

Furthermore the design principles behind the game are evaluated and the added value of such a method is discussed. Lastly the influence of this methodology of model understanding by the players and the conceptual merit and faults are discussed.

Table of Contents

Introduction, games for architecture?

As anyone who has ever worked on a large IT project will likely admit there are great difficulties in successfully completing any IT project. The problems range from 'simple' issues of complexity in the program to human errors and lack of knowledge about what the real goal of the IT project is. Solving these issues is one of the major aims of IT research today and this thesis aims to help in part solve some of the problems faced.

Specifically this thesis tries to help solve part of the problem of verifying the completeness and correctness of digital architecture models (From here on, architecture models). An issue that is related to many of the modern IT issues such as the modeling bottleneck and the high cost for the creation and maintenance of architectures.

It aims to do this by developing a proof of concept that game approached design can help to verify architectures.

This stems from the reasoning that often an architectural model is correctly created but still cannot be understood by non IT-schooled professionals. To validate architectural models the people who know the actual reality that has been modeled need to be able to understand the architecture. The intent of this thesis is to provide a proof of concept that a game based approach has great merit to help create this understanding of the architecture and through this allow the validation of the architecture.

To make the creation of such a proof of concept realistic within the constraints of a masters thesis we start by limiting the scope of what we look at. The concept of 'architecture' is broad and lacks any clear definition. We do not presume to give a verdict on what architectures are here. However to show that a game based approach can assist with the validation of architecture models we do not have to. Instead by having a game aid with the validation of some architecture we show that there is merit in the idea of having games assisting the validation of architecture.

The focus will lie on the method archimate due to pre-existing knowledge on this specific form of architecture as well as the relative clarity that architecture models in this language have.

So the aim of the thesis is the following;

"To provide a proof of concept for a game based approach to validating archimate models."

An important question that remains with this stated goal is what is a game? The definition of what a game is are as varied as the games that are out there [1] and just as for architecture we are not going to give 'the' definition for games. We will define games by a 'rough' set of demands to them.

Games have to be relatively quick to pick up, although they can be slow to master. Furthermore the pre-existing knowledge required needs to be clear.

Of course there are many other constraints that could be argued about. Do games need to be fun? Do you need multiple players? Is it essential for there to be a start and a finish? Although things worth arguing about questions such as these fall outside the scope of this thesis. If a 'game' is developed that can be picked up quickly and does not require extensive pre-existing knowledge (such as of architecture models) that helps to validate them then we will have made a significant step forward. The intent is to keep a broad open mind about the kind of game that can be developed and as such the other constraints are kept to a bare minimum. Now that we have an idea of the intent of the research it is time to turn to the methodology.

Method

The main aim of our research project was showing the possible strength of a game based approach to validating Archimate models. To realize this we set out to create a playable game to provide a proof of concept of the strength.

The research was set up on three consecutive research lines.

The first important thing to note is what kind of research we planned on to create the game. The methodology followed to achieve this is a design science approach[5] .

The way of evaluating this research (the game) is difficult. As shown in [6] there is no set of games that already meets the requirements we have for this game. Thus a simple comparison to other games or measures for the correctness of other games like this one is not an option.

At the same time there is no good measure for the quality of an architecture model yet. [7][8] Although there is considerable work, and indeed research, being done on developing such methods they do not yet exist. Seeing the lack of measures with a strong theoretical foundation and the constructive nature of the research the choice is made to go for empirical research. The fact that a proof of concept is the end goal in a field as of yet still partly undefined as architecture makes it necessary to support it with smaller scale empirical evidence and observations instead of giant studies. [9]

The first step followed was the extensive research of literature on what was available in the field of validation games. Aside from studying literature a cooperation was set up with the Dutch architectural forum (NAF's) workgroup on games to ensure that a broad perspective of what was available would be studied. Furthermore [1] and [6] were taken as

broad guidelines to cover the relevant example games and architecture to view. The archimate language itself was taken into the study as well to ensure a clear image of architecture such as we define it would be present in the game.

After the broad literature study on what games were out there the constructive part of the games development began. The methodology to create the game was based on methods to undertake the construction of a game [1][20][3] along with a healthy dose of creativity. As many of the sources stated say creativity is an essential part of

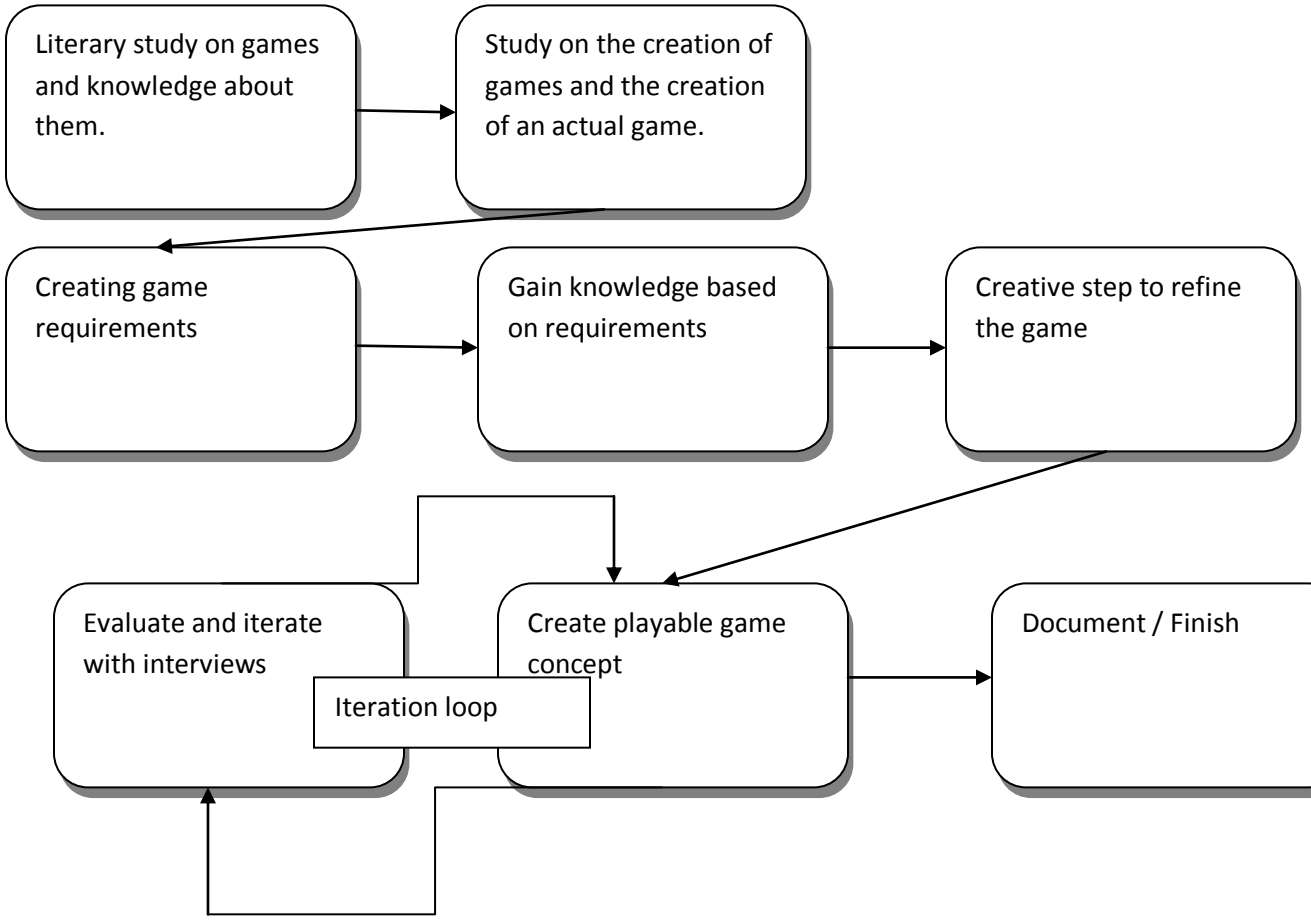
developing a game. Aside from this creativity a clear set of demands for the game along with the goals that you want to achieve with the game and who the possible participants are.

With the method for developing the game clear the final step still remained. Testing the correctness of the game for our goal. As stated in [1] and [9] this is no easy endeavour, especially in a field such as architecture. To show the effectiveness of the proof of concept we applied it to real life cases to see the effectiveness of the game and reviewed this. Interviews were held with participants before and after the sessions to try and measure their understanding of the architectural model as well as the correctness of the model. Furthermore the game was applied to the archisurance example [11] to see if it fit within the theoretical framework of the architecture. Although this by no means guarantees a perfect game it is a good enough measure to show if there is merit in the concept. Does it warrant further research or is there no hope for it, also looking at [12] and [13] and what they say about the merits and downsides of the game metaphor.

So to summarize we followed the following methodology, in three steps;

1. Study literature on the kind of games out there and the lessons they teach.
2. Look more in detail at prominent literature on how to develop a game and based on this construct an own game for a proof of concept.
3. Test the constructed game in one or more real-life situations and measure the effectiveness through interviews as well as apply it to the theoretical examples used in [11] to see if it performs as desired.

Based on three we then draw conclusions based on what we observed and looking back at our research aim. To show this in a model the method followed is roughly as follows;


Literary study on games and knowledge about them.
Study on the creation of games and the creation of an actual game.
Creating game requirements
Gain knowledge based on requirements
Creative step to refine the game
Evaluate and iterate with interviews
Create playable game concept
Document / Finish
Iteration loop

Types of architecture game

During the search for a game to help solve the issues presented earlier one of the things that was worked on was the general question of what kind of architectural games there are. A lot of work was done in exploration, alone as well as in combination with the workgroup of the Dutch architecture forum consisting of renowned architects, to get a clear understanding of what kind of games for architecture there are or could be. There is work being put into the creation of a lexicon of architecture games by the NAF's workgroup. However before this commenced a more general categorization to try and cover the fields of possible games was made.

The most interesting thing to view is not the lexicon in development but instead the categorization of the games that would 'cover' the field of architecture games

The associated categorization was put together in co-operation with the associated Dutch architecture forum workgroup on the subject and based on a few guiding principles.

First there could not be a high level classification based on a pure set of attributes. Games have a great deal of creativity in them and naming a few demands that a game has to meet to be a game, or fit into one of these categories, would destroy that. Instead the primary things that categorize a game are the intent of the game as well as the end result in the broader sense.

Second the categorization had to cover the breadth and scope of where games could be applied. From creating realization about the use of architecture to games that actually create architecture, although practical examples of these are still hard to find.

Finally there were 4 important steps in the architectural process that were recognized. For each of these 4 steps we found a 'category' of games. The 4 steps within the architecture process were;

1. Realization that architecture is necessary / has added value.
2. The creation of an architecture.
3. The analysis and validation of the created architecture.
4. The communication and spread of the architecture within the organization.

Obviously these steps do not always have to be executed in exactly this order or when executed always give a good architecture. Often enough the creation and validation of architecture is a cyclical process that knows many iterations, or the necessity of the added value is so self evident it requires no work.

However these are the 4 base components that every architecture runs through and that games could assist with. From these 4 process steps in the architecture process we came to the following 4 categories for games.

1. Convincing people of the added value of the concept of architecture through a game. Examples can be thought of such as the Ordina Alignment game or Van Beelen's IT governance game.

2. Creating architecture. A game or a number of games that support the creation of (representations of) architecture.

3. Analyzing and validating architecture.

4. Creating awareness of a completed architecture among the stakeholders. Important to note here is that the architecture is expected to be correct and stable and not need adaptation.

The categories themselves have proven remarkably robust and have so far covered everything we have come across or been able to think off theoretically. Obviously as insights progress so too might the 4 categories we have recognized.

However within this thesis these are the 4 categories of architecture games we will use. Further on when we refer to categories of architectural games these are the ones referred to.

Choice of type

The chosen category of the game is a choice based on things before and after the work on the thesis began. In part it is heavily influenced by the pre-existing work on games. As can be seen in work up till now [1][2] there are already games of some, but not all, categories. The example in [12] is a good example of a type 1 game. Similarly a good deal of type 4 games already exist, the lexicon in [6] shows a few examples.

So games of the 1st and 4th type already exist, leaving the most interesting research for games in the second and third categories. Before choosing which of these types was best careful thought was put into what was necessary to realize each. The second category of game would need to build from a knowledge of the nuance of the creation of architectures. What aspects come into it and how is an architecture made? The third category of games has different requirements. The architecture would already be there, so the question could not rest with how an architecture is made. Instead the questions that arise are ones concerning when an architecture is a good architecture. When is an architecture correct, and what requirements are there for completeness?

When viewing these two different kinds of knowledge required the creation of architectures has a large intangible aspect to it. Questions such as how one goes about the creation of an architecture are difficult to answer, and in fact some of the great questions in the field today. The current consensus is that it takes a good deal of experience and that not just any 'brookie' can easily develop good architectures. [5] The validation of architectures is a different branch all together. Of course it is not easy and takes a good deal of formal knowledge. But it is a lot better documented when an architecture is good and what aspects of a good architecture are. Furthermore the process that is followed is a fundamentally different one then for the creation of an architecture.[16] When one creates an architecture one has to go through a creative process and generally come up with 'new' knowledge. When one is creating architectures this has to be extended even further, the task is then to come up with a new way of creating new knowledge, meta knowledge creation. This seemed a bridge too far for a simple master thesis. Evaluating an already made creative step however was a far more attainable goal.

Based on these considerations the choice was made to develop a game of the third type.

The process of game development

There are a lot of ways to develop games. As can be seen in [1] and [6] the amount of games that can be thought off is limitless. And much as there is a great diversity in games there is also a great diversity in the process of game development. There are professional game developers ranging from 999 games to companies like Simagine with their very specialized and refined ways of developing games. At the same time there are games like the Ordina allignment game that are made by a group of entrepeneuring workers who feel a game approach will assist with a certain problem. The development of these games can range from one person having a brilliant creative influx to a slow refinement process in which a game through iterations gets better and better. Obviously there is also a near infinite combination of these possibilities. The important thing to note is there are almost as many ways to develop games as there are games.

The question remains what kind of process we will follow for the game development. To start the search goes for games similar to the one we seek to develop to learn what process was followed for their creation. Ss we saw in the previous paragraphs, 'category of the game' and 'kind of architecture games' the games of the third type that we know off are limited to none. So unfortunately there are no similar games whose development processes we can look at.

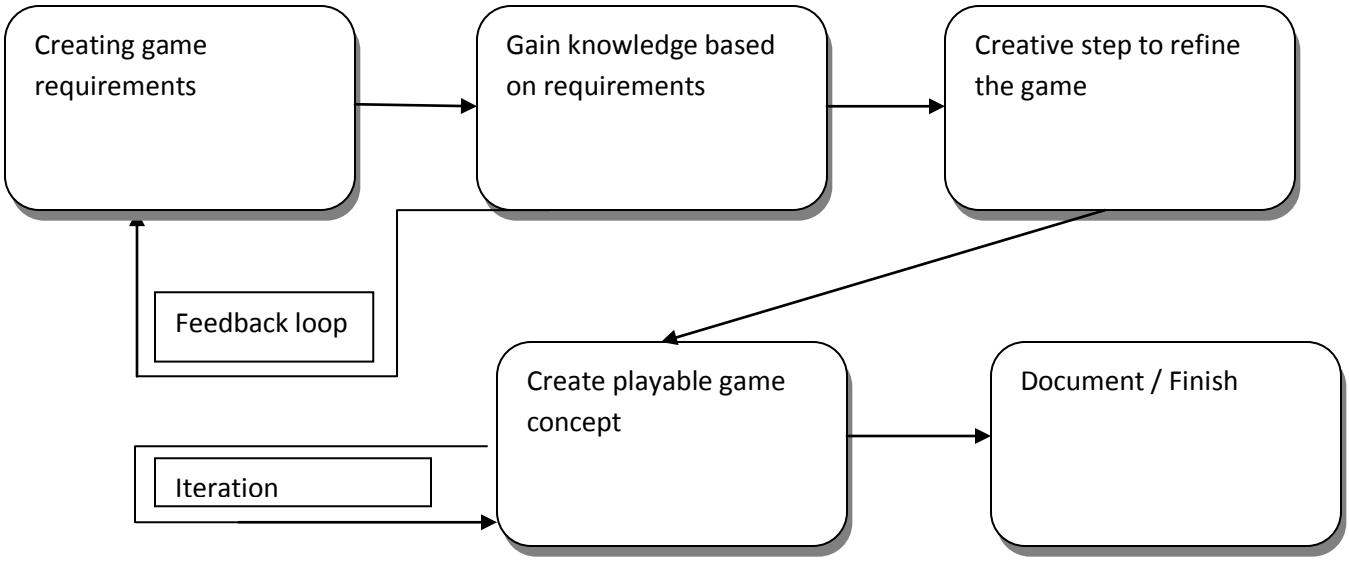
The first thing that had to be made clear during the development process was what the intention of the game and the game development process would be. Would the game need to provide a 100% perfect validation of every architecture, or merely a rough outline that an architecture was good? Was there a need for applicability to all architectures or could it be specialized per architecture? And even more basic questions, how long can the game take? How many players should it involve? As for the process questions arose such as; should the game that was developed be efficient? Useful in practice or merely a solid proof of concept?

The first part of the process of development was getting an outline of the requirements for the game. Before any truly 'creative' step took place the intention of the game as well as the game development was made clear. This was important due to the fact that the final aim of the game was already clear, a proof of concept for a type 3 game. As such not just any creative spur would suffice but only those that helped achieve this aim.

After the set of original demands was decided upon a second step in the development process had to be made. This was attaining the knowledge necessary for the restrictions within which we worked.

The game was to be based on archimate architectural models. Due to this a clear in depth study of the archimate language and just how archimate models were made was undertaken. The end goal of the game after all was not just to fulfill the requirements of a type 3 game, but to do this using the archimate modeling language. As this knowledge was gained the requirements were honed based on the new things learned. Certain aspects of the language, especially about actors that needed to be involved in validation, became clearer. Armed with the knowledge of the archimate language as well as the demands for the game, which we will go into later, the development process became a lot freer. Now the work of a creative step was undertaken, to actually develop a basic game concept that allowed the requirements we had set to be fulfilled, a new and innovative thing that was probably the least forcible part of the whole process.

After the creative step and with it the idea for a new game concept this concept was further iterated / tried out and honed to a final game concept. The development process is schematically given in the model below.



There are ‘general’ lessons that can be learned from this development, such as that requirements need to be made in an iterative model and that methods like waterfall design hardly work for complicated processes in distinct organization.

Furthermore two major lessons were learned. These were that the creative step had to follow after the creation of clear demands. This is due to the fact that before any kind of game is visualized the goal of the game should already be clear. It is better to have a clear idea and try to find a game concept for that then to have a game concept and forcibly try to fit goals into that.

The second important lesson was that after you had made the creative step it was best not to change the requirements again. You made a creative step based on a set of requirements; if the requirements are changed the creative step might no longer be able to ‘suit’ the requirements. As such the requirements had to be kept in mind, and still viewed critically. But if, after the creative step, the requirements still changed the creative step had to be re-examined to see if it still had the innate ability to fit the requirements. It was not just a matter of adapting it, it was actually a ‘go no go’ check every time. Now that something has been said about the development process it is time we go to actually look at the in depth knowledge. To start we will go on to talk about archimate.

Architectural basis, archimate

There are a great deal of architecture languages out there. Archimate is but one of many and the choice for it was arbitrary. The real choice faced was whether or not there could be a discussion about what architectures are or if it would be a given concept. Given the difficulties about such a discussion in this thesis we decided that ArchiMate models would be viewed as architectures. Given this and that there was pre-existing knowledge about Archimate it made for a good initial choice.

Aside from the pre-existing knowledge Archimate also has the strong advantage of being a standard. Although there is no single architecture standard yet archimate is one of the many standards. This means that there is a certain level of documentation and openness. Archimate is also focused on being an architecture language that takes all aspects of the enterprise architecture into its scope, instead of focusing on a specific part. [7][8] This helped give the insurance that no facets of the architecture would be forgotten in the game concept, as archimate covers them all. Still through this all the most importance message remains that the choice for archimate was one to avoid discussion about what architectures are. There can be a thousand definitions, the one given here is that archimate models are architectures.

Information transformed from archimate

Now that we know what we define as an architecture it is important to look at what information we can get out of an architecture. To get an idea of this we started by studying the design behind the language archimate itself as found in [11]. The aim was to see what information we could extract from the archimate models, in our case for a game. When looking into archimate several strict separations within the archimate language were found. The first split is one over the multiple layers of the systems. Into the business layer, the application layer and the technology layer.

The 3 layers are similar kind of systems, each being a dynamic system. However they focus on different parts of the system. However they are all covered in the same meta model. This means that to see what information we can transform from these 3 layers we can and will look at the encompassing more abstract meta-model. Of course in an actual implementation each of the layers has to be viewed independently as the actual information is in the models. The meta model is given in fig 1.

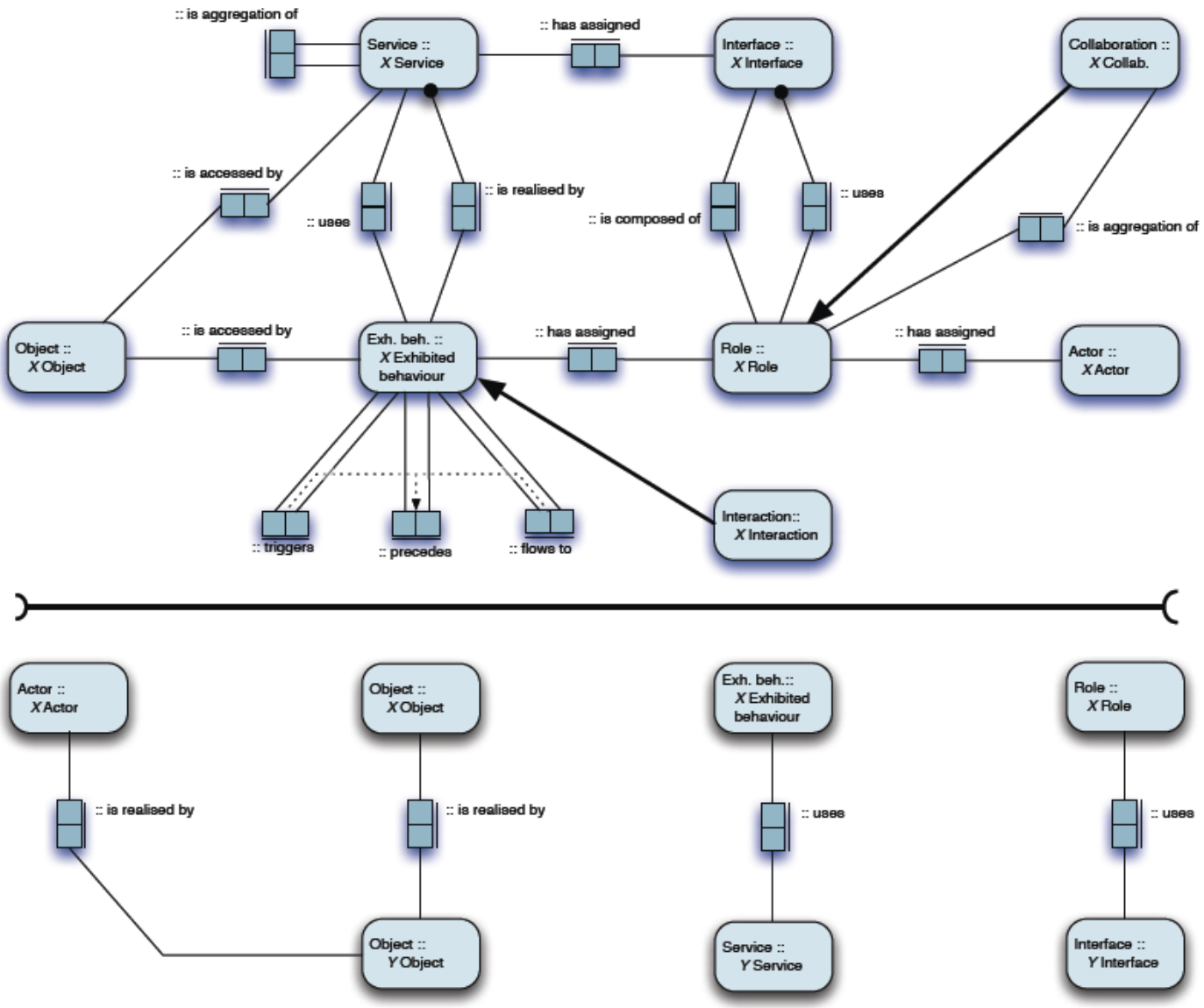


Figure 1 taken from [11], fig 7

The upper part of the model displays the build-up of all the layers, business, application and technology. The underlying fragment (under the bar) is needed to link the layers together in the larger architecture model. Important to note is that for some layers, such as the business layer, concepts like object are extended / refined. So they have several kinds of objects, however those are all still objects with extra properties.

Looking at the model it can be derived what information can be gained from archimate. Non exhaustively this is what objects there are and what behaviour they display. And furthermore what actors there are and what role they play. And the relations between all these, objects actors and roles.

It is worthy of note here that there is more information hidden in the architecture. This we will call the 'meta data reasoning'. Basically the stories behind the actors and objects. An object or actor is not just present but it was placed in a final architecture model with a reason. This is represented by the idea in archimate that every concept should have a clear contribution, both in the modeling language as well as in the actual architecture model.[7] [17]

The second conceptual split and associated information we can gather from Archimate is the more important one for our purposes. For dynamic systems (all relevant architectures are dynamic systems) archimate splits the components into three categories. The active structure concepts, the behavioural concepts and the passive structure concepts.

Basically who does things, what do they do, and what do they do them on. [11][7] A more precise definition is the following as taken from [11];

"Active structure concepts are concepts concerned with the execution of behaviour;

e.g., (human) actors, software applications or devices that display actual

behaviour. The behavioural concepts represent the actual behaviour, i.e., the processes

and activities that are performed. The active structure concepts can be

assigned to behavioural concepts, to show who (or what) performs the behaviour.

The passive structure concepts are the concepts upon which behaviour is performed."

These are the two main parameters we withdraw from archimate. The split into the three kinds of components and the relations between these components from the architectural model itself.

Archimate, what information do we miss?

When looking at the information we withdraw from archimate we see two streams in it. We see what kind of relations there are between components and we see the sort of components. However we also see an information gap in the kind of components, the 'context' data.

The split into three components is based on the buildup of natural language and the way that incorporates reasoning about active processes [11]. However as we can see in [13] there is more information within a natural language. There is what acts, on what it acts and what the action it takes is. This three way split that archimate makes is what you see in a normal action. However to have an *understanding* of the action a fourth parameter has to be added. Namely the why of the action. There is some reasoning behind the action that is undertaken. However in human behaviour, and similarly in architectures, this is not done simply because of some natural process. To reiterate this look at the last paragraph were we defined that every addition needed to have a good reason for existence.

There is some good reason for existence for every part of the architectural model, and the architect thought about the implementation. This data however does not seem to be captured in either of the meta models we have at present. Of course if the system is fully 'logical' it can be deduced from the model. Similarly if one is a domain expert it can be argued that they will know the reasons for relations. But architecture models are made to be understandable for those that do not yet know every detail of the system. And to make sense in 10 years, or for people from another corporation. As such with the aim of validation of architectures we will need additional information.

As a foundation for our game we will use the 3 type of components, active structure elements, behavioural elements and passive elements. It is a split based on natural language and one that humans can understand fairly well due to years of priming. [11][13] However as we saw before to get understanding of architecture the three components mentioned will not be enough. We will need a fourth type of component namely the reasons for behaviour. It is a reasoning that will lie behind the architectural choices if the architect followed the design principles. However it is not explicitly caught anywhere in the model.

We will work from the assumption we have both the model and the architects tacit knowledge about the choices made. With that

assumption we feel safe taking the narrative human language based approach. The three type of components about interaction with a fourth component pertaining to the WHY of the interactions backing them up. With this knowledge we have everything we need to verify the correctness of the architecture and reasoning behind it. Now it is time to start looking at what would make a game to do this work.[18]

A game for architecture, model validation

When looking at developing a game the first thing to do is to look at what makes games work. Fortunately a great deal of literature is available on this, sources such as [1] [2] [3] and [4]. Unfortunately the more these sources are studied the more the realization dawns there is no easy answer to what makes games work. Some are co-operative, others competitive and other games are played alone.

However even over this broad field of games some things do keep coming up. First there has to be some kind of interaction between actors. These actors can be humans, think of a chess match, a soccer match or monopoly. They can also be game parts interacting with humans, think of cards in solitaire being drawn by a player. Or even, in extreme cases, game parts interacting with other game parts. Think of a game piece in chess striking another game piece. Or a die roll that decides random events by the game on game pieces.

The second thing that keeps returning in games is that the games have a set of rules, broad or small, in which these actors can operate and interact. These interactions need to lead to 'some' end goal. This can be both a goal in the game or, as is often seen in corporate games, a meta goal outside of the game as well as a goal in the game.

When looking at a definition for games it becomes difficult, a good start is made in [4], they state that a game is;

"*A game is a system in which players **voluntarily** engage in a **goal-oriented**, artificial conflict, that results in a quantifiable outcome. **The activity takes the form of a process which is** defined by rules, **yet offers freedom of action**." [4]*

A good start, and indeed a definition that can be worked with. Even validating an architecture is a form of artificial conflict. After all, in some way the players need to overcome an artificial challenge (perhaps challenge would suit better then conflict), namely transforming a model to a model with a higher chance at validity.

There is a great deal of freedom when developing a game. There needs to be a goal, there need to be rules, a process, freedom of action, and an eventual outcome.

Still all of these elements do clearly return in the question we have posed. The goal is to get a better validated architecture. A quantifiable

outcome in that success can be achieved or not. There are rules that define the architecture within the game can work, and the activity of the game or validation is inherently a process.

So it is clear enough that within the very broad scope of what makes a game work a game can be developed to suit our question.

Aim of the game

What should be the precise aim of the game? The research question remains broad on what we want to do, to prove with a proof of concept there is merit in games for validation. What does that mean for the goal of this game?

With our chief goal in mind in the research question it is clear that a formal validation should not be the aim of the game. Many people have tried formal validations for architectures. Aside from the fact there is no an unambiguous widely accepted definition of what an architecture is the simple rules of when an architecture is valid vary too greatly and are too vague. The aim of the game instead lies in rough validation. The hope should not be to provide certainty on the perfect validation. Merely to get an idea that the architecture is better tested and tweaked then it was before.

The second aim of the game should be accessibility. The whole idea of a game to help with validation is that it has to have added value over the architectural model. This means that the capacity to understand the game should be different from those needed to understand the archimate model. In other words; The knowledge and skills needed to play the game should be different and not wholly include those needed to understand archimate models.

Finally another design choice crops up. The whole research focuses round the research question. This asks for a proof of concept, this means that the aim of the game will not lie in perfection. The game does not yet need to be perfect in neither investment nor executability. It has to be good enough in these to prove that the method of using games is sound.

Looking at these choices for the aim they can be summarized thus in no particular order:

1. Rough validation of an archimate model
2. Accessibility meaning different demands to play the game then to understand archimate models
3. The game is a proof of concept, not a final perfect product

Design choices for the game

With the aims of the game clear in mind we now look at the design choices for the game.

The first design choice made for the game is that we want to have the certainty of all the knowledge necessary. In other words, if knowledge is required the game needs to be able to 'get' it into the game.

The first choice we make from that is that the game will not be player based. The employees within an organization as well as the architect should all be potential actors within the game, might their knowledge be required.

No constraints on players involved with the architecture, everyone is a possible participant.

In other words, the game will not be made to players but players shall be found to suit whatever game is made. However, to be realistic this constraint has to be limited. The game cannot demand to involve outside experts. The game should be playable with only those involved with the architecture, outside players should not be a necessity.

The second choice is that we will work from the architectural model. The aim is to help validate the archimate model. This means that a one way transformation from the archimate model to something else is undesirable. Whatever final form the game takes after the transformations it still needs to benefit the validation of the initial archimate model.

The archimate model is the foundation for the game and has to be able to directly benefit in its validation from the game.

As seen in the information transformed from archimate there is the 3 layer model of passive, active and behavioural structure elements. It was also earlier that the most important information in our view (as per [21]) the reason for the acting was not necessarily present in the model. This returns in the game to ensure the completeness we aim for.

The reasons behind the interactions between the elements of the archimate model have to be incorporated in the game.

With this we have a sufficient set of aims and design choices to work with the actual design of a game. Let us look at these in a short list;

1. Anyone, and solely those, involved with the architecture need to be possible participants with the game (Full knowledge accessibility).
2. The archimate model is the foundation for the game and has to be able to benefit from the game in its validation due to this (No 'new' core architecture, what is the architecture is defined).
3. The fullness of knowledge needs to be a part of the game, implicitly or explicitly. (Validation requires full knowledge content).

The game

With the last chapters it was explained what the rough aim of the game, the validation of architectures, was. The constraints aims and choices for the development of the game were also explicitated.

Furthermore it was explicitated how the game development process in general has gone.

So what was the result for this specific validation game?

To start once more with the 3 pillars the game was build upon;

1. The architect is present for the game preparation or playing.
2. The players, or really their knowledge, can be selected as players for the game.
3. The players need no great architectural knowledge.

Let us start with a short reasoning for each of these pillars, to start with the first.

As was noted in; 'Archimate what information do we miss' and [11][13] the context information for why relations are there is not stored in the architecture. The information is present in the architects head. Since this information is essential for the understanding of the architecture this knowledge will have to be gotten from him either in preparation off or during the game.

The second comes with the fullness of knowledge we saw. To validate the reasons the architect has for his architectural choices domain experts are necessary. People who know the domain the architecture is about and can judge whether the understanding of it was correct. To get this knowledge these people need to be involved as players. Similarly people without knowledge of the domain cannot actively participate. This means that the players together need to have a, rough, covering of the knowledge in the architecture or that part that is being validated.[8][13] This means we assume we can select the players who will have this knowledge. The choice was the 'easy' way to ensure the necessary information was present in the game. It is obviously a very different choice from formal proving tools. Normally a game has to be playable regardless of the players, if they meet certain minimum requirements of knowledge. Here we deviated from this normal idea in that the players are strictly selected. And not just that the role the player plays within the game might also be heavily influenced based on the knowledge he has. It seemed the best way to bring the knowledge

of the players and the knowledge needed for parts of the architecture together.

The third pillar may seem strange but is perhaps the most important one. Those people who already have a strong understanding of archimate can just read the models. The whole necessity of the game lies in the fact that those people mentioned in pillar 2 do not have the architectural knowledge. To involve those players with knowledge but without knowledge of archimate the game needs to be accessible.

Knowledge model and creative step

Now that it is clear what pillars the game would be build upon and with the design choices and constraints explained in game design we arrive at the creative step. As made clear in general game design the truly difficult step, as it cannot be forced. As was analyzed in the missing information the main gap spotted is the lack of transfer of the tacit knowledge of the architect, specifically what purpose relations serve, and the difficulty validating this.

The knowledge is largely tacit and fluid, often process knowledge, and cannot be 'just' written down. If it could be easily formalized or written down it would already be incorporated in the architecture itself. It is an experience based feeling of how things work and how the process flows that both the people doing the work feel and the architect have when the architect makes the architecture.[19] The first step in the creative process was the realization that this was the most difficult to validate in a normal context and a game would be an ideal place to try validate this. A game allows for an experience of a non-linear process and it becomes possible to through that understand the intention of the relations the architect has put in the architecture.

To be able to get this experience it is important to take all the knowledge about what is involved in the interaction, which is in the architecture model, as argued in [13]. As we saw earlier in the explanation of archimate archimate has 3 components that fulfill the role, active structure components, behavioural components and passive structure components.[11][13] The idea was there should be some way to incorporate these 3 components and through living through the experience automatically create the tacit knowledge, or feeling that it is missing, between them. This way the architect and players would both get a fairly good idea if the architecture fit with both their views.

At this point it took awhile before a creative step was made. Locked in the position of having to do 'something' with the three aforementioned components that together make up all the parts in an architecture, and the knowledge of the relations between them.

Eventually the step just 'snapped in', largely after studying on the structure of the sentences again. Active components (active structure elements) display behaviour and they do this on passive structure elements. The possible behaviour that is displayed as well as by whom

and on whom is all captured in the architecture model. It is important to note here that these are rarely 1-way relations. Going through the architecture and having a temporal component is essential, as a client can soon also become an actor on a previous actor (then client).

From here we went to the step that every interaction / relation in an architecture is basically one of these relations. An actor acts in a certain way on a client, for a reason. Furthermore in Archimate such interactions only occur when a step is taken in the model. Through this it is possible to see every active action taken and in this way step through the model.

This means that it is possible to make all the active steps in an architecture and, in this way, see a few things. Firstly the fact of whether or not the architecture is complete in itself can easily be seen. If one ends up with two architectural parts which have no meaningful interaction with each other or each others resources then it is already clear that it is not complete. Obviously this is a situation that will rarely occur as most architects already check this.

More importantly though by making the steps through the model one can incorporate a temporal aspect into the relations. Every active structure element has a start point in the architecture, a path of 'influence' through the architecture and an end point where the original actor ends up. If this line is marked its path through the architecture and the actions it has undertaken can be seen, followed, and understood. Understood at least if at every action it takes one has to think about why it is taking a certain action.

And that was the creative step that lay at the core of the game progress. The players needed to be forced to think actively about the behaviour the actors displayed. The way chosen to do this was to have the players play the actors associated with their field of knowledge. Every actor has a start point and an end point in the architecture, not necessarily different ones, and paths to get there.

The players' choices at every possible step are the behaviour of the actor, and should coincide with the behaviour the architect had in mind. If this is the case and all paths can be followed successfully and cover everything then the following situation is reached.

The domain expert has covered the idea of the architect behind his architecture and it fits with his own view of the real situation. Furthermore it covers all aspects of the architecture that he knows and he finds no flaws in it. The choice is made here to assume that if an architect has a correct idea of the real situation the model he has made to display that is correct. The players do not have the knowledge to check the actual model, and it seems best to assume the competence of the architect.

It is time to take a look at what type of game this leads to. After taking a look at the actual game rules we will look at how to transform an archimate model into a playable game model.

Game instructions

Preparation

Place different coloured pawns (or marked pawns) at each starting location marked by the game master (Typically the architect). Each marked start location has one or more designated end locations. The map is laid out, meaning the assignment points are laid down as well as the routes between them (which are still locked).

Start

Each player moves in turn starting with the youngest player. A player goes through the following steps (Some of which are not always performed, depending on the situation).

1. Move a pawn to any 'reachable' assignment, and try to complete the assignment
2. Pick up available items on routes or in squares to assist with or as required for assignments
3. Trade items with another pawn it can reach.
4. Drop available items at a tile or on a route it can reach for other pawns to pick up

A player can go through the steps 2 to 4 as often as he likes. Step 1 may be done at most 3 times in a row before another player gets a turn (and the opportunity to skip it if he wants).

Game goal

Pawns start at their designated start location and can move across any explored path as far as they want. They have one or more designated end points. When an unexplored path is entered the pawn has to 'open' the associated card (provided by the game master) and complete the assignment to continue. If the pawn withdraws, after reading the assignment, the path stays closed.

Opened paths can generate passive structure elements in their own or other tiles (as described on the card).

Once a pawn reaches an end point the route of every step where it undertook action is ‘marked’ and a new pawn is generated at the start location.

The game ends when a line can be drawn, across marked squares and routes, between all end locations.

Variant

If the players wish it is also possible to merely mark start locations and possible routes from there. This allows a greater sense of exploration although it can slow the progress of the game.

Game preparations game master

As seen these are fairly simple instructions. Although the instructions are simple the preparation for the game is a bit more complex. To start there has to be a finished archimate model to work with. Within this model all active structure elements need to be marked. Any active structure element that comes from an earlier ‘interaction’ or involves an actor of an earlier interaction is marked as an assignment point. If there are no further routes to take for an actor from an assignment point it is also marked as an end location.

Any active structure element with no previous location is marked as a starting location.

The passive structure elements are incorporated into the game as items that can be carried along (or left or locked in a square if the item demands).

The behavioural elements are the choices that force people down a route. Furthermore they can also be ‘nodes’ where several routes cross and are several more routes. Hence a behavioural choice. So they are represented as nodes where one or more routes meet going to other nodes. A node can be the creator of passive structure elements, an assignment, or merely a crossroad.

Reading it like this it might seem difficult to see exactly what goes into it. To help clarify some below is provided a first simple game transformation, for the archisurance example.

First let us look at the archisurance model taken from [18] fig 8

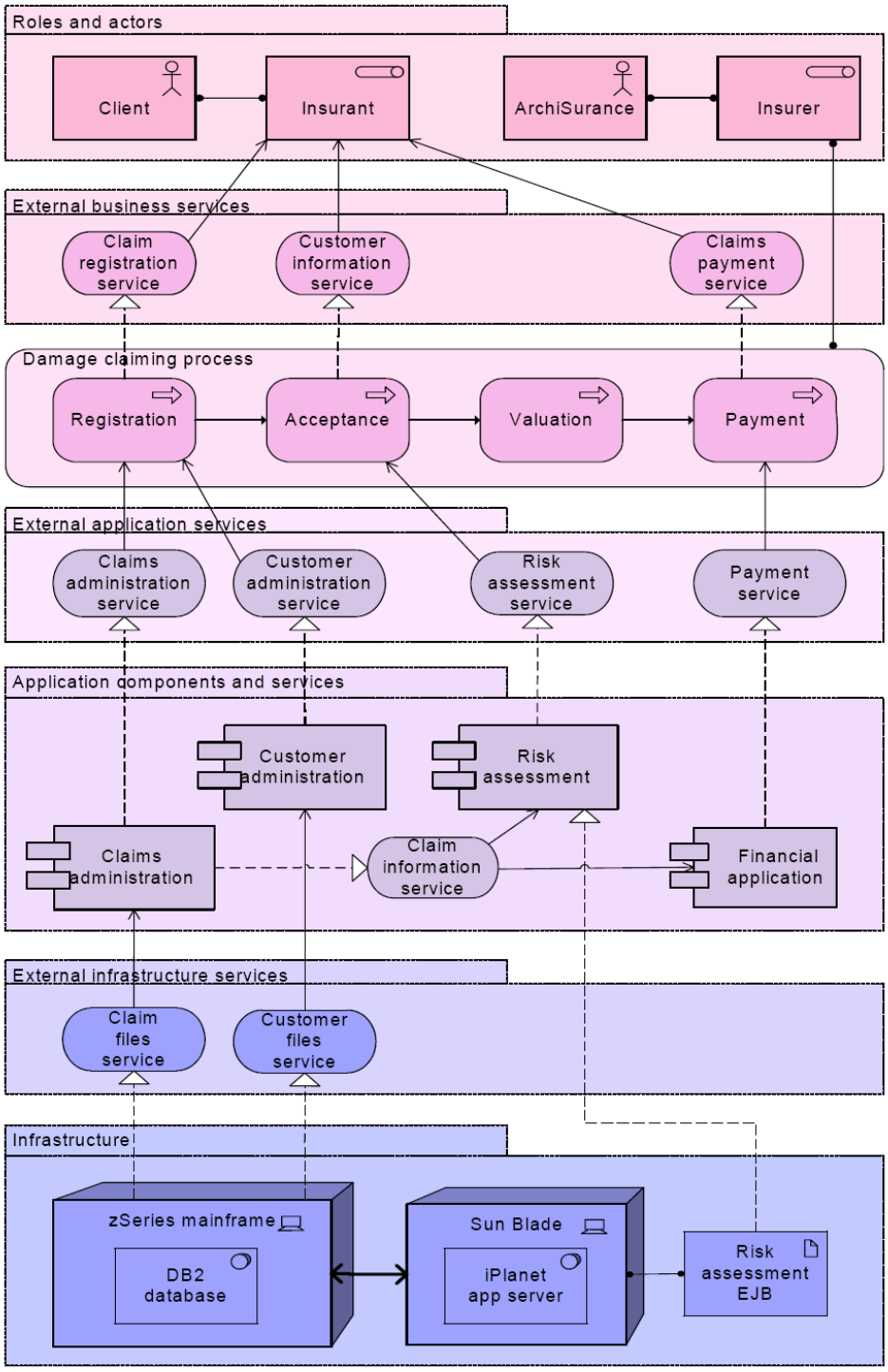


Figure 8. Example of an integrated enterprise architecture.

Figure 2, Fig 8 taken from [18]

Now within this model we first need to mark the components. Red squares surrounding a box mean it is an active structure element. A yellow 'line' round it means it is a start location as well. A green line with the red box means it is an end location. To make this model takes the originals architects knowledge of the model. For archisurance unfortunately we do not have this. So there is some liberty taken in the original intent and meaning behind this model. The blue squares indicate passive structure elements The yellow squares indicate it is a behavioural element. This leads to the following marked model.

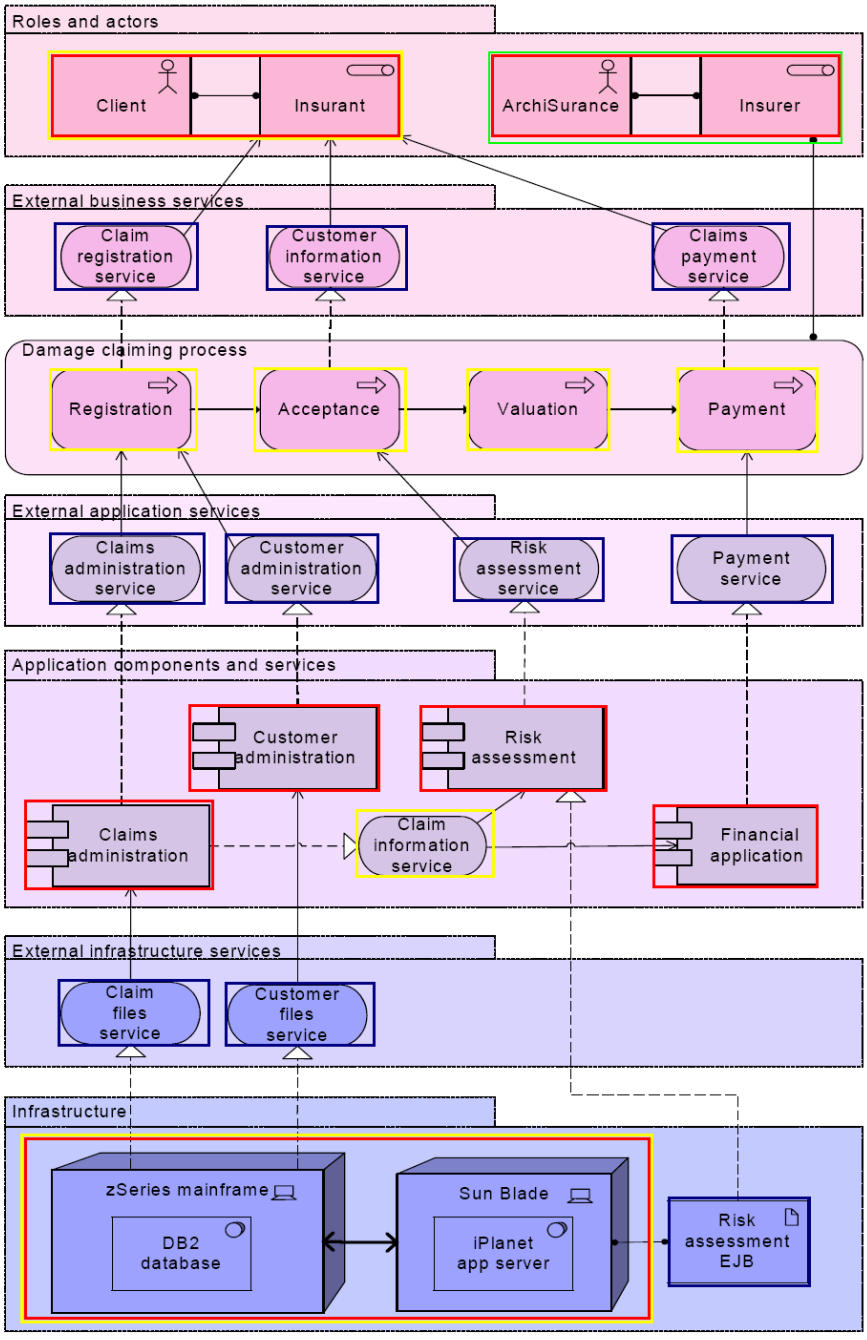


Figure 3

Obviously in the marked model above there can be a lot more focus as well. As we know of the archisurance example the registration step (as well as the acceptance, valuation etc) have more detail then ‘just’ a step. This detail can be caught either by expanding this model (making a really big one) or making ‘sub’ maps. One can go into a registration area and walk through that, starting with a certain set of elements they carry and coming out with different or modified ones, or just having walked and marked the paths. For the sake of workability in this example we will not add all the in-depth options further expanding the model.

Now let us take a further look at the archisurance model, marking two possible routes through it, to see how that works.

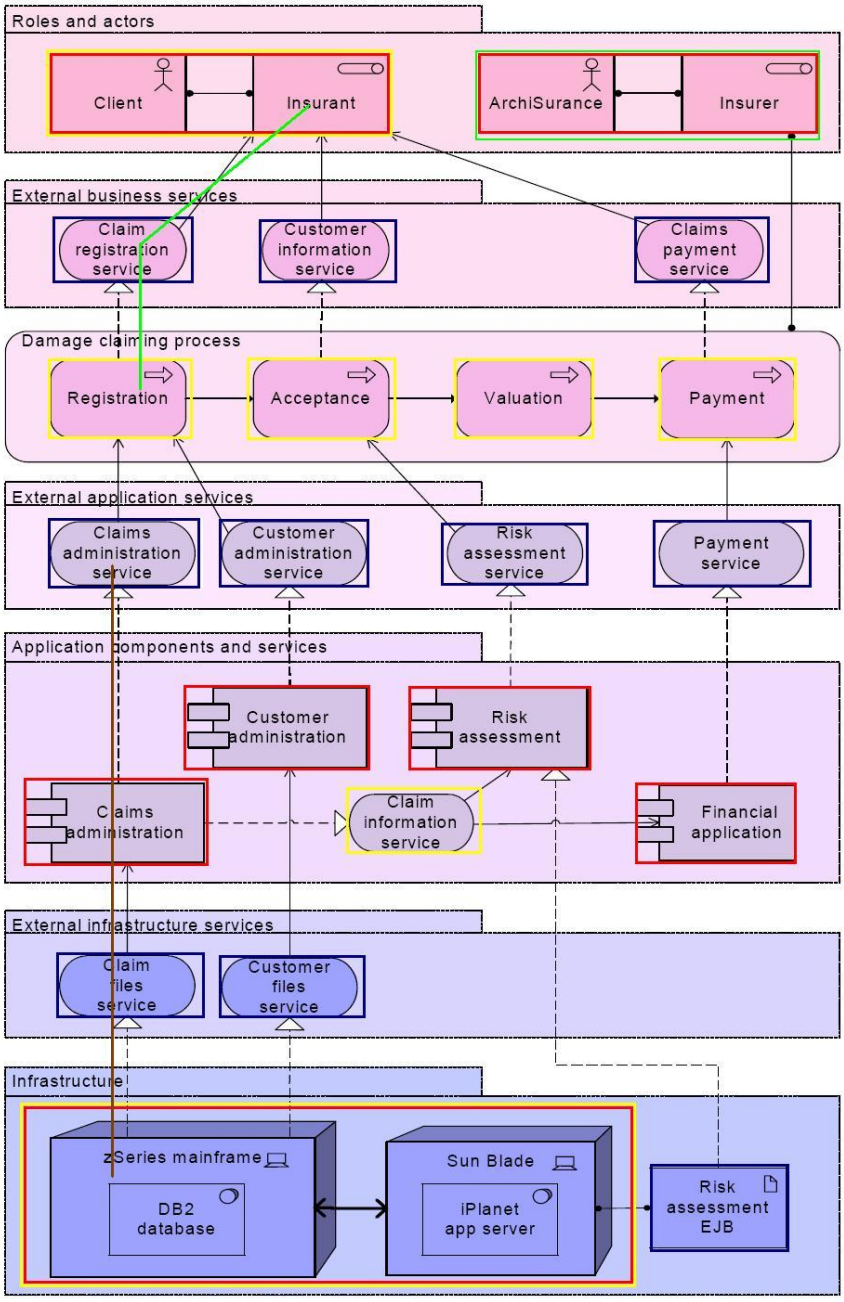


Figure 4

As can be seen in fig 4 we have marked a green route from start to registration, and a brown route to registration through the model. Let

us first take a look at the brown route. One leaves the database towards the claim administration. Going to the claim administration one 'picks up' a still empty set of claim file services. Likely the files that the claim administration works with. (Parts of the database content really). Claims administration is a 'node', one can go to the claims information service here or up towards registration, picking up the claims administration service (administration files) along the way.

Furthermore at the node claims administration= one might have an assignment or be able to transform some 'items' (For example into a filled registration file). In this case, since the original archisurance model creators' knowledge is not available, there will be several assumptions about what the original intent was.

The claims administration registers claims from an insurer. This means the file service 'item' gets filled by information brought in from the claim registration service. Furthermore the process of claim or customer administration is only complete when the payment step is fulfilled (as this is also part of the claim administration, client administration etc).

Thus the actor database would move up towards registration, taking along the items of the claim file service, and then at registration require information from the claim information service. The path to claim information service only opens up when the claims administration has a filled item and is thus still closed.

At this point the first players turn would end. The client actor now has his turn. He can move down only one route. The registration the acceptance or the payment node are possible routes. However to open up the acceptance node the registration node needs to be completed and for the payment one the valuation point. This is based on the architectural interpretation we give the model. As such the client player can merely move down to registration. At the registration node he will encounter the db2 database player and have the choice / option to transform the others item to a filled claims administration service. Once this item is at the registration node the player can immediately move on to acceptance (as that route opens up, if all other requirements are filled). Of course a wise player would also go to pick up the customer information service item. And the customer administration still has to be sated.

With the filled item the brown player, when it is his turn, gets more options as he can now go to the claims information service. From here

on the game can continue (depending on the complexity of the model) for quite awhile until the end condition is fulfilled. This is but a small sample of how the game would go.

From the sample above we immediately saw that the marked model we provided above is still incomplete. Aside from the markings within the model defining each part of the architectures function we also see the architects' knowledge crop up a lot of the times. Why can't the claim information service be entered without sufficient knowledge in the claim administration? Acceptance can only happen after registration has taken place, and continues on with the knowledge from registration? The preparations to explicitate this all can be quite extensive as the implicit knowledge is written down in the game form. Requirements for opening up nodes and possible transformations. Lets see what it would look like when filled in for the given archisurance example. First we will give the options at each node or point of behaviour. Then we will explicitate demands for any routes that exist.

1. Client/Insurant
2. Registration
    a. With claim registration service and claims administration service and customer administration service create filled claim registration service
3. Acceptance
    a. With customer information service and risk assessment service create filled customer information service
4. Valuation
    a. Create cost item
5. Payment
    a. With cost item, payment service and claims payment service create filled claims payment service
6. Insurer
    a. Be end point
7. Claims administration
    a. With filled claim registration service create fulfilled claims administration service
8. Customer administration
    a. With filled claim registration service create fulfilled customer administration service
9. Risk assessment
    a. With filled customer information service and risk assessment service and risk assessment EJB create fulfilled risk assessment service
10. Claim information service
    a. Choose to go to risk assessment or financial applications.

11. Financial administration
    a. With payment service and filled claims payment service create fulfilled payment service.
12. DB2 Database

Route demands (if none stated none exist).

1. Registration -> acceptance requires filled claim registration service
2. Acceptance -> valuation requires filled customer information service
3. Valuation -> payment requires cost item
4. Payment -> insurer requires filled claims payment service && if player = brown -> fulfilled payment service
5. Claims administration -> claims information service requires fulfilled claims administration service
6. DB2 database -> Risk assessment requires claims administration -> claims information service = open

As seen when explicitating all the card 'options' and route demands there is a large deal of information. This should not surprise considering the amount of information caught in the architecture model. The important advantage is that this information is given in a different and easily understood format to the players in the form of cards and direct goals. The abstract layers and width of the model are taken away and instead direct goals and paths remain.

Now that one example has been viewed it is time to look at how the archimate model transformation happens in general. Although the steps seem fairly simple there is still a lot of thought that is carried out in the steps, they are not as simple to execute as they may seem.

Archimate transformation

Now that one example of the transformation has been shown let us look at the formal requirements for the transformations of archimate models to the game.

First every part of the architecture has to be marked (active structure element, passive structure element or behavioural element).

Those active structure elements that have only outgoing lines are start locations. Those active structure elements that have only incoming lines are end locations.

Now the architect shows the knowledge of the dependencies he has. First he should write down what each structure or behavioural element does (if anything explicit) in transformation or creation of items. It is important that some forms of data will be items but need not have been passive structure elements. Looking at the archisurance example the valuation step creates data regarding the value of a certain claim. However *on this layer of abstraction* it does not have any passive structure elements attached to it. This does not mean that no data is created and send on (one step).

After noting down all the transformations that are done at certain nodes it is time to write down the dependencies for routes.

What requirements are there for certain routes to open, what items are needed, what other routes are needed, and what explicit player demands are there. When the information is noted (or ideally filled in on neat pre-set cards that can be layed down) the game can theoretically begin.

Game in practice: evaluation

Next to the archisurance model the game has also been tested on a real life model with a set of non-architect users. The organization modeled in this case was the scouting foundation I am involved in. This allowed certainty about the correctness (and intentional flaws) in the architecture beforehand. Furthermore it made for a real-life but still manageably complex architecture. One of the reasons this was a favoured organization was because of the great amount of communication happening within. This would lead to a model with a great deal of communication lines making it difficult to play through it easily.

Evaluated model

The model is given below (figure 4). Be aware this is a simplified model, especially the depths of the stichtingsbestuur (foundation board) and the groepsbestuur (group directors) were largely abstracted from. This was done to prevent an added layer of complexity in the model that can be viewed largely separate from the main process. They give input, through what (internal) processes they get to this input is for the moment considered unimportant. Also, to start, with the whole 'financial' aspect was left out. This to see if this omission would be noticed by the players. Would they actually validate an incorrect model?

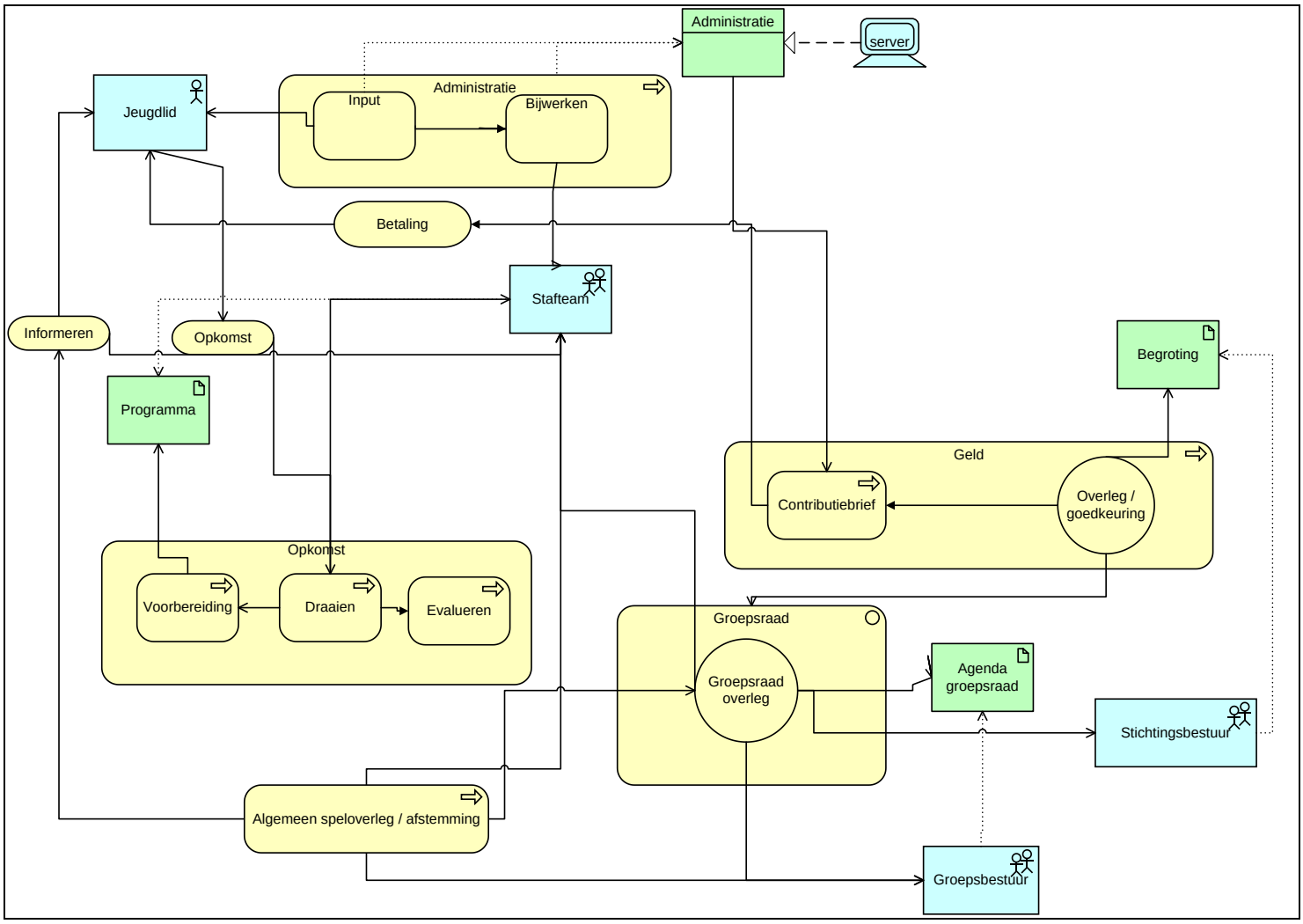


Figure 5 simplified scouting model

The first thing that was an unpleasant wake-up call was the time needed for preparations. To actually make cards to write down all the rules on and display the entire model in a good game format took a good deal of preparation. It did seem however that standardization (fill in blank cards etc) could save a good deal of time here.

What went well?

When the game itself commenced initial results seemed promising. The staff-members participating had a far better understanding / feel with the game then they did with the model and the great amount of communication lines was brought back into understanding by playing through them and noticing the demands between them (who needed to be present, then what it was etc).

Aside from promise there was also one problem sighted, this came in regards to finance. Many staff members do not involve themselves with the finances, by itself a sensible choice. However during the gaming the issue we encountered with this was that some staff members, when faced with the financial part of the model, were unable to provide proper validation or falsification. They just lacked the knowledge to say anything sensible about it. This once more emphasized the importance of players knowing the domain they are playing through. This became especially important as keeping people focused and involved becomes all the harder as they lose touch with part of the game. Furthermore it stalled other players considerably. Specifically the Stichtingsbestuur player was hampered by the stafteam player (who had difficulty seeing the relation between the financial administration and the decision making progress in the groepsraad). This was further enhanced by the initial flaw in this part of the model. Although after a good deal of discussion (and persuasive power on the stichtingsbestuur player) they did decide it was lacking it was difficult for the players to find the exact thing missing.

Points of improvement

So although it was quite easy as an architect to see something was missing, and roughly where, the exact details of what was missing were difficult to find for the players. Obviously in this sample case it was known to us (from the architects' role) what was missing. However it does deserve note that finding the specifics of what was missing were difficult for the players. The rough location in the model where

something is missing and what is missing seemed easy enough. But to actually define what was missing in a model sense was difficult.

The signalation step of invalidation thus seems far easier then actually pinpointing *why* the model is not valid.

Lessons learned

Two lessons can be drawn from this. First it is very important to ensure the knowledge of the players fits the domain they are playing. If none of the players had had any knowledge about the financial aspect then they might have 'missed' the flaw. Thus player selection, one of the initial choices, really is critical. If the wrong players are selected for the wrong positions then serious flaws in the model might be overlooked.

The second lesson drawn from the issue is that for larger models some 'smoothing' function might be useful. A way to go over issues (noting them) so that one player does not stall the game. This of course is in the end a cost-benefit analysis that comes down to the investment in time and manpower made in the architectures validation.

A third important lesson is the 'after game' phase. To maximize understanding of what just happened and what they went through talking to the players after the game is an important tool. Although doubts should be expressed during the game people are apt not to do so, especially in a social context with colleagues present. Talking to players after the game can help ascertain this did not befall and they have no doubts on the model they did not wish to express. Furthermore it helps solidify the idea of the architecture and understanding of their own work processes for most players (or so it was in the example case).

With the game played, how was the understanding of the players of the final model?

Model understanding by the players

Through playing the game the players got a considerably better understanding of the contents of the model. They were far more apt to see the relations when they actually experienced them. Furthermore due to the forced constraints between moves that were explicitated (see game preparations game master) the actual relations fit better with their expectations and experience of it in reality.

At the same time a fair observation had to be made in that no new understanding was gained of parts they previously did not understand. So although a financial part of the model was 'added' by players, those actors that had no understanding of the financial aspect before did not gain a sudden understanding of it.

In other words, the model becomes clearer but knowledge that was not present about the organization structure is not suddenly explained by playing the game. The understanding of parts of the model which they already knew grows. New knowledge that was not already part of the players' knowledge base is not added to it. It could be hoped that the game could also help spread knowledge but this is unfortunately not evident.

The third lesson, talking to players afterwards, mentioned in the previous paragraph does add something to the knowledge base. For this research it was self-evident, with the 7 interviews before and after the game, this would happen. However it also became clear that the game itself does not directly add to the knowledge of the players. If one discusses the game afterwards it is a good stepping stone to start from. The game does open up questions with players. When these are present then answers can often be provided allowing a growth of the knowledge for the players. This however is not inherent in the game but depends both on the curiosity of the players and an extra evaluation step after the game.

Investment

Having looked at both the conceptual transformation as well as a real world implementation it is now time to look at the investment made. The initial investment to get the architecture into a 'game' is considerable. It requires the explicitation of the tacit knowledge and the writing down of what the idea behind all parts of the architecture are. However a good documentation of an architecture should already contain all this information somewhere. If the information is not written down you are, after all, suddenly dependent on the knowledge in the head of the architect (or of the individual employers, and then why have an architectural model?).

The present estimate is that the transformation step takes about the same time as the creation / writing down of the architectural model, assuming the knowledge is present.

Give all the knowledge of an architectural model, on explicitated paper, to an architect and ask him to make the archimate model to get an idea of the investment of making the initial game transformation. This might be further smoothed by automated tagging, but such developments were not a part of this thesis.

The investment of playing the actual game is far more limited. Playing through the model can take anywhere up from 30 minutes depending on the size of the model, the amount of players needed to cover all knowledge fields and the level of abstraction taken up in the model and experience of the players.

The time needed to process the results varies greatly on what is found. If the architectural model is validated this can be seen swiftly and can be noted in several minutes.

If an evaluation step is added this should be expected to take 5 to 15 minutes per player.

The time investment grows rapidly if the architecture proves to be invalid. The game is not fit to say precisely what the flaw is and immediately solve it. As such if flaws are found extra investigation into the precise nature will still have to follow. This may take a considerable extra investment talking to the involved players, looking at their work and where the shoe 'wrings' before the model can be adapted.

It is important to note that the greatest investment lies in initial preparation and in fixing possible flaws in the architecture if any.

Conceptual merit and faults

Now it is time look at the result of the work. There is a game developed based on solid literature and tested in a small real life scenario. So what is the answer to the main question? Is there merit in the concept of a game based approach to validating architectural models?

The first answer that crops up is a yes. There is a workable game that had added value. Like any yes there are some buts however.

The thing that became evident through the development process is that two things are required for a successful architecture game. The first is a manageable, but realistic, investment in the knowledge behind the architecture. Explicitating previously tacit knowledge always requires a time investment (as anyone that has ever documented knows). As such for every architecture model the choice has to be made what the added certainty to the validity is worth and if this outweighs the investment. Personally I believe that often the added chance at validity is well worth the investment, if the proper level of abstraction is taken. This means that yes the game can have clear added economic purpose. But it is not worth validating the entire architecture at all levels in one go. Instead taking those parts where doubts exist about their validity and playing through these (and the directly related parts needed to make them work) is the right investment to make.

The second consideration is a more dangerous one. To make this game work the architect needs a real understanding of his own architecture. This sounds obvious, is this not essential for every architect? However reality has shown that often enough architects make architectural models without a true understanding of what they are modeling. Based largely on what they hear and see without understanding the full reasoning behind it. This is why architectural models often fail, fields like requirements engineering exist, and we even have concepts like domain experts.

For the game to work the architect actually has to understand what he has modeled.

This can be considered both good and bad. An architect understanding the idea behind his model is a desirable option, as is being able to check this. On the other hand, exactly those models where the architect does not have this knowledge require the most checking. Similarly a model where the architect does not understand his own model is not inherently wrong, merely a model all the more in need of validation.

And it is exactly at that border line where the game encounters its first problems. If the architect does not understand his own model it is difficult to play the game properly (although we have done no field tests with such cases).

So the merits of the game lie in the added validity check it gives through a different approach, the ability of the architect to see if he understands his own model and an added layer of documentation.

The faults lie in the fact the game is not designed for, or likely fails, if the architect does not understand the reasoning behind his own architecture. What this means is that exactly those models that are most difficult to validate and have the most need for it still fall short. One can merely see they are flawed models but not see what the exact flaws are.

Conclusion

Looking at all we have done what is the conclusion? Let us look back to the original question;

"To provide a proof of concept for a game based approach to validating archimate models."

When we look at this question the answer would be a clear yes. Yes a game based approach to validating archimate models can work. It does not give 100% certainty, nor can it validate all models. However in many cases it can work and have great added value. In the end the application of such methods as a game based approach is just that, the application of a method. We are not yet at the stage where we have one simple formal paradigm that when followed can always provide a correct and formally validated architecture.

In these methods we developed we are encountering many problems, such as the modeling bottle-neck and the gap between the information specialists and the organizations. Often with the right people and the right approaches these issues are already solved, yet just as often they still fall short. And in those cases the game approach has added value.

It is a different way of working that seems to have its own unique ways of functioning. It has different requirements from most methods, such as the demand on specific people and the explicitation of the tacit knowledge of the architect into a game universe of its own. And exactly this as well as the way it works with this information makes it so worthwhile. The different relationships and application of these opens up new methods and new people who can suddenly understand knowledge they previously could not. Furthermore although the gaming method has its own requirements these do not overlap with the usual modeling and architecture problems. So the breadth of architecture possibilities grows.

The game approach has merit. Apply it at the right times, when necessary and when the knowledge required for it is there. And when that is kept in mind, then reap the benefits.

Literature

[1] Léon de Caluwé, Gert Jan Hofstede, Vincent Peters: Why do Games Work?. Kluwer, Deventer (2008)

[2] R. Hunicke, M. LeBlanc, and R. Zubek. "MDA: A Formal Approach to Game Design and Game Research". In: Proceedings of the AAAI-04 Workshop on Challenges in Game AI, pp. 1–5, 2004.

[3] K. Salen and E. Zimmerman. Rules of Play, Game Design Fundamentals. Cambridge, MA: MIT Press, 2004.

[4] Ilona Wilmont,(2009), A gaming approach to collaborative modelling, Master Thesis in Information Science, Radboud Universiteit Nijmegen, Thesis Number 91 IK

[5] Hevner, A. R., March, S. T., Park, J. & Ram, S.: Design Science in Information Systems Research. In: MIS Quarterly, vol 28, pp. 75-106 (2004)

[6] Herman van der Bij.: Lexicon Architectuurspellen, Landelijk Architectuur Congres 2009. 27 november (2009); in Dutch.

[7] Proper, H., Verrijn–Stuart, A., Hoppenbrouwers, S.: Towards Utility–based Selection of

Architecture–Modelling Concepts. In: Hartmann, S., Stumptner, M., eds.: Proceedings of the

Second Asia–Pacific Conference on Conceptual Modelling (APCCM2005). Volume 42 of

Conferences in Research and Practice in Information Technology Series. Sydney, New

South Wales, Australia, Australian Computer Society 25–36 (2005)

[8]M.Lankhorst, L van der Torre, H ter Doest, A Stam, ArchiMate-methode verbindt Architectuur-domeinen, Informatie, April 2004

[9] Khe Foon Hew, Wing Sum Cheung, Use of three-dimensional (3-D) immersive virtual worlds in K-12 and higher education settings: A review of the research, British Journal of Educational Technology,41 , 1,33-55, 2010, National Institute of Education, Nanyang Technological University in Singapore

[10] S.J.B.A. Hoppenbrouwers, H. Weigand, and E.A.J.A. Rouwette: Setting Rules of Play for Collaborative Modelling. In: P. Rittgen, edt., International Journal of e-Collaboration 5 (4), p37-52. Special Issue on Collaborative Business Information System Development. 2009.. IGI Publishing, USA.

[11] Lankhorst, M., Proper, H., and Jonkers, H. .: The Architecture of the ArchiMate Language.

Enterprise. In: Proceedings of the EMMSAD 2009, held at CAiSE 2009, Amsterdam, the

Netherlands. p367. LNBIP vol 29. Berlin: Springer.

[12] Ordina-allignment game http://www.ordina.nl/Downloadcentrum/~/media/Files/Onze%20dienstverlening/Consulting/The%20Alignment%20Game.ashx?forcedownload=1

[13] Taylor, J.: The "rational" organization reconsidered: An explanation of some of theorganizational implications of self-organizing. In: *Communication Theory*, 11(2), 137–177 (2001)

[14] Drew, P. & Heritage, J.: *Talk at work.* Cambridge University Press. (1992)

[15] Garcia-Lorenzo, L, Nolas, S-M & de Zeeuw, G.: 'Telling stories and the practice of

collaboration'. In: *International Journal of Sociology and Social Policy,* vol. 28, nos. 1–2,

pp. 9-19 (2008)

[16] Proper, H., Verrijn–Stuart, A., Hoppenbrouwers, S.: Towards Utility–based Selection of Architecture–Modelling Concepts. In Hartmann, S., Stumptner, M., eds.: Proceedings of the Second Asia–Pacific Conference on Conceptual Modelling (APCCM2005). Volume 42 of Conferences in Research and Practice in Information Technology Series., Sydney, New South Wales, Australia, Australian Computer Society (2005) 25–36

[17] http://www.opengroup.org/ArchiMate/doc/ts_ArchiMate/

[18]M. Lankhorst (et all), (2004), ArchiMate Language Primer: Introduction to the ArchiMate Language for Enterprise Architecture &

ArchiMate team, Enschede: Telematica Instituut, 2004 Publicatienr.: TI/RS/2004/024 Projectref.: ArchiMate/D1.1.6a

[19] John A. Zachman.: A framework for information systems architecture. In: IBM Systems

Journal, v.26 n.3, p.276-292, (1987)

[20]Eva Brandt , Jörn Messeter, Facilitating collaboration through design games, Proceedings of the eighth conference on Participatory design: Artful integration: interweaving media, materials and practices, July 27-31, 2004, Toronto, Ontario, Canada

[21] ArchiMate. http://ArchiMate.telin.nl Website project ArchiMate

[22]Zagal, J., Rick, J., & Hsi, I. (2006). Collaborative games: Lessons learned from board games. Simulation& Gaming, 37, 24-40.