Object Oriented Modeling
A method for combining model and software development

Wouter van Lelyveld
October 2010

Delft University of Technology
Faculty of Technology, Policy and Management

ABSTRACT

When requirements for a new model cannot be met by available modeling software, new software can be developed for a specific model. Methods for the development of both model and software exist, but a method for combined development has not been found. A compatible way of thinking is required to combine modeling and software development into a single method. This paper will discuss six methods from which the strengths will be combined in the Object Oriented Modeling (OOM) method. After a description of the basis of the OOM method, the possible applications of the model are discussed.

Keywords: *Modeling, model development, software development, methodology, Object Oriented Programming*

1. INTRODUCTION

The economy, transportation and health care are all complex manmade systems with increasingly hard to understand relationships. In order to understand complex systems, models can be used to understand how they work and can be controlled. Using experience from model development, methods have been developed on how models can be created in the correct fashion.

Also, the use of computers and software has become customary over the last decades. This has resulted in exponential increase of computer software for which development methods have been introduced as well, to help create software that meets its requirements.

With the use of computer software, it became possible to model more complex systems, as the software can be used to quickly compute model results and generate visualizations of the outcomes. This helps the user of the model to understand the relationships in the system more easily, and makes it possible to quickly compute the effects of changed variables in the system, creating scenarios.

Modeling software development

Because of this, various modeling software packages have been developed that make it possible to quickly create a working model, like Vissim[1] and Powersim[2] for continuous modeling or Rockwell's Arena[3], for discrete modeling. All these modeling packages have a wide range of possibilities, but it is usually not possible for the modeler to develop the software further to meet specific demands he might have that are not available with standard functions of the software. Also, as the software packages have been developed as a commercial product, users have to purchase these packages in order to use them. Some software does have the option to create runtime versions so users do not require the original software, but these usually have limited options. For development of models that are made publicly available, it may therefore preferable to develop own software.

Next to these arguments, other reasons may exist to develop software to facilitate one specific model as well. When software is developed specifically for one model, methods for both model and software development can be helpful in the development of the application. Software development methods have useful steps that can prevent difficulties in software development, model development methods help to create a useful model. However, for the development

[1] http://www.vissim.com/
[2] http://www.powersim.com/
[3] http://www.arenasimulation.com/

of software designed to facilitate a model the two developments coincide. When the steps in the two development methods are not compatible, it can be hard to use them simultaneously.

That is why it would be useful to have a method that combines both features of the model and software development methods. Because a method that specifically focuses on the development of modeling software has not been found, this paper will introduce such a method, named the Object Oriented Modeling (OOM) method. This method combines the useful development aspects of both existing model and software development methods into one method, and includes steps for both types of development.

First, the difference between a method and methodology is explained, after which some model and software development methods will be discussed. From this a number of useful features are derived that are basis for the OOM method. Finally the steps of the OOM method will be described and its possible uses are discussed.

Method versus methodology

To clearly define the meaning of the word method, the definition is used as given by Brinkkemper [1996]. In the context of information systems development, he defines a method as 'an approach to perform a systems development project, based on a specific way of thinking, consisting of directions and rules, structured in a systematic way in development activities with corresponding development products' [Brinkkemper, 1996]. Despite the different context, this definition is still useful. The article states that methodology is a larger concept which includes a way of thinking. This definition is in line with the different aspects of methodologies that Wijers and Heijes [1990] describe in a framework for modeling methodologies. Wijers and Heijes distinguish a way of thinking, a way of modeling, a way of working, a way of control and a way of support. They also state the way of thinking is often overshadowed by or implicit in the techniques and methods embedded in that methodology, but is of great influence on the ultimate appropriateness of methodologies because important features of the method often depend on the underlying way of thinking [Wijers and Heijes, 1990].

For combining both model and software development methods into one, it is therefore important that the way of thinking that is used for the methods is described as well. As the way of thinking in methods needs to be compatible in order for them to be combined in a new method, a proper description of the way of thinking is required to find common ground between the ways of thinking if these are not the same. This common ground in the way of thinking needs to be basis for the new method in which both methods are integrated.

2. MODEL DEVELOPMENT METHODS

In this paper two different model development methods and four different software development methods will be discussed. As mentioned in the previous section, it is important to note the underlying way of thinking as well. This will be done after discussion of both types of methods to see if a common ground can be found. The two development methods were the only methods found in a brief search for modeling methods. The methods are the model cycle [van Daalen et al, 1999] and the "Co-Evolutionary Method for Modeling Large Scale Socio-Technical System Evolution" [Nigolic, 2009].

The model cycle

The model cycle is an important development method that can be used to create a model and has been created to develop models that can effectively help solve complex problems. There are different versions of the model cycle that have slight differences, but most consist of the following basic steps [van Daalen et al, 1999]:

- Defining goal and function
- Conceptualization
- Model construction
- Model assessment
- Model use/ experimentation

The method suggests that the steps may have to be revisited when the result does not satisfy the requirements. Feedback from the users can then be used as input for a new cycle in the method. This can be visualized as follows:

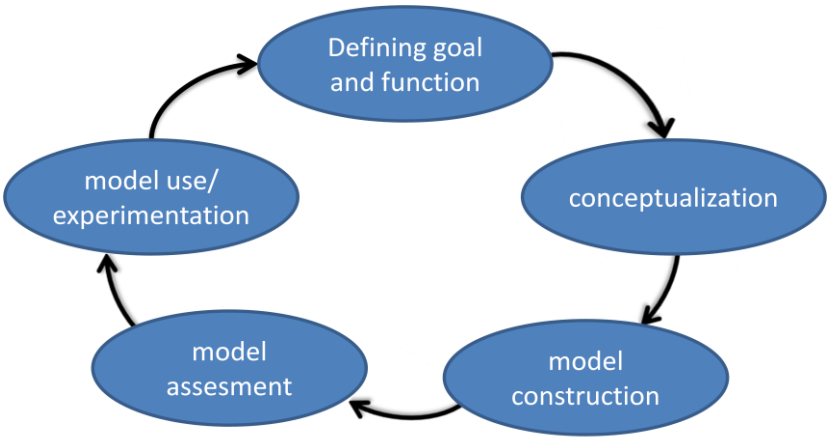

Figure 1: The model cycle [van Daalen et al, 1999]

Co-Evolutionary Method

The second method described is made by Nikolic, who describes a co-evolutionary method for modeling Large Scale Socio-Technical System Evolution, or λ-systems. The method is made specifically for a larger group that collaborates to create a model collectively. Because the goal is to create an evolutionary model with both social and technical aspects, parts of the model are made by people with different specialties. The different steps of the method that Nikolic describes are [Nikolic, 2009]:

- Create the collaboration conditions
- Collect and formalize knowledge
- Collect facts
- Implement the model
- Verify the model implementation
- Analyze the model outcomes
- Validate the model

In the dissertation Nikolic specifically mentions the importance of validation. As evolutionary systems can have unpredictable outcomes, validation of this type of model can be hard. This is especially true if the model includes simulation of social behavior, which evolutionary systems often do [Nikolic, 2009].

Key features model development methods

The model development methods have very similar steps. Both methods basically describe planning, conceptualization, implementation and testing of the model. As the combined method will also be used for model development, these steps will have to be integrated. With this in mind, the software development methods are discussed.

3. SOFTWARE DEVELOPMENT METHODS

Separately from the model development methods, various software development methods have been developed over the years. As computers and software became increasingly more advanced the development methods evolved as well. In discussing the software development methods, it can be noted that the methods do have similarities with the model development methods. The four methods described result from quick research into influential software development methods. Various other methods were found, but these four seemed to have had the most influence in software development. The methods are ordered from the oldest to the youngest, so the evolution over the years can be recognized. The oldest development method with significant influence was the waterfall model, first introduced by Royce in 1970.

The waterfall model

The sequential steps from the waterfall development method originate from the manufacturing industries where changes after the initial design were costly. Therefore the method has sequential steps that are not revisited once they are completed, avoiding expensive redesign. Although different versions exist, the basic different steps in the waterfall model are [Royce, 1970];

- Determine requirements
- Analysis and design
- Implementation
- Verification
- Maintenance

The approach is document driven, which means that it makes it easy for the manager to track the project's progress [Martin, 1999]. Although the waterfall

model is still well known in software development, the strict separation between different phases often makes it an example of how software development should not be managed. This is because requirements and the design are never perfectly described the first time around, and phases must be frequently revisited with new-found perspectives from the implementation phase. Like the basic steps, different variations of the visualization of the waterfall model are known. In figure 2 Royce's original depiction of the method is shown.

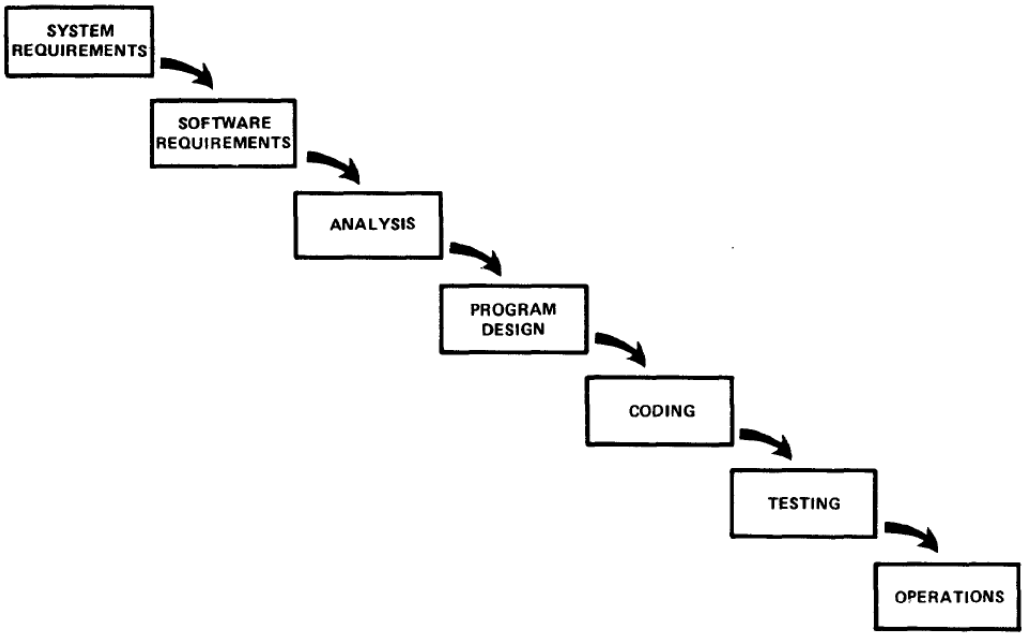

Figure 2: The waterfall model [Royce, 1970]

The V-model

Recognizing the need to look back at the initial requirements, the V-model was developed in Germany in 1986 [Boehm, 1988]. The V-model is similar to the waterfall model, as the basic steps are the same. The difference between the V-model and the waterfall model is that after the implementation step, the following steps look back at the design to test if it fulfills the initial requirements. The process of looking back to the requirements is a first step towards the loop that is created in Boehm's spiral model.

The different steps in the V model are [FHWA, 2005];

- Concept of operations
- Requirements and architecture
- Detailed design
- Implementation
- Integration, test and verification
- System verification and validation
- Operations and maintenance

These different steps can be visualized as follows;

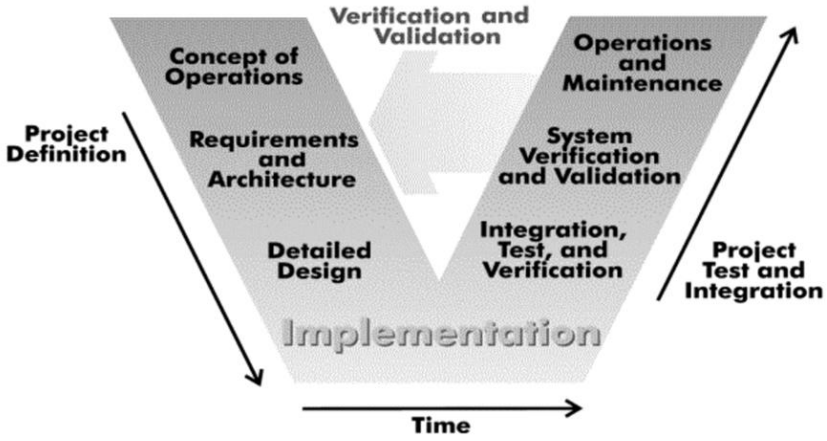

Figure 3: The V-model [FHWA, 2005]

Boehm's Spiral model

Boehm's spiral model [Boehm, 1988] is designed for large software development projects, where large risks are involved if the wrong choices are made in the development of the software. Boehm recognized the risks of development using the waterfall method, and tried to combine the advantages of several development theories into his development method [Boehm, 1988].

One of the most important aspects is that it is risk-driven, as each iteration contains a risk analysis to make sure the correct alternative is chosen. This emphasis on risk management has to do with the magnitude of the projects involved. One iteration was originally suggested to be six months to two years, so an extra iteration due to incorrect risk-analysis could result in very high costs. The different steps of the development method are [Boehm, 1988];

- Determine objectives, alternatives, constraints
- Evaluate alternatives, identify & resolve risks
- Develop, verify next level product
- Plan next phases

In Boehm's original paper, the development process is visualized as follows;

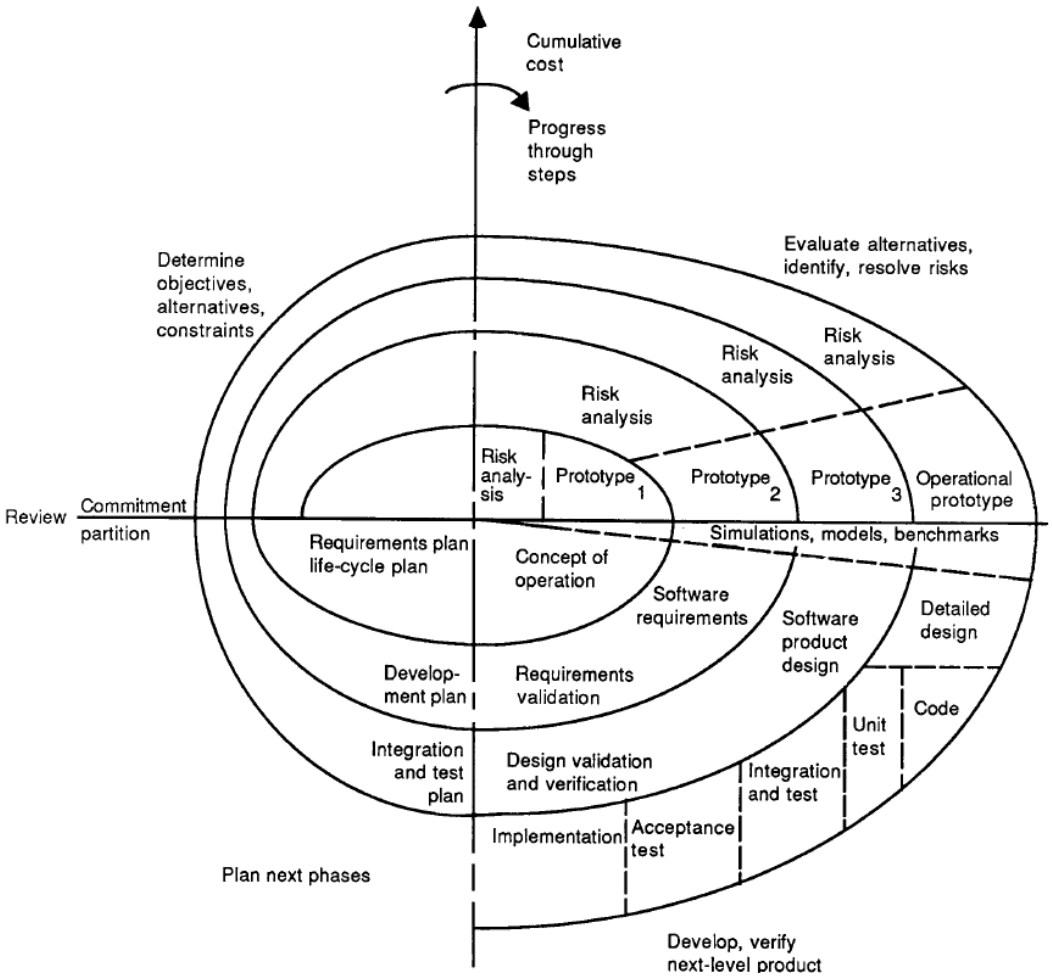

Figure 4: The Spiral model [Boehm, 1988]

Iterative and Incremental Development (IID)

The basic idea of IID is similar to that of Boehm's spiral model; the design cannot be correctly specified directly at the start of the process. IID really is a combination of two concepts that have become accepted in software development over the years; iterative and incremental development. In iterative development preliminary versions of the product are released to get feedback early in the design process. With incremental development separate pieces of the software are developed simultaneously. Although the concept IID is commonly known and accepted, the individual steps cannot been traced back to one specific piece of literature or year. This suggests IID is

more of a concept than a method. Still, for iterative design Hung defined different steps as follows [Hung, 2007]:

- Planning
- Requirements
- Analysis & Design
- Implementation
- Testing
- Evaluation

The different steps of iterative development are visualized by Hung as follows:

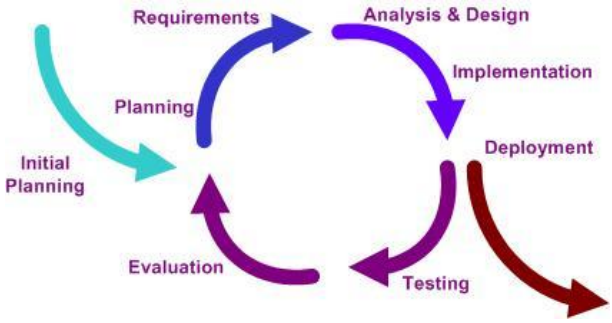

Figure 5: Visualization of iterative development [Hung, 2007]

Key features software development methods
Although the software development methods have changed significantly over the years, some key elements can be recognized in each of the methods. In each of the methods the requirements of the software are determined in one of the first steps, before the analysis and design. Subsequently, the software is developed, and finally tested. In most of the methods, the developed software is compared with the original requirements, which can lead to a new round through the different steps of the method if it does not match the initial requirements or new requirements have been identified. This cyclical aspect is an important feature that was introduced by Boehm, which should be considered in every new software development method.

In a combination of model and software development it would be best to include key features of both model and software development methods. Still, in order for the methods to be combined, a common way of thinking needs to be applied. This will be discussed in the next section.

4. THE WAY OF THINKING

When developing software specifically for one model, the primary way of thinking should be that of the model development, as the final purpose of the application is to provide a useful model. That is why the software has to 'fit' the way of thinking of the model development. Therefore, the way of thinking in the model development will be discussed first. Still, the way of thinking that is applied in model development should keep the way of thinking of the software development in mind, as incompatibility of the two ways of thinking will not result a successful product.

Way of thinking in model development
An underlying principle that can be seen as basis for both modeling methods is that of systems thinking. In systems thinking the system is considered as part of a larger system, rather than as an isolated entity. A systems thinking approach is systemic (holistic instead of in pieces) and/ or rational systematic (stepwise instead of intuitive) [Flood and Carson, 1988].

With a systems thinking approach the problem is always addressed within the context of its environment. Each system has its own characteristics which determines its behavior, but is also dependent on the interaction with other systems. When the relationship between variables in the system can be quantified, a model can simulate the system response to changes in variables in reality.

Table 1: Useful features of the development methods

Development method	Useful features
Model cycle	• Gives specific steps for model development
Co-evolutionary method	• Breaks the project down into small pieces • Creates user feedback early in the process
The Waterfall model	• Delivers easy to manage milestones • Creates a clear distinction between processes
The V model	• Reflects on the initial requirements
Boehm's spiral model	• Fosters development with unclear specifications • Incorporates prototyping
Iterative and Incremental Development	• Provides output early in the process • Able to deal with changing goals and requirements

Way of thinking in software development

The underlying way of thinking in the software development has to be compatible with the way of thinking in the model development. Looking at the description of the way of thinking in the model development, a relevant way of thinking in software development is the object oriented thought process.

This way of thinking is basis to modern software that uses Object Oriented Programming (OOP) [Weisfeld, 2008]. OOP entails that all data is stored in objects, which belong to a certain class. The object class can be used to define the different attributes that the objects have. A classic practical example that is used in OOP is a library; Books, shelves and authors can be defined as classes, where the individual books are seen as objects in the book class [Frishberg, 2001].

The OOP principle is not used in all software, so the choice for this way of thinking limits the number of options for the software. As in this new method the OOP principle is used in modeling, the method is named Object Oriented Modeling (OOM).

5. COMBINING THE METHODS INTO OOM

With the way of thinking aligned, useful features of the methods can be combined into one new method. Table 1 provides an overview the useful features of the discussed development methods.

From this overview, the features of iterative and incremental development will be integrated. Iterations will be kept short, which has the advantage that feedback is acquired fast and possible changes to the design requirements can be processed more quickly. This also breaks the project down into smaller pieces that are easier to manage. By revisiting the process steps frequently, unclear requirements can gradually become clearer, and prototypes can be released early. This is visualized in a spiral, just like in Boehm's spiral model.

Developments steps

Both parts of the model development do still have the development steps. The steps for model iterations in the OOM are based on the steps in the model cycle, as these are best compatible with the iterative character of the spiral. With the steps from the model cycle added, the model iteration of the OOM method can be visualized in figure 6.

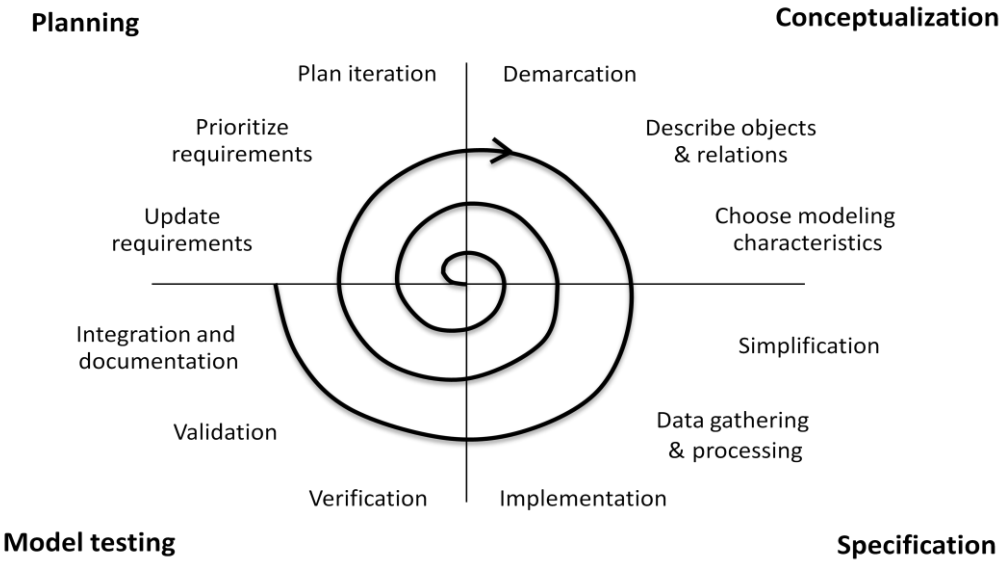

Figure 6: Visualization of a model iteration in the OOM method

The software development steps are not taken directly from one specific method, but are a combination of the different software development methods described.

The planning phase for both development types remains the same, as here the required development type can be determined. Based on the requirements, it can be chosen to further develop the model, the software, or both. Multiple iterations can therefore run simultaneously, making use of incremental development.

For further development of the software, the OOM method is visualized in a similar spiral, but uses steps based on the software development methods. With these steps placed in the spiral, the OOM method for software development is shown in figure 7.

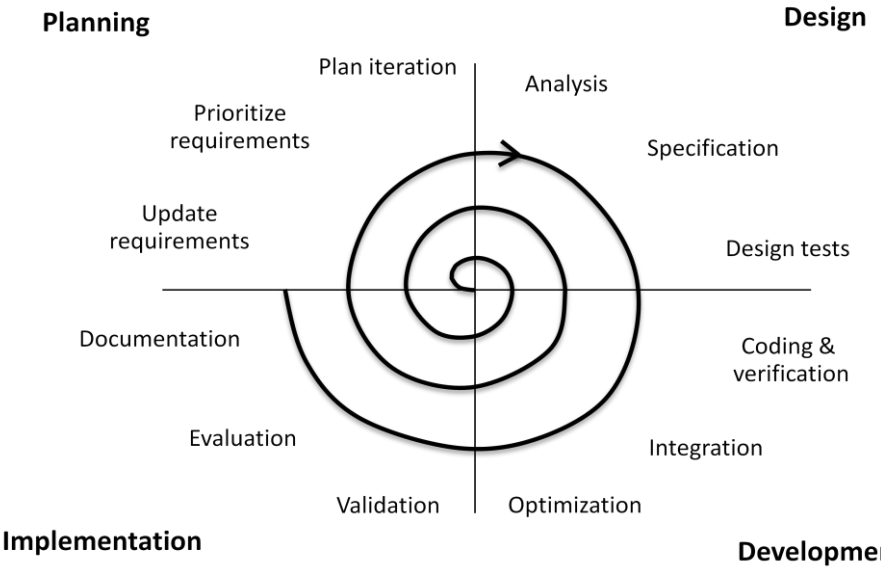

Figure 7: Visualization of a software iteration in the OOM method.

6. APPLICATION OF THE OOM METHOD

As the OOM method is created for the purpose of structuring synchronous development of model and software, the application of the method is limited to just these cases. Furthermore, as a result of aligning the way of thinking in the software development with that of model development, the type of software that is suitable in this method is also limited to software which uses OOP.

As a result, the method can only be applied when a model is being developed in OOP software. In this case however, the method can contribute to a correct development process, which is hard to accomplish using model and software development methods separately.

The usability of OOM method has been validated in the development of the Energy Transition Model[4] (ETM), where it has resulted in a model that satisfied the requirements. Here, the combination of systems thinking from model development with the object oriented thought process from software development has led to a model which is more transparent and more flexible than the original version of the ETM which was built in Microsoft Excel.

[4] http://www.EnergyTransitionModel.com

7. CONCLUSIONS

A new method for the simultaneous development of a model and the software has been developed. The object oriented method is a common way of thinking that is suitable for model and software development methodology. The method comprises a spiral with 4 phases. The first phase (planning) is identical for software and model development. The following 3 phases have a branch for software and a branch for model development that follow a similar way of thinking.

The new method is tailored to models that are developed in Object Oriented software. The method has been applied in the development of an online Energy Transition Model where it has proven valuable in providing a higher level of transparency and flexibility.

8. REFERENCES

Boehm, B.W., 1988, *A Spiral Model of Software Development and Enhancement*, IEEE Computer, Vol. 21, No. 5, 1988, pp. 61-72

Brinkkemper, S., 1996, *Method engineering: engineering of information systems development methods and tools*, Department of Computer Science, University of Twente

van Daalen, C., Thissen, W, and Verbraeck, A, 1999, *Methods for the Modeling and Analysis of Alternatives,* Handbook of Systems engineering

and Management , John Wiley & Sons, New York, page 1037-1076

van Daalen, C., Thissen, W., 2001, *Continue Modellen Deel A – System Dynamics*, reader TB231, TU Delft, TPM

Federal Highway Administration (FHWA), 2005, *Clarus Concept of Operations*, US department of Transportation, Publication No. FHWA-JPO-05-072

Flood, R.L. and E.R. Carson, 1988, *Dealing with Complexity: an Introduction to the Theory and Application of Systems Science*. Plenum Press, New York

Frishberg, R., 2001, *JavaScript Object-Oriented Programming,* http://articles.sitepoint.com/article/oriented-programming-2, last visited August 22, 2010

Hung, T., 2007, *Software development process*, Connexions module: m14619, http://cnx.org/content/m14619/latest/, last visited October 18, 2010

Martin, 1999, *Iterative and Incremental Development (IID)*, C++, http://www.objectmentor.com/publications/IIDII.pdf last visited August 26, 2010

Nikolic, I., 2009, *Co-evolutionary method for modeling large scale socio-technical systems evolution,* Next Generation Infrastructure Foundation, TU Delft

Royce, W., 1970, *Managing the development of large software systems*, Proceedings of the 9th International Conference on Software Engineering 1987, page 328-338

Weisfeld, M., 2008, *The Object-Oriented Thought Process*, Third Edition, Addison-Wesley, USA

Wijers, G., Heijes, H., 1990, *Automated Support of the Modelling Process: A view based on experiments with expert information engineers*, Advanced Information Systems Engineering, Springer Berlin / Heidelberg