# Modeling 2.0?
# State-of-the-art in participatory modeling

Alexander Nossum
alexanno@stud.ntnu.no

May 6, 2008

**Abstract**

Participation is currently on a strong rise, especially with the acceptance of Web2.0. In the field of software engineering participation among developers and stakeholders have been a crucial criteria for success. With ever more complex system and users demanding more flexibility and individualism, there is a lack of supporting and effective methodologies. Technical solutions, especially SOA, have spawned up and are beginning to claim a de-facto standard in the industry. Platform independency, interoperability, context sensitivity and ad-hoc virtual enterprises are some of the requirements put on by stakeholders.

Comprised, this requires a new approach to system development and the general notion on software systems.

In this article we will try to give a high-level overview on an approach that try to address the above mentioned issues. The approach is founded in conceptual modeling and use modeling as a tool for capturing knowledge.

The article is motivated by an assignment in the course: *TDT4252 Modelling of Information Systems - advanced course - Spring 2008* and is the work of students in a learning phase. Thus, it should not be considered to necessarily grasp all aspects of the topic, neither be trusted to be absolutely correct.

# 1 Background

Before we consider new untraditional approaches, it is useful to begin with a short introduction to the traditional as well as historical aspects of modeling in the software industry.

## 1.1 Data modeling

The term modeling in the context of information technology is essentially a way of capturing something in a conceptual, often high-level, abstract figure. Traditionally, modeling has mostly been used to model data or information related to specific software. Techniques such as DFDs[1] and ER-diagrams[2] are the most common "legacy" data modeling techniques found. They are still widely accepted and used frequently.

## 1.2 System modeling

Development of more complex software triggered the spawn of new programming techniques and languages, the most successful currently being the theory of object-orientation (OO). This led to a need to plan, evaluate and possibly simulate software development. To answer this demand there were a great number of languages and techniques developed in the beginning of the nineties. UML [3] related directly to the OO-ideology and is currently *the* most accepted form of modeling OO-systems as well as a number of other branches of modeling. Discussions on the quality and appropriateness of the language ranges from heavy critique to virtuous blessings (Sølvberg and France -). Despite the quite heavy critique, the language seem to be almost a de-facto standard for modeling information systems in the industry and rather often UML seems to be the *only* technique used for *any* kind of modeling.

## 1.3 Organisational modeling

Organizing people with the motivation of collaboration it is necessary to formalize the organizing into what we refer to as an organization. Enterprises are one type of organization which often have economical goals and are thus driven by any sub-goals that fulfil this.

Enterprises are often very large in terms of employees, implemented formal and informal processes, partners, suppliers, customers and technical resources, to

[1]Data-Flow-Diagrams
[2]Entity Relationship diagrams
[3]Unified Modeling Language

mention a few. This leads to a need for techniques that can analyze the high-level situation, make it sensible and possibly simulate different case situations. This is where enterprise modeling plays it's main part. Enterprise modeling is focused on the high-level situation of an organization. Process modeling is in particular used frequently, and techniques such as BPM[4] have captured the interest from large parts of the industry.

That said, enterprise modeling is not only focused on processes. Modeling the structure of the organization, resource modeling and similar are also popular domains that each have their own specialities.

## 1.4 Modeling methodologies

In the previous brief introduction to just some important modeling domains, we have seen that the techniques used and their correspondent domain are very diverse. In order to perform the actual modeling, regardless of technique and domain, one need to have some form of methodology that drives the modeling task. In this section some of the current and traditional commonalities of the modeling task will be discussed.

The modeling task are almost always motivated by something other than the modeling itself, so called modeling goals. These goals varies off course from each task performed, but some commonalities can be found. We can split the overall goals into three main parts:

- Documentation

  Capturing the knowledge for later use. The motivation could be enforced by for example an authority with the primary goal being for example to be certified according to a standardization.

- Sense-making/communication

  Intention being to communicate the knowledge captured so it is easy to understand for other actors. Models are often high-level with fairly low complexity to preserve the pragmatic qualities of the model.

- Simulation/execution

  Model is intended to be comprehensive enough so it can be interpreted by for example a computer and further generate a system which could be executed. The comprehensiveness of the model affects other qualities often related to sense-making abilities. However, the knowledge captured are more detailed and possibly more precise.

[4]Business Process Management/Modeling

### 1.4.1 Abstraction levels

Regardless of the goals of the modeling, the model could be categorized in two main abstraction levels; Type-level and instance level. Essentially this shares the ideology of object-orientation in programming, with their class and instance levels. Type-level is on a generic level and often take into consideration all possible outcomes that could occur in the domain the model captures. Instance level, on the other hand, could be seen upon as an instantiation of a type-level model. The model focus is aimed at what actually happens/happened in the domain "at runtime", and is therefore not as generic as the type-level. Mainly the instance level only captures *one* occurrence of the type-level model.

Choosing the abstraction level for the model is crucial for the end result and to achieve the primary goals. A general trend in modeling is that modeling is an analysts task and requires abstract, generic thinking. Thus, instance level modeling is not considered, or known to be a modeling approach.

Traditionally the modeling task is performed at type-level with the actors being experts at specific topics, such as; modeling experts, domain experts, managerial expert and so on. The group to perform the task is often assembled with heavy use of external actors, as it is considered that, for example, the modeling experts can not be found in the organization performing the modeling. In addition the number of actors are quite few, often around a handful of people, thus enabling the benefits of small groups vs. large groups.

### 1.4.2 Implementation

A result of a modeling task is often a model which is intended to change the domain it models. This often means that routines and employees needs to change, we can say that we implement the model. Implementation of models are common to process modeling and especially in the segment of business process modeling. Further on we will focus on process modeling and experience from this segment. However the experience from process modeling can easily be used directly or adapt to other areas of modeling.

When implementing a process there are some issues that must be considered. In the late nineties a trend in radical change appeared, called BPR[5]. The idea is that for a successful process improvement, the whole process and the involved actors need to change, and change radically. Experience from the BPR era is in general that most BPR projects failed brutally. Off course this was most likely due to many factors. Among these we can find the lack of *participation with the employees* (i.e. only managerial decisions), lack of knowledge on the actual work situation and too high ambitions for a new process. When performing BPR,

[5]Business Process Re-engineering

the goal is to remove the bottlenecks both in the process as well as for the way employees worked and thought. The idea is in it's isolated state good, but in practice it failed and BPR became a shun term. However, experience from BPR is important to consider, if only to avoid doing the same mistakes.

Another aspect to consider is the problem of enforcement in an organization. Often processes, norms and rules are enforced upon the employees without any clear motivational factors. One especial problematic issue is the changing of "legacy thinking", routines that are almost instinctively performed, but not necessarily providing the best way of performing the task. This enforcement results in demotivation and thus often non-successful implementation. The motivational factors plays a crucial role in modeling and especially the implementation part.

Goals can act as important motivational factors, but are often on a level that is hard to easily comprehend for the employees in the domain which is modeled ("the affected stakeholders"). This frequently occurs in the traditional type-level modeling as discussed earlier.

## 2 Motivating participation

As we have seen in the previous section, the traditional approach for modeling revolves mostly around a high-level, abstract, expert-driven approach. This motivates for a new approach. In this section we will concentrate on topics including, what is to be modeled or captured, how this is captured, objectives and resulting benefits obtained. These subjects shares and combine into some common motivating elements which comprise a *motivation for participatory modeling.*

### 2.1 Knowledge

Modeling is often driven by an objective of capturing knowledge in some form. The knowledge to be captured could, in general, be just about anything. It is, nevertheless, important to have an idea of what one wants to actually capture.

In an organization it is important to *externalize* the knowledge held by different actors (*tacit knowledge*). The objective being to disseminate the knowledge and potentially provide continuous learning throughout the organization - *internalization.* Knowledge in an organization is often held closely around the original owner of the knowledge. Such as work experience for the individual workers. The sharing and reuse of such knowledge is often found to be very limited, resulting in overhead costs in "relearning" and "re-experiencing". From a pure knowledge perspective, an ideal situation would be to reuse all knowledge available (implicit and explicit) in the organization. Efforts on this is usually found in almost every organization, with examples such as knowledge repositories, databases, libraries, intranets, internal wikis and often huge amount of text documents, structured and non-structured. Externalization of knowledge is thus usually present in some form in many organizations, although lacks on externalizing tacit knowledge are rather frequent.

Common methods for externalization of knowledge is to write textual documents and make them available in some form. The problem that arise with textual describe knowledge is the ability to easily make use of the knowledge for others - internalization. Texts often describe knowledge in a detailed and thorough form, requiring an extensive effort from the internalizing actor. Thus it is quite common to not make use of the externalized knowledge due to the required effort for internalization.

As indicated, the knowledge is more or less, originally, held by the individual workers. This fact is often underestimated and not taken into consideration when performing a modeling task. It is fairly common to include domain-experts to get domain-knowledge, but the "regular" workers are not necessarily included in the actual modeling. Combined, the fact that most "regular" workers held the most appropriate as-is knowledge and the modeling objective often is to capture this knowledge suggest a clear motivation for a new approach in the modeling methodology.

Although we up until now have concentrated mostly on knowledge as an almost fact-based level, does not exclude the fact that knowledge can be on different levels. Very often a primary goal for modeling is to capture the as-is work situation, such as work/business processes. This is often performed by actors in a leader role which often have an "intended/espoused" perception of the work situation. Thus the model captures an ideal situation and not the actual work situation as it is.

Another aspect of knowledge is the creation of new knowledge, often referred to as innovation. Innovation and creativity feeds from a situation where input is from different and diverse view points. Enabling such situation requires that a heterogeneous group is participating and not homogeneous-minded "experts" which tend to promote group-thinking more often (Strangor 2004). Modeling could easily promote and support creative tasks and is an ideal approach for intuitively and quickly capture ideas and new knowledge. Important for such modeling tasks are especially the modeling actors and the language used, hence leaning heavily on the participation and the selection of involved actors. This will be discussed further in detail later in the article.

## 2.2 Modeling on different abstraction levels

Earlier we have looked briefly on the two major abstraction levels a model can be in, namely type level and instance level. Bearing in mind that the domain experts and regular workers held most of the knowledge which is intended to drive the model, it is natural to discuss the benefits and deficits obtained through the different abstraction levels.

Type level models are of a high abstraction nature. Modeling a generic model requires experience, training and an an analytic approach. These requirements are not common among domain experts and not likely to be present among regular workers. In addition the task of modeling on type level requires a fair amount of effort due to its inherit complexity, regardless of the expertise of the modeler. These issues suggest a closer investigation on instance level modeling.

Modeling on instance level is essentially to capture one specific event that can or has happened. A parallel idea is a normal exemplification of a situation, used for example when one wants to explain a concept to someone. Exhibiting this idea further, we see that exemplification is very intuitive for people when they need to communicate or explain their own knowledge. This phenomenon could be exploited in a modeling task, enabling for easier captation of the knowledge, reducing the requirements for the actors participating in the modeling task.

Although instance level models are able to easily capture events they are usually not very detailed and they do not capture events that could occur on a generic level. This is a lack of instance level models, and should be taken into consideration when choosing abstraction level. However, instance level and type level models are revolving around the same domain and thus are not orthogonal, but not fully parallel either. An effort must be performed in order to be able to extract concepts from instance level models into the generic type level model. Especially interesting is the idea of automating this process. One could imagine capturing several instance level models which could then automatically or semi-automatically produce a type level model through aggregation and reasoning on several instance level models. However this is a fairly complex task which we will keep as an idea in this article and not go into details on.

## 2.3 Activation of models

Traditionally models are used in the design phase of a software system. Usually with primary goals such as analysis and captation of requirements specification and further the initial architecture of the system. Although models used for these tasks are very useful they tend to be outdated or even declared "dead" very fast. Thus limiting the "ROI[6]" for the modeling task. A limited lifespan of a model directly affects the usefulness of the model and can possibly limit the available resources directed towards modeling tasks.

To prevent models from being short lived artifacts in the design phase there are suggested several new techniques. One of them is to *activate* the models.

[6]Return On Investment

Activation of models essentially means that the models and the (deployed) system affects each other throughout their lifespan. There are several approaches for activation of models, among them are; Model Generated Workplaces (MGWP) and Model-configured User-Composed Platforms and Services (MUPS) (Lillehagen and Krogstie 2008). In general they all focus on the idea that the model affects the system, and ideally that the system affects the model. We focus on approaches that incorporates participatory modeling and thus approaches such as Model Driven Architecture (MDA) (OMG 2003) and language-centric approaches like Domain Specific Modeling (DSM) (Frankel 2004) and Business Process Modeling Notation (BPMN/BPEL) (White 2004) are left out of this article.

In a traditional software system users are often given the responsibility to configure the system and thus affecting the systems behaviour. This is traditionally used by setting restricted parameters either through a GUI[7] or through regular files or databases. The trend in software systems, and especially in large ERP[8] systems, is that they are highly configurable. Case studies from implementation (eg. configuring) of large ERP systems indicates that the time and costs involved in such process is very high (≈1 year) (Al-Mashari and Al-Mudimigh 2003). Although these case studies are quite old, they give an indicator on the required effort in essentially configuring and adapting a software system to an enterprise. This gives a clear motivation for a new, less time-consuming, more integrated and less costly approach. Active models may provide this by giving more flexibility to the users by enabling more intuitive methods to *alter*, and not configure the system, but still keeping the overall goals for the system (eg. business rules etc. . . ).

Lillehagen and Krogstie (2008) exhibits, among others (Tinella et al. -), the use of model generated workplaces. The idea is that the user can model their preferred way of working and automatically affecting the system in such way that it reflects the model and the work situation of the user. In essence this is a form of very advanced configuration of the system, but in a completely new fashion using models instead of static parameters. We can think of this kind of modeling as a way for the user to express and capture their domain knowledge into a model and then directly affecting the system and thus activating the domain knowledge for the user. This means that the initial implementation of the system does not need to be a generic adaptation to the whole domain, it only need to support this kind of user-centric active modeling and "building of the system". Benefits from this kind of approach are huge. The greatest benefit is that the end-user can activate his own personal domain knowledge directly in his workplace. Implementation costs for the system can be dramatically reduced since this effort is a more continuous process left (mostly) up to the end-user. Off course still, there has to be a implementation

[7]Graphical User Interface
[8]Enterprise Resource Planning

process to adapt the system "core" to the enterprise, but this is a far less cost than the cost of adapting the system to fit with the enterprise as a whole .

In this section we have looked briefly on the idea of activating models and some concrete approaches that enable this. The area of active models are huge and this article leaves out the greater part of them, although we have tried to shed some light on some approaches that truly fit into the participatory modeling idea.

## 2.4 Benefits of modeling

Every modeling task have one or more goals. They can either be tacit, implicit or explicit. In this section we will highlight some of the characteristic goals in the area of participatory modeling. Although we do not cover in detail all goals of modeling we can in general deduce that participatory modeling can obtain the same goals as for a traditional modeling task.

Participatory modeling is primarily characterized by the involvement of non experts in the sense that "regular" workers are included in the modeling task. This introduce new aspects of goals for the modeling task, primarily social aspects, that traditionally have been looked upon as secondary goals. We will highlight the ownership and increased efficiency obtained from participatory modeling.

Involving end-users in the modeling task gives a very high degree of ownership to the resulting model and the captured concept. Since the end-users are in fact the majority (and possibly the most important) stakeholders for the model it is important to be aware of the obtained ownership of the model. Given a strong ownership to the model naturally provides a higher level of motivation for implementation of the concepts modeled. In the era of BPR[9], low degree of ownership and motivation for change was an important factor for it's "failure". However when an ownership and motivation towards the model is obtained "bottom-up" in the organization the change becomes natural to the end-user because of the participation performed by him. This introduce several benefits when implementing or activating the model, for example using the MGWP approach or even a change in a business process. The motivation and ownership to the model can also increase efficiency in the work process and provide a personalization and individualism to the workplace far greater than a traditional software development project can.

In traditional modeling sense-making is often the primary goal for modeling, especially the fact that models in general are easier to interpret than say 1 dimensional tools such as natural language. Participatory modeling fits perfectly for such goals. A subgoal of sense-making is to capture knowledge, tacit, but existing knowledge or new knowledge created during the modeling task. An innovative centric process it is very useful to have as much divergent domain knowledge as pos-

[9]Business Process Re-engineering

sible. Participatory modeling provides just this, including many “regular” workers, or tacit domain experts in the modeling task. Combined this make participatory modeling ideal for innovating tasks and for externalization tasks.

# 3 Participatory approach

In the previous sections we have shed some light on the traditional modeling approaches as well as introducing some motivation for a new approach, in particular with a participatory emphasis. The discussion has been quite general and no concrete new approaches has been discussed. In this section we will try to introduce two approaches spawn from the idea of participatory modeling. The two methodologies have very different focus and perception of participation and can almost be considered the current two extremes in participatory modeling.

Despite of this, the methodologies share some commonalities which can be considered the foundation of the participatory modeling concept. We will start out with the foundation and specialize further by introducing the two methodologies and their perception of participation.

## 3.1 Common features

The overall driving force in participatory modeling is the involved people. Traditionally these are referred to as *stakeholders*, with an underlaying meaning that they are experts in some form. Participatory modeling extends the concept of stakeholders to involve more representatives from the “actual” real-world stakeholders - more “untraditional” stakeholders. Thus including end-users, regular workers and people across the organisational hierarchy. The rational behind this extension of stakeholders are motivated by the realization of what the goal of modeling in fact is. Goals in modeling consists almost always of capturing knowledge, explicit or tacit domain knowledge, or innovation of new knowledge. Either way, the complete knowledge is *not* held exclusively among so-called domain experts (management etc...) and modeling experts. The actual end-users and regular workers often have the greatest knowledge on what actually is happening in the organisation. In addition they have extensive experience in the instantiated enterprise models, such as business processes, goals, rules, implemented systems, interfaces and so on.

This all argues for involving more differentiated stakeholders in the modeling process. Naturally, when involving more people in a process, participation is required for it to be an efficient process. However, we should bear in mind that the overall goal is *not* to include more people but to *capture more knowledge.*

Another focus area in participatory modeling is activation of models. This is mainly a bi-effect as it is inherited by other upcoming forces such as MDA, BPMN among others (OMG 2003, White 2004, Tinella et al. -, Lillehagen and Krogstie 2008). Although activation of models is important participatory modeling focuses mostly on activating models through the users or through participation and thus driving the system and not primarily on driving the design-phase for the development of a system. Since the foundation of participatory modeling is not revolving around activation of models but more on the modeling task its applications can be wide and thus can support MDA and similar "design-centric" approaches.

## 3.2 Modeling language

Introducing more non-modeling experts in the modeling process provides a lot of issues compared to traditional modeling processes. One especially important factor for the modeling task is the language used to capture the knowledge and hence, comprising the models.

There are few strict rules to follow when choosing the language to model in, but in general non-complex languages are considered better in a participatory setting. As seen in the UML-area, which is one of the most popular languages, still, experts have trouble "navigating the meta-muddle" (Sølvberg and France -), thus arguing that "simpler is better". However this will give trade-offs on other aspects.

One can be pessimistic, or pragmatic, and state that no language will ever be perfect. However, it is important to know what trade-offs one takes when choosing the language. We will briefly introduce SEQUAL, a language quality framework presented by Krogstie and Sølvberg (2003). SEQUAL is modeled in figure 1 and provides comprehensive quality aspects to be considered. In participatory modeling these qualities need to be considered thoroughly before designing or choosing a language. Participant/modeler appropriateness and comprehensibility are probably the most important foundational aspects to consider, however this depends greatly on the task in question, the domain and in general the situation for the modeling process.

An especially interesting quality for a language in this context is the domain appropriateness. In participatory modeling the domain should be defined upfront, but it can also change during the modeling process, for example during an innovative process. The language however should be able to support the chosen domain. Experience from languages that try to fit a general domain (UML etc) often explode in their meta language and becomes very hard to use without special expertise (Sølvberg and France -). In order to limit these issues there are proposed approaches that are more adopting to the domain, namely domain specific languages/modeling (DSL) (Frankel 2004). The approach is considered to

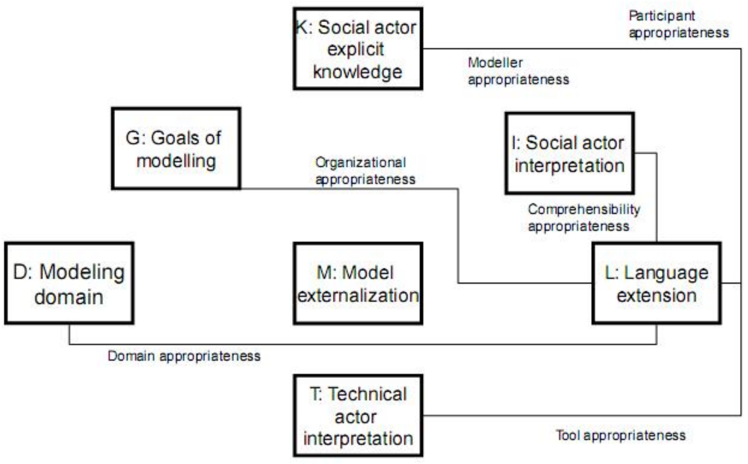
K: Social actor explicit knowledge
Participant appropriateness
Modeller appropriateness
G: Goals of modelling
Organizational appropriateness
I: Social actor interpretation
Comprehensibility appropriateness
D: Modeling domain
M: Model externalization
L: Language extension
Domain appropriateness
T: Technical actor interpretation
Tool appropriateness

Figure 1: SEQUAL Language quality framework (Krogstie and Sølvberg 2003)

be easier for the social actor in terms of comprehensibility and as the role of the modeler. Issues with tool appropriateness is, however, rising. There are not that common for tools to support design on meta level, however there seems to be a trend toward this (Bézivin et al. -, Cook et al. 2007).

## 3.3 The two extremes

As we have mentioned, participatory modeling currently includes two very diverging approaches. In this section we will try to exhibit the two approaches since we found that enlightening the extremes will provide a natural way of enlightening on the field of participatory modeling. The two extremes are a *continuous* approach and a *fixed time* approach. Each have their own focus and perception on participation, especially in terms of time and place dimensions. We will here try to explain the main factors and differences in the two approaches. While we only exhibit the two approaches, there exists off course several other approaches to participatory modeling, but they are more or less combinations of elements from the two we exhibit here.

### 3.3.1 Continuous modeling

This approach to participatory modeling revolves around the end-user of a software system. The ideology leverage the distance between the models and system and, ideally, provide a natural, interactive modeling environment for the user to interact and change the system. In terms of participation we will concentrate on the actors in focus and the time and place dimensions. As the section header reveals, the

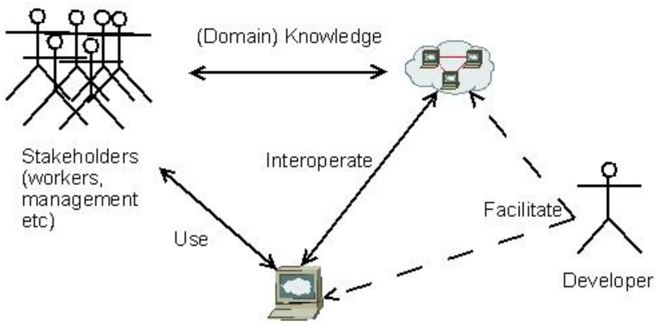

Figure 2: Continuous, interactive modeling, enabled through the user.

approach is performed in a continuous fashion, not fixating time as a limitation. The place dimension is also very fluent concentrating mostly on a "client-based", distributed process. In the approach the actors are primarily the end-users, including regular workers, but not excluding other stakeholders. The important aspect is that the role of the stakeholders are a end-user (ideally) working on a deployed system.

Typically, the ideology can be used to capture instance level models into a knowledge base and then nearing what is coined as model-mining Schimm (2002). This can be very useful as it captures actual work processes, situations and so on from the actual concrete worker performing it. While capturing knowledge from the end users is very useful, the idea is not that new and methodologies exist that support knowledge capturing in enterprise working environments.

However, a new and innovative idea, is to support highly context-sensitive systems, fully (automatically/intelligent) adaptable for the end user, through modeling performed by the end user. This idea is often referred to as Model Generated Workplaces (MGWP) (Lillehagen and Krogstie 2008) or Model Designed User Environments (Tinella et al. -). The general, high-level idea is modeled in figure 2 where we see that the user interacts with both the model and the system and (maybe) most importantly, the system interacts with the model.

The idea is that users have an almost unique perception of what a good system is. Introducing more individual working areas/assignments in enterprises with a trend in more adaptable and ad-hoc virtual enterprises, the need to fulfill a user and an enterprise needs fast and resource-efficient, is increasingly important. With this in mind, a traditional "hard-coded" system will not satisfy the upcoming needs of either enterprises nor end users.

However, an intelligent, context-sensitive system requires a very complex architecture and "backbone". With the re-spawn of component based architectures and

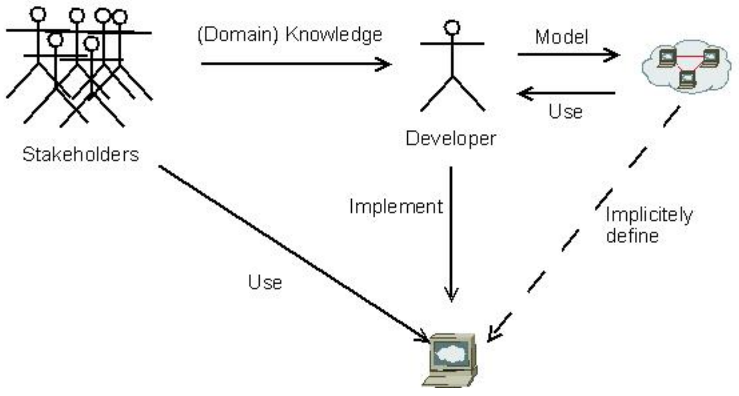

Figure 3: Traditional modeling and development process.

the evolution of the Internet, comprising the SOA[10] idea is a major leap toward enabling such context-sensitive systems. Tinella et al. (-) propose, from prototype studies, high-level architectures and guidelines for enabling this kind of system. We believe the technical complexity of such systems are solvable and will not go into detail on it in this article.

While technical aspects may be solved, there are still other social issues that need to be addressed. As we have discussed earlier, the traditional modeling task is currently not for the “common people”. This is mostly caused by the complexity of languages, generic modeling and a “fear” of abstract, concept modeling. We believe these issues, also are solvable. However, much effort must be put into the user-interface of such a modeling approach. The language is, as discussed very important, limiting the level of learning required for the end user. Enabling instance level modeling, we believe will help leverage the complexity of the task and give much effort to the tool support quality.

This idea of modeling is quite new and has yet to be fully accepted and implemented in enterprises. However the research performed, especially case-studies in enterprises shows very promising results (Carstensen et al. -, Tinella et al. -).

There are several other aspects under the field of continuous modeling in a participatory fashion, however, in this article we have just briefly discussed the main aspects. The overall idea is depicted in figure 2 which can be compared to the traditional modeling and development approach in figure 3

### 3.3.2 Conference modeling

The previous continuous modeling approach concentrated on modeling as a way of interacting, evolving and giving context to a fully operating system. The approach

[10]Service Oriented Architecture

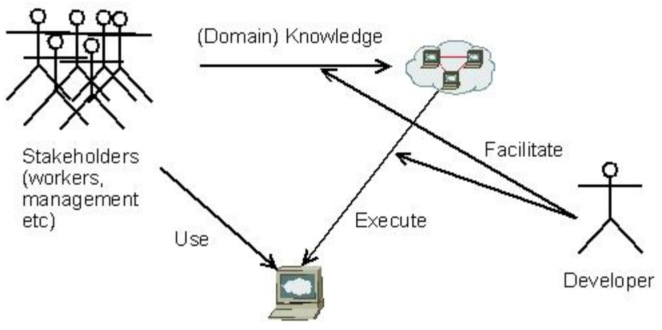

Figure 4: Conference modeling, driven by stakeholders

have several limitations in its application areas, foremost, creativity intensive tasks or capturing and aggregation of tacit knowledge. This approach, often referred to as *conference modeling* (Gjersvik et al. -) is, in a sense, more closely related to the traditional sense of modeling and focus on the pure modeling task. The edge in this approach is the involvement, number and kind of actors involved in the modeling task.

Goals for the modeling are often “social-driven” such as; increase motivation for change, employing a sense of involvement in decisions, increase ownership and so on. Many of the goals are quite similar to those motivating the field of change management and organisational leadership. Thus the focus is somewhat shifted towards a notion of organisational results or influence from the modeling process. The idea is depicted in figure 4, and as we can see the “process” is not as interactive towards the final system as we found in the continuous modeling approach, but still the role of the developer has been shifted towards a facilitator and the stakeholders becomes the drivers. Although the figure shows an execution of the model into a software system, this does not necessarily needs to be performed. Often the model is not activated directly and can just support a change in the organisation.

As mentioned, the conference modeling is suitable for involving a large number of people and perform the modeling in a fixed period of time. The approach is aimed towards collecting knowledge from a large set of actors with, ideally, very different backgrounds and thus incorporating a broad domain perspective. This is very suitable for an innovative task such as design-phases in for example product design, software design, business process design/management and so on.

Conference modeling does not emphasize on activation of models and is thus more aimed at sense-making and communication of knowledge as a primary goal. This gives many benefits towards the aspects concerning required complexity of the language, ad-hoc meta-modeling, tool support (post-it notes vs. software) and similar.

Benefits obtained from involving a large, diversified number of actors are, as mentioned, mostly on the “social” side. Case studies have revealed great results on this methodology, especially in creativity intensive organisations (Gjersvik et al. -, Stirna et al. -). Benefits is especially concerning change management, as ownership of models and the concept they capture lead to increased motivation towards them and as a result the actual people that implements the model is motivated towards it and issues will potentially greatly decrease.

## 4 Concluding remarks

In this article we have tried to shed light on some of the important aspects of participatory modeling. The topic is very new and is still in the research phase. When we started on the article we had little understanding on what participatory modeling consisted of. Therefore some aspects is very briefly discussed or even left out. Nevertheless we feel that the article gives an overview on the essentials of the topic, but could possibly be more elaborate on some parts. However, the scope of the article was not to elaborate in detail every aspects of participatory modeling, but to give an overview of the topic in general. We recognize that elaboration on activation of models and more details on case-studies could be benefitial, however we find that these aspects are described in more elaborate research articles which we have tried to refer to in the text.

We consider the two approaches, discussed in the last section, to be fairly good in giving a high level understanding on the topic, but they require an understanding of why such approaches are useful, eg. the background and motivation.

In general we find that there are still some issues to be solved for participatory modeling to be widely accepted and embraced by the industry. However we believe the era of Web2.0(/3.0), SOA and the general idea of end-user participation and (ad-hoc) composition of services, will rise the need for conceptual modeling in participation and thus will be more accepted both at the user and the managerial level.

## References

Al-Mashari, M. and Al-Mudimigh, A.: 2003, Erp implementation: lessons from a case study, *Information Technology and People* .

Bézivin, J., Hillairet, G., Jouault, F., Kurtev, I. and Piers, W.: -, Bridging the ms/dsl tools and the eclipse modeling framework, - .

Carstensen, A., Holmberg, L., Högberg, P., Johnsen, S. G., Karlsen, D., Lillehagen, F., Sandkuhl, K. and Stirna, J.: -, Integrating requirement and solution modelling: Approach and experiences, *whitepaper* .

Cook, S., Jones, G., Kent, S. and Wills, A. C.: 2007, *Domain Specific Development with Visual Studio DSL Tools*, Addison Wesley.

Frankel, D. S.: 2004, Domain-specific modeling and model driven architecture, *MDA Journal* .

Gjersvik, R., Krogstie, J. and Følstad, A.: -, Participatory development of enterprise process models, - .

Krogstie, J. and Sølvberg, A.: 2003, *Information systems engineering - Conceptual modeling in a quality perspective*, Kompendiumforlaget.

Lillehagen, F. and Krogstie, J.: 2008, *Active Knowledge Modeling of Enterprises*, Springer.

OMG: 2003, Mda guide version 1.0.1, - .

Schimm, G.: 2002, Process miner - a tool for mining process schemes from event-based data, *Lecture Notes in Computer Science* .

Sølvberg, A. and France, R.: -, Navigating the metamuddle, *whitepaper* .

Stirna, J., Persson, A. and Sandkuhl, K.: -, Participative enterprise modeling: Experience and recommandations, - .

Strangor, C.: 2004, *Social Groups in Action and Interaction*, Psychology Press.

Tinella, S., Lillehagen, F., Solheim, H. and Karlsen, D.: -, Model designed user environments, *Whitepaper* .

White, S. A.: 2004, Introduction to bpmn, *BPTrends* .