Collaborative systems modeling and group model building: a useful combination?[1]

E.A.J.A. (Etiënne) Rouwette[1]
S.J.B.A. (Stijn) Hoppenbrouwers[2]

[1]Methodology Department
Faculty of Management
Radboud University Nijmegen
Thomas van Aquinostraat 1.2.31
PO Box 9108
6500 HK Nijmegen
The Netherlands
tel +31 24 3611468, fax +31 24 3611599
e.rouwette@fm.ru.nl

[2]Computer Science Department
Faculty of Science
Radboud University Nijmegen
Heijendaalseweg 135
PO Box 9020
6500 GL Nijmegen
The Netherlands
tel +31 24 3652645, fax +31 24 3653356
stijnh@cs.ru.nl

Abstract

Client involvement in modeling is the hallmark of simulation-based methodologies and applied fields such as information systems development and environmental modeling. Unfortunately, comparison of assumptions and exchange of practical guidelines has failed to take place between methodologies and fields of application. We hope to work towards such an exchange by making an initial comparison between collaborative techniques from information systems development and system dynamics.

Collaborative systems modeling refers to client involvement in information systems development. The field has decades of experience in developing formal models of business

[1] The authors would like to thank Thomas Beck for his comments on a previous version of this paper.

processes and related information structures, and has spawned a range of methods and tools to involve clients in modeling. There is ample evidence concerning the usefulness of alternative approaches. A large part of the literature on group model building covers similar topics. Recent discussions that raised attention in both fields point to further similarities: repeatability of the modeling process (versus dependence on skill of the modeler), quality of modeling and implementation of results.
In this paper we explore whether both approaches to client involvement can learn from each other. We look at differences and commonalities between goals, modeling languages, procedures and methods, and tools and techniques.

Introduction

Stakeholders, experts or decision makers have always been a crucial information source for system dynamics modelers (Forrester, 1961, 1992). Data on central model elements, such as policies that show how information is converted into action, is typically only available in decision makers' mental models. From the 1970s on, the role of client involvement in implementation of results received explicit attention (e.g. Roberts, 1973). Procedures for involving clients were discussed soon thereafter (for an overview of approaches, see Andersen, Vennix, Richardson, & Rouwette, 2007). Group model building emerged as a general term for system dynamics modeling in close cooperation with clients.

In operational research and the systems field similar developments took place, and a suite of approaches developed under the name of Problem Structuring Methods (PSMs). In developing their approaches, practitioners from the system dynamics and PSM fields have frequently compared their approaches, borrowed from each other's techniques or discussed their methodological assumptions (see for example the papers at the 1994 International System Dynamics Conference, the special issue of System Dynamics Review on group model building, Howick, Ackermann, & Andersen, 2006; Lane, 1994). A number of hybrid approaches, integrating elements of system dynamics and particular PSMs, have been described in the literature. Lane and Oliva (1998), for example, describe the theoretical basis for integrating system dynamics and soft systems methodology. Strategic Options Development and Analysis or cognitive mapping (e.g. Eden & Ackermann, 2001) offers tools and techniques that are also used in system dynamics studies. The similarities between system dynamics and operational research approaches goes so far that (in one of its meanings) the term systems thinking is used to cover both types of approaches (Forrester, 1994).

In addition to combining different methods, approaches are sometimes also tailored to specific content areas. An example is Van den Belt's (2004) mediated modeling, which

combines insights from group model building and consensus building on environmental issues. Another example is the application of expert system dynamics to the medical and biological sciences (e.g. Sosnovtseva & Mosekilde, 2006). Here methodological insights are applied to a particular content area, and there is little exchange of methodological principles.

Participative modeling methodologies outside of operational research or the system field have less often been compared to group model building. This is an important gap in the literature, since there is the potential of a fruitful exchange of ideas such as between the operational research and system dynamics fields, but this potential has not been realized so far. The purpose of this paper is to compare group model building and collaborative systems modeling.

Collaborative systems modeling refers to client involvement in information systems development. Comparing group model building to collaborative systems modeling seems particularly useful, since the latter field has decades of experience in developing, most typically, formal models of business processes and the information structures related to them (and a large variety of other, related models). A range of methods and tools to involve clients in modeling is available; the usefulness of alternative approaches is reported in several studies. Part of such alternative directions are several (non system dynamics) causal modeling approaches for collaborative systems modeling, for example Soft Systems Methodology (Checkland & Holwell, 1998; Checkland & Poulter, 2006) and cognitive mapping (Ackermann & Eden, 2005). A more general discussion on the use of causal mapping for information systems development is given by Hodgkinson and Clarkson (2005). Banker and Kaufmann (2004) discuss the use of various simulation approaches, including discrete event simulation, to information systems development. An interesting parallel exists between the tailoring of GMB to a specific content field and, in information systems development, the tailoring of development methods to specific uses, called *situational method engineering* (Ralyté et al., 2007).

Although similar topics are addressed in the group model building literature, there are clear differences between both fields. Whereas group model building aims to develop policies that improve the problematic situation, the intended result of collaborative systems modeling is an integrated information system design that captures essential requirements and structures of the information system at hand. We expect that exchange of ideas between collaborative modeling in system dynamics and in information systems development may enrich both disciplines, similar to the exchange of ideas that is taking place between the GMB and PSM fields. Since other causal modeling approaches are being used for collaborative systems

modeling, an important topic is: what are specific benefits that group model building has in addition to these approaches. Two chief questions thus arise:

- To what extent are the goals and means of GMB and CSM interchangeable or complementary?
- Do these differences in goals lead to fundamental differences in facilitation of client involvement?

In the following we address differences and commonalities between goals, modeling languages, procedures and methods and tools and techniques of both approaches. The aim of our paper is to show whether, and if so, where these approaches can learn from each other.

Goals

In this section we first describe the goals of both group model building and collaborative systems modeling. Next we address commonalities and differences.

Goals of group model building

Group model building can usefully be applied in dynamically complex situations. In these situations one or more indicators that are important to an organization develop in an undesired direction, while the reasons for this development are not directly clear. Typically these problems touch upon the expertise and responsibility of multiple actors and parties in an organization, and each party or actor only has a partial view of the problematic situation. By bringing the parties involved in the problem together and facilitating a joint modeling effort with these parties, group model building is expected to create a more shared view on the problem and on actions available to improve the situation (Andersen et al., 2007; Richardson & Andersen, 1995; Vennix, 1996; Vennix, Akkermans, & Rouwette, 1996). The system dynamics model created in this joint effort aims to explain problematic behavior by capturing the essential structure of the problem.

Thus group model building aims for two sets of outcomes. The first are outcomes related to participants' direct involvement in modeling:

- improved quality of communication,
- mental model change or learning,
- consensus and
- commitment with regard to proposed actions in the problem.

These goals are important not only because they guarantee a high quality input for modeling but more so because each decision maker is thought to have a degree of discretion in

implementing options: commitment of those involved in the problem is instrumental in implementing conclusions of the modeling effort.
The second set of outcomes concerns the technical goals of modeling. The model should be technically correct in the sense that it passes a set of validation tests (e.g. Forrester & Senge, 1980). Simulations with the model should point to high leverage points for steering problematic behavior in the right direction, and these high leverage points should logically connect to proposed options in the real world.

Goals of collaborative systems modeling

In collaborative information systems modeling, business-IT alignment and human-IT alignment are central goals. The aim of the modeling effort is to develop an IT system that supports the essential business processes as understood by those involved. Models are crucial artifacts in the development of IT systems. In fact, prominent branches in contemporary systems development are *Model Based Architecture* (OMG, 2003) and Model Driven Systems Development (Stahl et al, 2006). In software development, perhaps the most famous example of a (combined) modeling approach and language are the Rational Unified Process (Kruchten, 2000) and the Unified Modeling Language (Booch et al., 1998). However, a number of such approaches/languages exist, some of which are focused less on (technical) software development and more on information/business/enterprise modeling, for example Business Process Modeling Notation (BPMN, a schema language; OMG, 2006) and Semantics of Business Vocabulary and Rules (SBVR, a standard structure for expressing business concepts and rules; OMG, 2005). In addition, the creation of various flavors of *ontologies* (put simply, conceptual networks aimed at sharing concept definitions) is a strongly related practice (Guarino, 1998).

Whereas in work processes, business concepts are used, an IT systems design and implementation (also) need technical concepts. Specialized classes of model are used to address specific needs and audiences. In principle, mappings between, say, business-oriented models and technically oriented models should facilitate the alignment of the "business world" and the "technical world" (Hoppenbrouwers, 2008). A technically correct model should essentially be correct in formal syntactic sense, and be complete. Social goals boil down to agreement by stakeholders (validity in the opinion of both individuals and the group):

- understanding (agreement on interpretation of models by stakeholders),
- consent (agreement on the accurate and appropriate description of the domain by the model) and
- commitment (agreement on actual implementation and deployment of systems based on the models).

However, in practice the bridging of the gap between socially embedded processes of conceptual modeling and rational processes of system engineering has proven extremely hard and to some extent is still unsuccessful.

Capturing business processes in a model requires both formal and informal language and communication. The model helps in eliciting the explicit and implicit knowledge used in operating and communicating about the business processes. Increasingly, modelers focus on capturing business rules: "organizational rules under jurisdiction of the business" (OMG, 2005; Ross, 2003). Business rules guide or automate decision makers' behavior in a particular domain. An example of a business rule in banking is the following: withhold authorization of an increase in long-term credits if a company's equity is below 20%. Similar business rules are specified to capture, for example, legal expertise, medical protocols, safety regulations, and tax regulations.

Even in domains characterized by structured knowledge such as law, business rules are difficult to capture in a formal format (while from an implementation point of view, formalization is an absolute requirement). Translating the complexity of business processes and human communication into concepts and relations boils down to interpreting ideas from one world so they fit the more structured language/text of another world. This brings differences of values and interpretations of stakeholders to the fore, introducing into systems development elements of a negotiation (Rittgen, 2007). The challenge in reconciling differences within and between both worlds consists of creating a formal model that is acceptable and useful as well as based on business and human concepts. Thus, IT system development increasingly aims to reconcile technical and social goals.

Development of an IT system is often the responsibility of computer engineers. Accordingly, techniques and methods for building these models typically have a technological background and are adapted for use in business environments. Clarifying work processes and the way decisions are made (e.g. trough business rules) takes the form of communication about formal models. Eliciting and structuring concepts and validating formal models are thus central activities in IT systems development. Clearly, since information systems are often developed by teams of people, communication has always played a role in IT systems development. However, communication in teams of software developers is facilitated by their shared (technical) background and regular use of formal models to capture concepts and relations. Communication between IT system developers and non-technical stakeholders (e.g. users, commissioners, managers) typically involves clarifying informal communication and decision premises. As a result, techniques and methods for involving users in IT systems development typically do *not* involve formal models. Examples are Joint Application

Development (Wood and Silver, 1995) and scenario-based approaches (Carroll, 1995). Recently, advanced appliers of technology in business have become aware of concrete advantages of applying formal techniques in business governance and decision making. Formal representations of business rules, Business Process Management (BPM), and business intelligence are examples of this approach. However, the burden of formalization is usually too great for practical application, and requires the involvement of highly specialized, highly expensive experts.

In conclusion, collaborative systems modeling sets both technical and social goals, though the latter are underemphasized. Technical goals boil down to rationally capturing knowledge of people in the business, so that the business is accurately represented in a formal model. Based on these formal models, computational tools and techniques (e.g. information systems, business analysis and simulation) can be developed to support the business.

Comparison of goals

Group model building and collaborative systems modeling both have technical as well as social goals. Both approaches attempt to construct informal as well as formal models that are valid representations of the problem/domain under consideration. In group model building the subject is typically a dynamically complex problem. The intended use of the model is then primarily to identify high leverage points to alleviate the problem. Social goals primarily concern stakeholders' consensus and commitment to the conclusions of the modeling project.

The primary difference between both approaches is that SD tries to find an explanation why the current system behaves the way it does (Beck, 2008). Most of the effort is spent on building a formal model which represents the current system best and therefore reproduces the (unwelcome) behavior. Once you have a simulation model it is considered relatively easy to find the policies which will make everything better. This is usually where the SD intervention ends. IT systems development (collaborative or not) spends less attention to this first analysis step. Sometimes the process starts with describing the current situation (current business processes, data flow, functions etc.) after which the business analyst stares for hours at these AS-IS descriptions and determines what these business processes, data flows and functions have to look like in the future (TO-BE). In this phase the business analysts might involve the stakeholders, end-users or subject matter experts and collaboratively propose a TO-BE solution. However, these TO-BE processes are never dynamically checked to prove that they will indeed perform any better than the current AS-IS processes. It is a static comparison and arguing based on experience, gut-feeling and also of politics and selling. The purpose of the IT modeling in a first phase is then to make sure all the business people and IT people have the same understanding of the TO-BE solution so that the sponsor can sign it

off and system developers have a clearly stated mandate, describing what is in and out of the scope of the system to be. It is not really about proving that TO BE will be better than AS IS (Beck, 2008).

A secondary difference concerns the timeframe. Collaborative systems modeling is not so much focused on alleviating a particular problem which is bounded in time. Instead the approach aims to capture the essential elements and structures that form the basis for decisions on the design of an information system. These design decisions concern a longer time frame. Although in principle nothing prevents the use of system dynamics models over a longer time frame, the approach is more geared to capturing the properties of a specific problem, or class of problems, with regard to a predefined time horizon.

Modeling languages

Modeling language in group model building

As described in the previous section, group model building and system dynamics aim to capture the structure of complex dynamic problems. In system dynamics, a set of interacting feedback loops forms the most important part of the model. As an example, imagine a government agency that experiences increasing delays in its service to clients. Representatives of all departments that have a role in the work processes involved will be invited to participate in group model building sessions. Typically the model will show how the work process in each separate department may be rational with regard to local goals, but create unexpected and undesired consequences for other departments. The structural explanation for the increase in service delays would need to show how each department reacts to the other and forms a part of a reinforcing feedback loop.

Problematic behavior is said to arise in particular from feedback loops contained in the problem structure. An understanding of self-reinforcing and balancing loops is necessary to explain behavior over time. For creating this structural understanding, qualitative or quantitative models may be used. The hallmark of system dynamics are quantitative models consisting of a set of differential equations. These models are visually depicted in the form of so-called stock&flows models. Qualitative models are depicted in the form of (non-formalized) stock&flows models and causal loop diagrams. Recently a type of models has been proposed that offers a middle ground between conceptual and fully formalized models. These so-called Marvel models add three types of information to causal loop diagrams: current values of variables and strength and speed of relations (Van Zijderveld, 2007). Marvel models then allow the modeler to see the effect of a change in a parameter value on behavior patterns. System dynamics models of any kind typically consist of many variables and relations. In the

system dynamics community several authors have addressed the benefits of qualitative versus quantitative models and causal loop diagrams versus stock&flows diagrams (Coyle, 2000, 2001; Homer & Oliva, 2001; Warren, 2004). Rouwette, Vennix and Van Mullekom (2002: 14) find that most system dynamics models are between 20-200 variables (with a minimum of 20-30 and a maximum of 200-1000 equations). In larger models part of the structure is often repeated, for example when three types of products in inventory are distinguished. However, even in large models feedback complexity is preferred over detail complexity. After dynamic behavior is explained in terms of interacting feedback loops, the model is used to identify policy interventions in the problem. The ultimate goal of group model building is to improve future behavior.

Modeling language in collaborative systems modeling

Whereas in system dynamics formal models are largely similar (based on differential equations), in collaborative systems modeling a vast number of academic and industrial approaches co-exist (both formal and informal, the formal ones based mostly on discrete mathematics), either peaceful or not. This has in fact led to what is sometimes referred to as the "YAMA syndrome" (Yet Another Modeling Approach). The field is divided in many specialized sub-disciplines that focus on different aspects of systems. An integrated view is thus difficult and does not seem to have high priority in the field. The ultimate goal of IT systems modeling is to construct a system that proves to be effective. Proof in this sense typically means either formal mathematical proof (for example, showing that some formal specification is realized by means of a formalized computational machine) or trial-and-error testing. For complex systems, in particular those involving or even including businesses and/or human beings, provable quality is very difficult to achieve. For instance, a language such as UML (Booch et al, 1998) is relatively generic and widely accepted, but not formal (i.e. cannot, in its standard version, be mapped 1:1 to an appropriate, standard formalism) and thus cannot be a common basis for automated model checking or software generation. Apparently IT systems modeling may focus on highly domain-specific mathematical quality, or generic, informal integrated complexity, but not on both.

Comparison of modeling languages

With regard to modeling languages, there is a striking difference between group model building and collaborative systems modeling. While the first field a consensus seems to exist on the preferred modeling language (causal loops, stock-flows; formalized as differential equations), in the second many different approaches are used in parallel throughout industry (though mostly based on discrete mathematics). In fact, considerable utilitarian overlap exists between many of the methods and languages used.

Procedures and methods

Procedures and methods in group model building

System dynamics modeling consists of series of phases that each may be supported in different ways. We first describe the phases of system dynamics modeling in a general sense and in the next section focus on procedures to facilitate client participation in modeling. A general outline of the phases in system dynamics modeling is the following (Richardson & Pugh, 1981):

1. identification of the problem and model purpose;
2. system conceptualization;
3. formalization and parameter estimation;
4. analysis of model behavior: sensitivity analysis and testing;
5. estimation of model validity or evaluation;
6. policy analysis;
7. model use or implementation.

In the first phase a preliminary problem definition is chosen, in which the problem boundaries, time horizon and the reference mode of behavior are identified. In the following phase, other concepts central to the problem are identified and typically captured in visual form. In this way the model structure grows as new variables and relationships are added. In the formalization phase each relationship is translated into a mathematical equation. The resulting set of equations allows the model to be simulated over time. Model behavior is then analyzed to understand model behavior and the influence of structure on behavioral patterns. Testing includes changing initial parameter values or changing relationships between variables, and observing the effects on model behavior (e.g. Ford, 1999). The phase of testing the model for its validity is crucial to the modeling process and widely discussed in the literature (see e.g. Forrester & Senge, 1980). Model validity concerns the adequacy of the model for representing the problem under study. Forrester and Senge (1980) refer to validation as the process of building confidence in a model. For this they identify a large number of structural and behavioral tests. Confidence in a model increases as more tests are successfully passed. In this phase a balance needs to be struck between adding more detail to the model structure and therefore increasing its complexity, and the ability to understand a model. In the policy analysis phase, parameters or larger sections of model structure are changed in order to see their impact on system performance. The goal is to identify changes that steer outcome variables in the preferred direction. In this phase a scenario analysis can be performed by running the model under different conditions for exogenous variables, which clarifies the robustness of policy interventions.

Procedures and methods in collaborative systems modeling

In IT systems modeling a general outline of phases is the following (many slightly differing versions of this list exist; we give our own version. Please note that the phases rarely follow each other up linearly, nowadays: the process is "iterative"):

1. problem definition: why should we develop this information system?
2. requirements analysis: what do we want from the information system?
3. functional design: what *exactly* will the system do?
4. technical design: what will the structure, the architecture of the system look like; what existing components are to be built in (if any)?
5. realization: actual construction (programming, generation) takes place
6. testing: both at a technical level and at a usage level
7. deployment: the actual introduction of the working system in the organization.

The first two phases typically involve the most intensive interaction with users and stakeholders, though various other phases also require such interaction. Participative specification of requirements generally puts high demands on time and money. However, in situations where requirements analysis is not completed satisfactorily, IT projects typically fail to deliver the benefits users and developers hoped form (Standish Group, 1999).

A common critique of participative requirements analysis is that is takes a long time and success is not guaranteed. Also, *evolution* of information systems (rapid development and change driven by changes in the business) is an increasing problem. So-called agile development of software (for example, using the SCRUM approach; Schwaber and Beedle, 2002) is thought to be helpful in addressing this problem, but does not readily extend to the requirements phase.

Comparison of procedures and methods

In conclusion, phases of model construction show similarities. In both approaches the client is most intensely involved in the first phases. A phase where elements of modeling might help IT system development is in testing (Beck, 2008). System developers frequently use prototyping, which can be considered a form of simulation model. Hence, there is not so much a need for formal simulation models during the implementation phase you just show the end users what you have already built and he or she will like it or ask you to change it. "Testing" is sort of running simulations as well. Software code is developed in a development environment. Once you have a piece of code, the developer conducts a functional unit test where he/she tests the functionality against the design specification (does it work as designed)? Later on system developers conduct functional integration tests where all software pieces are tested together. Then the software code is moved to a training environment which should be much closer to reality (the production environment). Some potential end-users

then test (or simulate) the application in a user acceptance test. By using simulation, criteria such as user-friendliness and touch-and-feel can be tested as well.

Tools and techniques

Tools and techniques in group model building

In many of the modeling phases described above, information contained in stakeholders' mental models is crucial. In group model building a variety of procedures is available that facilitate client involvement and help to elicit and test mental models (see for example Vennix, 1996, 1999). As a foundation for choosing between different procedures, Andersen and Richardson (1997) develop a set of guiding principles and so-called scripts for group model building sessions. Guiding principles capture basic ideas in the interaction with clients, such as break task/ group structure several times each day, clarify group products, maintain visual consistency and avoid talking heads. Scripts are more concrete instances of these principles and refer to small elements of the interaction process (Andersen & Richardson, 1997; Luna-Reyes et al., 2006). The following table shows scripts described in Andersen and Richardson's (1997) original paper.

Phase in modelling	Script
Defining a problem	Presenting reference modes Eliciting reference modes Audience, purpose, and policy options
Conceptualizing model structure	Sectors, a top down approach Maintain sector overview while working within a sector Stocks and flows, by sector Name that variable or sector
Eliciting feedback structure	Direct feedback loop elicitation Capacity utilization script System archetype templates "Black box" means-ends script
Equation writing and parameterization	Data estimation script Model refinement script "Parking lot" for unclear terms
Policy development	Eliciting mental model-based policy stories Create a matrix that links policy levers to key system flows "Complete the graph" policy script Modeller/ reflector feedback about policy implications Formal policy evaluation using multi-attribute utility models Scripts for "ending with a bang"

Table 1. Group model building scripts (cf. Andersen and Richardson, 1997)

The method developed by Hines (Otto & Struben, 2004) integrates the phases of modeling with available scripts and techniques for client involvement.

Tools and techniques in collaborative systems modeling

In line with the great number of existing languages and methods in IT systems modeling, the number of tools and techniques in the field is also vast; too vast to even attempt listing them. However, truly *collaborative* modeling methods are seriously underrepresented (which does not mean there is no collaboration going on). Collaboration support is very often reduced to sharing documents and organizing superficial group reviews *in hindsight*, leaving detailed design and implementation decisions to technical experts.

Comparison of tools and techniques

In group model building a development can be seen from an overall methodology to a set of more detailed methods and techniques from which a facilitator can choose when planning an intervention. Collaborative modeling methods in information systems development are less frequently described, and typically models are presented to and checked by stakeholders after they are developed.

Conclusion and discussion

In this paper we sought to compare collaborative systems modeling and group model building with regard to two questions: 1. To what extent are the goals and means of both fields interchangeable or complementary? 2. Do these differences in goals lead to fundamental differences in facilitation of client involvement?

With regard to the first question, there seem to be more similarities than differences. Group model building and collaborative systems modeling both have technical as well as social goals. An important difference is that group model building and SD enable testing of the desired situation, whereas information analysts put less emphasis on the step from current to desired situation. Simulation of the desired situation might make it easier to test whether the TO BE situation indeed performs as expected (Beck, 2008). Group model building is typically used to provide answers to a particular problem, while models used in information systems design are expected to describe situations lasting on a longer time frame.

Although goals of both approaches seem similar, modeling language and tools and techniques are clearly different:

- in group model building there is one preferred modeling language, while in collaborative systems modeling many different approaches are used;
- in group model building detailed scripts for involving clients in particular phases of the intervention are developed, while in collaborative systems modeling client participation often comes down to critiquing previously developed models.

With regard to procedures and methods both approaches show similarities. The phases of model construction are alike and in both fields the client is most intensely involved in the first phases.

So although goals are largely similar, the field of collaborative systems modeling seems to be more fragmented and less accustomed to joint development of models. SD modeling might add benefits in the phase of testing (Beck, 2008). It seems that modeling scripts developed in system dynamics could also be applied in structuring information systems development. In the future we intend to study this contribution by applying scripts in information system development cases. Group model building, in turn, can benefit from the micro level studies of communication processes in negotiating about models (Rittgen, 2007). An issue that deserves further attention is use of formal simulation for testing models for information system design.

An issue we addressed in the introduction of this paper concerns the benefits group model building has to offer in relation to the application of causal modeling approaches to information systems development. An area where group model building seems to make a unique contribution to the information systems field is in its application to Enterprise Resource Systems (ERP) implementation. A master thesis study by Venderbosch (2007) focuses on an ERP system at ONEgas in the Netherlands, in particular on the optimization of the corrective maintenance process. Using five group model building sessions and data from the ERP system, the researchers show how maintenance may be improved. In this case the modeling sessions provided the platform for clients to develop a clear understanding of the structure behind their problem, which formed a basis for interpreting data from the ERP system. An extensive evaluation of the process shows that the clients' insight into their work process improved and commitment to implementing recommendations is high. Implementation of ERP, often combining a generous availability of quantitative data with a lack of understanding of the core structure behind data, presents a clear opportunity for the use of group model building. For this range of problems, group model building's base in system dynamics and its ability to integrate qualitative and quantitative data offer an advantage to conceptual modeling approaches such as SSM and cognitive mapping (see also Killingsworth, Chavez, & Martin, 2008).

References

Ackermann, F., & Eden, C. (2005). Using causal mapping to support Information Systems development: some considerations. In V. Narayanan & D. Armstrong (Eds.), *Causal mapping for research in Information Technology* (pp. 263-283). Hershey, PA: Idea Group Publishing.

Andersen, D., & Richardson, G. (1997). Scripts for group model building. *System Dynamics Review, 13*, 107-129.

Andersen, D., Vennix, J., Richardson, G., & Rouwette, E. (2007). Group model building: problem structuring, policy simulation and decision support. *Journal of the Operational Research Society, 58*(5), 691-694.

Banker, R., & Kaufmann, R. (2004). The evolution of research on information systems. A fiftieth year survey of the literature in Management Science. *Management Science, 50*(3), 281-298.

Beck, T. (2008). SD and Information System Development. Personal communication.

G. Booch, J. Rumbaugh, and I. Jacobson (1998). The Unified Modelling Language User Guide. Addison Wesley, 1998.

Carroll, John M. edt. (1995): Scenario-Based Design: Envisioning Work and Technology in System Development. Chichester: Wiley.

Checkland, P., & Holwell, S. (1998). *Information, systems and information systems*. Chichester: Wiley.

Checkland, P., & Poulter, J. (2006). *Learning for action. A short definitive account of Soft Systems Methodology and its use for practitioners, teachers and students*. Chichester: Wiley.

Coyle, G. (2000). Qualitative and quantitative modelling in system dynamics: some research questions. *System Dynamics Review, 16*(3), 225-244.

Coyle, G. (2001). Maps and models in system dynamics: Rejoinder to Homer and Oliva. *System Dynamics Review, 17*(4), 357-363

Eden, C., & Ackermann, F. (2001). SODA – The principles. In J. Rosenhead & J. Mingers (Eds.), *Rational analysis for a problematic world revisited. Problem structuring methods for complexity, uncertainty and conflict* (pp. 21-42). Chichester: Wiley.

Ford, D. (1999). A behavioral approach to feedback loop dominance analysis. *System Dynamics Review, 15*(1), 3-36.

Forrester, J. (1961). *Industrial dynamics*: Pegasus Communications.

Forrester, J. (1992). Policies, decisions, and information sources for modeling. *European Journal of Operational Research, 59*, 42-63.

Forrester, J. (1994). System dynamics, systems thinking, and soft OR. *System Dynamics Review, 10*(2-3), 245-256.

Forrester, J., & Senge, P. (1980). Tests for building confidence in system dynamics models. *TIMS Studies in the Management Sciences, 14*, 209-228.

N. Guarino (1998): Formal Ontology and Information Systems. In: Proceedings of FOIS`98, Trento, Italy, EU, Edited by: N. Guarino. Pages: 3-15. Amsterdam: IOS Press.

Hodgkinson, G., & Clarkson, G. (2005). What have we learned from almost 30 years of research on causal mapping? Methodological lessons and choices for the Information

Systems and Information Technology communities. In V. Narayanan & D. Armstrong (Eds.), *Causal mapping for research in Information Technology* (pp. 46-80). Hershey, PA: Idea Group Publishing.

Homer, J., & Oliva, R. (2001). Maps and models in system dynamics: a response to Coyle. *System Dynamics Review, 17*(4), 347-355.

Hoppenbrouwers, S.J.B.A. (2008): Community-based ICT development as a multi-player game. In: proceedings of conference "What is an Organization? Materiality, Agency and Discourse", may 2008, University of Montreal, Canada.

Howick, S., Ackermann, F., & Andersen, D. (2006). Linking event thinking with structural thinking: methods to improve client value in projects. *System Dynamics Review, 22*(2), 113-140.

Killingsworth, W., Chavez, R., & Martin, N. (2008). *The dynamics of multi-tier, multi-channel supply chains for high value government aviation parts.* Paper presented at the ISDC, Athens, Greece.

P. Kruchten (2000). The Rational Unified Process: An Introduction. 2nd edition. Addison Wesley, 2000

Lane, D. C. (1994). With a little help from our friends. How system dynamics and soft OR can learn from each other. *System Dynamics Review, 10*(2/3), 101 – 134.

Lane, D. C., & Oliva, R. (1998). The greater whole: towards a synthesis of system dynamics and soft systems methodology. *European Journal of Operational Research, 107*, 214-235.

Luna-Reyes, L., Martinez-Moyano, I., Pardo, T., Cresswell, A., Andersen, D., & Richardson, G. (2006). Anatomy of a group model-building intervention: building dynamic theory from case study research. *System Dynamics Review, 22*(4), 291-320.

OMG (2003): Technical Guide to Model Driven Architecture: The MDA Guide v1.0.1, adopted by OMG's Architecture Board in June, 2003. Object Management Group.

OMG (2005): SBVR Pre-final Specification, August 6, 2005. Object Management Group.

OMG (2006): BPMN 1.0: OMG Final Adopted Specification, February 6 2006. Object Management Group.

Otto, P., & Struben, J. (2004). Gloucester fishery : insights from a group modeling intervention. *System Dynamics Review, 20*(4), 287-312.

Ralyté, J., Brinkkemper, S., and Henderson-Sellers, B. (2007): Situational Method Engineering, Fundamentals and Experiences. IFIP vol. 244, pp. 313-327. Boston: Springer.

Richardson, G., & Andersen, D. (1995). Teamwork in group model building. *System Dynamics Review, 11*(2), 113-137.

Richardson, G., & Pugh, A. (1981). *Introduction to system dynamics modelling with DYNAMO*. Cambridge, MA: MIT Press.

Rittgen, P. (2007). *Negotiating models.* Paper presented at the CAiSE 2007.

Roberts, E. (1973). Strategies for effective implementation of complex corporate models. *TIMS-ORSA Interfaces 8*(1, part 1), 26-33.

Ross, R.G. edt. (2003): Business Rules Manifesto, version 2.0. Available at: http://www.businessrulesgroup.org/brmanifesto.htm

Rouwette, E., Vennix, J., & Van Mullekom, T. (2002). Group model building effectiveness. A review of assessment studies. *System Dynamics Review, 18*(1), 5-45.

Schwaber, K. and M. Beedle (2002): Agile Software Development with SCRUM. Prentice Hall.

Sosnovtseva, O., & Mosekilde, E. (2006). Preface. *Journal of Biological Physics, 32*(3-4), 183-189.

Stahl, T., V¨olter, M., Bettin, J., Haase, A., Helsen, S. (2006): Model-Driven Software Development: Technology, Engineering, Management. Chichester: Wiley.

Standish Group (1999): CHAOS: A Recipe for Succes. Technical report, The Standish Group International, West Yarmouth, Massachusetts, USA.

Van den Belt, M. (2004). *Mediated modeling. A system dynamics approach to environmental consensus building*. Washington: Island Press.

Van Zijderveld, E. (2007). *MARVEL - principles of a method for semi-qualitative system behaviour and policy analysis.* Paper presented at the International Conference of the System Dynamics Society, Boston, Mass.

Venderbosch, T. (2007). *Using Group Model Building to optimize the maintenance process in an ERP environment at ONEgas.* Radboud University Nijmegen, Nijmegen.

Vennix, J. (1996). *Group model building. Facilitating team learning using system dynamics*. Chichester: Wiley.

Vennix, J. (1999). Group Model-Building: Tackling Messy Problems. *System Dynamics Review, 15*(4), 379 - 401.

Vennix, J., Akkermans, H., & Rouwette, E. (1996). Group model building to facilitate organisational change. An exploratory study. *System Dynamics Review, 12*(1), 39–58.

Warren, K. (2004). Why has feedback systems thinking struggled to influence strategy and policy formulation? Suggestive evidence, explanations and solutions. *Systems Research and Behavioral Science, 21*(4), 331-347

Wood, J. and D. Silver (1995): Joint Application Development, 2nd Edition. Chichester: Wiley.