Collaborative Concept Mapping on the World Wide Web

Gil Regev
Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland
gil.regev@epfl.ch

Xavier Gilbert
International Institute for Management Development (IMD), Lausanne, Switzerland
gilbert@imd.ch

Alain Wegmann
Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland
alain.wegmann@epfl.ch

Abstract

The Collaborative Concept Mapping project is a joint research project between IMD and EPFL. The goal of the project is to create a collaborative tool that helps managers to define business strategies in the face of complex situations. Addressing complex problems requires collaboration on models in order to integrate multiple views and create shared understanding. Because of the rapidly changing nature of the business world, models have a short life expectancy. Managers cannot wait for the creation of the ultimate models before testing them in the real world. In order for models to be created and shared by people, a method and a tool for building them are needed. In this article we propose a method and a tool that integrate systems theory, management learning, and software engineering practices for addressing complex problems.

Keywords

Management, learning, strategy, modeling, systems theory, complex problems, concept maps.

Introduction

"Man's population and gross product are increasing at a considerable rate, but the complexity of his problems grows still faster, and the urgency with which solutions must be found becomes steadily greater in response to the increased rate of activity and the increasingly global nature of that activity. Augmenting man's intellect, in the sense defined above, would warrant full pursuit by an enlightened society if there could be shown a reasonable approach and some plausible benefits." (Douglas Engelbart, 1962)

Most real world problems are complex. The rate of complexity tends to accelerate as we put together more complex systems. When considering business strategies or new system developments, managers and engineers often face complex problems. We need methods and tools to approach this complexity but these should be *lightweight* so that they can be used in an action oriented business environment.

In this article we will briefly review the nature of complex problems and ways to address them. We will show the need for collaboration and explain our approach, which brings together systems theory, management learning, and software engineering. We will then shortly describe the collaborative tool we are developing to support our approach.

Addressing Complex Problems

Most real world problems are complex problems. Complex (or wicked) problems have several characteristics of which the most important for, our discussion, are: the fact that they involve multiple stakeholders who generally do not agree on the problem to solve; they require iteration; they "require complex judgments about the level of abstraction at which to define the problem," (Buckingham Shum et al, 1996).

Morin (Morin and Le Moigne 1999) defines the 7 principles of complex thinking. In this discussion, we will consider a subset of these principles that describes systems in terms of structure and behavior.

Morin describes behavior in terms of *feedback* and *recursive* loops. Feedback loops can either be negative, moving towards stability, or be positive, moving towards amplification. Recursive loops are responsible for the auto-production and auto-organization of the system (Morin and Le Moigne 1999, pp. 262, 263).

Morin explains the need for distinguishing and linking. Distinguishing is the act of describing the different elements of a system. Linking is the act of describing the relationships that exist between the elements. Another principle of complexity is that the whole is more than the parts. In other words, we can see things in the system when we have an overview of the whole that we cannot see when we consider the parts in isolation (Morin and Le Moigne 1999, p. 261). These properties of the system are sometime called *emergent* properties. Seeing the whole constituted by the different parts and their relationships is thus necessary.

Weinberg links emergent properties with observers (Weinberg 1975, p. 60): "Properties "emerge" for a particular observer when he could not or did not predict their appearance. We can always find cases in which a property will be "emergent" to one observer and "predictable" to another." What is obvious for one person is not obvious to another. By taking into account the multiple views or perspectives that different people have we are less likely to miss important factors.

Thus we can see that collaboration is necessary in order to integrate multiple views. It is also necessary for reasons of appropriation. Appropriation is needed because, to define and implement a business strategy, teams need to agree on what is to be achieved and how it is to be achieved. By participating in the definition of the strategy, people "appropriate" this strategy to themselves. They will be more likely to carry out its implementation.

We very often confuse collaboration with communication. Schrage (Schrage 1995) describes how collaboration is different from communication. His main thesis is that collaboration requires more than just fast and efficient communication.

The Need for a Shared Space to Support Collaboration

Schrage shows that the single most important element needed for collaboration is a shared space (Schrage 1995, p. 94): "Shared space literally adds a new dimension to conversation, a dimension embracing symbolic representation, manipulation, and memory."…"It takes a shared space to create shared understandings." A shared space enables several people to share a set of artifacts and so to build a model that is an externalization of their shared understanding. Thus, the important aspects of a shared space are the symbolic representation, the manipulation of this representation, and the ability to memorize and recall it. The

symbolic representation can be simply text but in order to "distinguish and link" we usually resort to spatial, graphical representations.

Economic and environmental constraints encourage people to move to so called distance collaboration where collaboration does not happen at the same time and same place but at different times and different places (i.e. asynchronously). This places more constraints on the shared space required to support collaboration. Whiteboards and notebooks are not sufficient anymore. We need tools that enable anytime, anyplace collaboration.

The Need for an Approach

Collaboration and its benefits - creativity, shared understanding, shared learning – is not an effortless process. It is time-consuming and requires thought before and after action. In business settings, we often strive for immediate benefits, which places us in a mode that we can call the *bias for action*. Missing from the bias for action is the need for modeling before and debriefing after action. Shared understanding and dealing with complexity will not emerge magically out of mere action. At this point, it is useful to introduce the experiential model of learning (figure 1) developed at IMD (Gilbert 2000, Strebel et al. 2000, Marchand et al. 2000) that can be used to understand how action needs to be complemented to generate learning. We generally start by gathering information from our environment. This is followed by a conceptualization phase in which we create theories and models, and define a learning agenda to confirm or improve our models; this gives us a plan for action. It is followed by action and then reflection in which we review our actions with regard to our models and plans and learn from them. This kind of reflection is extremely critical to learning.

While there's no doubt that focusing too much on information gathering, conceptualization, and reflection and doing little action is not a good strategy in today's economy, it is also true that only action with none of the other activities may be equally harmful. In fact, what is needed is to achieve a balance between information gathering, action, conceptualization, and reflection.

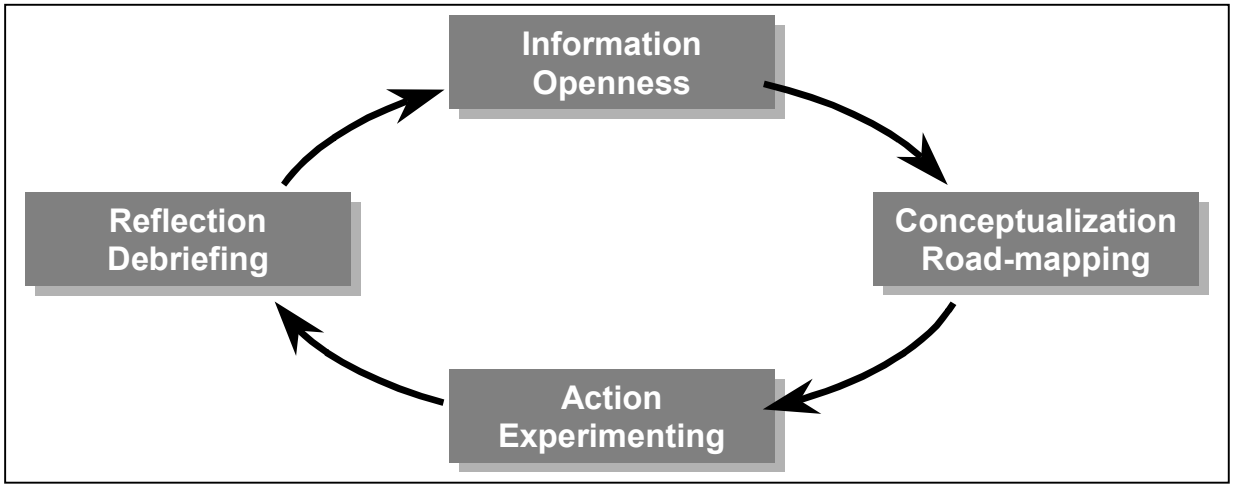

Figure 1. IMD's experiential learning model

The experiential learning cycle assumes that each of these learning steps is merely "work-in-process": information keeps flowing, models pre-exist in the minds of experienced managers and embed hypotheses concerning the effects of future actions, action retains a share of experimenting, and debriefing draws tentative conclusions to be recycled.

The experiential learning model should be seen as occurring repeatedly in small iterations within the scope of one project or one activity. As we will see later, this leads us to propose extreme modeling as a way of rapidly creating models and testing them.

The Need for a Modeling Technique

Externalizing one's knowledge so that it can be shared with others is described by Nonaka and Takeuchi (Nonaka et al. 1995) as transforming tacit knowledge into explicit knowledge. It is generally believed to be impossible to make all tacit knowledge explicit. Creating models is an act of externalizing (or formalizing) knowledge. Shipman and McCall (Shipman and McCall 1999) defined a formalization scale. In their model, free text is at the bottom of the scale, requiring the least amount of formalization and formal modeling languages are at the top of the scale. The difficulty in externalizing knowledge increases, as the tool we use is higher on the formalization scale. Not only do we have difficulties in creating models, we also have difficulties making our models understandable by other people because of, at least, the following reasons:

- **People externalize knowledge in different abstraction levels**. This corresponds to Lakoff's *basic level categorization* (Lakoff 1987, p. 32). People are generally unaware of the *basic level* at which they operate. While it is fairly easy to externalize knowledge at our basic level, it requires much effort to externalize knowledge above or below this level (i.e. by generalizing or specializing).
- **People externalize knowledge in different contexts**. The context in which knowledge is externalized is very often *automatic*. People are generally unaware of the context in which they externalize knowledge.
- **People use different words when they refer to the same entity**
- **People use the same words when they refer to different entities**

Our modeling techniques need to have some semantics in them that help people explicit their abstraction level and context but these semantics should not make the technique too formal. We will need to agree on some form of common vocabulary (i.e. a thesaurus). The discussion of a thesaurus is out of the scope of this paper so we will concentrate on the 2 other reasons.

Concept Maps

Our first attempt at modeling was based on the creation of concept maps. Joseph Novak (Novak and Gowin1984, Novak 1998) and his team developed the concept mapping technique in the 1970's as an aid for science education in schools. Concept mapping is the act of creating networks or maps of concepts and describing their relationships. Novak defines the term concept as being "A perceived regularity in events or objects, or records of events or objects designated by a label" (Novak 1998 p. 22). This definition has the advantage of being very general and is thus supposed to enable people to create concept maps with relative ease. This apparent simplicity leads to three main problems:

- People get stuck in the process of creating a concept map, usually at the beginning or after a certain number of concepts were created because of this apparent freedom.
- People tend to create shallow concept maps that either describe too little or too much and in which the relationships between concepts are rarely named, making it difficult to understand their nature.
- Concept maps normally describe the state of a system at a certain time but the concept mapping technique does not address this subject. Systems are thought of as static entities. This is understandable when we remember that concept maps were originally designed to convey teachers and students' understanding of natural science systems, which, as they are taught in schools tend to be of fairly static nature. Business

systems, on the other hand, are highly dynamic thus requiring more than just concept mapping techniques to be described.
- After a while, the produced concept map tends to be highly complex and frequently not understandable by other people.

System Diagrams

System diagrams (Senge et al. 1999, Risch et al. 1995) were defined in the field of systems thinking. System diagrams were designed to describe behavior rather than structure. They show entities linked by events to form negative or positive feedback loops. The structure, then, is an implicit property of a system diagram. While they are very useful for modeling behavior and for simulation, system diagrams suffer from the very same ills that concept maps do, as it is unclear what is a good entity to describe and what are the events that should link the entities.

Argumentation Diagrams

Argumentation diagrams were designed to help teams to address complex problems by making the decision-making process explicit (Buckingham Shum et al, 1996, Conklin and Begeman 1988). Argumentation diagrams such as IBIS (Conklin and Begeman 1988) and QOC (Buckingham Shum et al, 1996) generally describe problems, ideas with their plus and minuses or options, and criteria for selecting between these ideas. While very effective in capturing and documenting the decision-making process, argumentation diagrams, do not model the structure and behavior of the system.

Software Engineering Diagrams

Software engineers have always been preoccupied with the modeling of reality in order to produce programs. Recently attempts have been made to move towards a standard modeling language which could support business as well as program modeling. This language is called the Unified Modeling Language (UML). UML is the standard notation of the Object Management Group (OMG 2000), an influential consortium of software engineering companies. For our purposes, however, UML has a syntax that is too rich – there are too many different kinds of symbols and diagrams and even though it is supposed to be able to represent business level diagrams it is too focused on software engineering. This makes it very much unusable to non-software engineers.

Our Approach

We have shown the need for an approach, a tool supporting the approach, and a modeling technique to be supported by the tool. In this section, we present our approach, which is based on:

- Extreme modeling
- Modeling structure and behavior
- A collaborative shared space

Extreme Modeling

Extreme Programming (Beck 1999) is a software development process that emphasizes, among others, the following practices: short development cycles with concrete deliveries to clients, pair programming where two developers collaborate side by side to create a program, heavy reliance on testing to make sure that the program runs correctly and that new bugs are not introduced during changes.

Extreme programming was developed to counter the software development methodologies developed during the 1970's that emphasized the completeness of the analysis and design phases before action could be taken - in this instance, programming, testing, and deployment. These methodologies are known today as adhering to the waterfall model that is known to lead to long development cycles (i.e. inaction) and to produce systems that are not usable and do not meet customer expectations and requirements.

As an analogy to Extreme Programming we can say that our approach can be called Extreme Modeling. We want to create models that describe useful aspects of the business but we cannot take forever to create them, making sure that every little detail is correct before we use them. One of the principles of Extreme Programming is that: "Preproduction is an unnatural state for a system and should be gotten out of the way as quickly as possible." (Beck 2000 p. 131). Similarly, we can say that a model that is not applied or shared is in an unnatural state. Our goal, then, is to rapidly create models that can be shared with others, so that they can be refined and quite rapidly tested in the real world; thus providing feedback and enabling further refinements.

Extreme Programming was designed to address the rapidly changing nature of system requirements, which leaves no place for long phases of analysis and design. In our case Extreme Modeling is designed to deal with the rapidly changing nature of our business world that shortens the useful life expectancy of our models and the potential incorrectness of the models we produce. This requires us to continuously test and rework our models in order to reflect the changing nature of the reality we observe.

Extreme programming is meant to provide learning as a more adequate solution develops (double-loop learning). Extreme modeling, by following the experiential learning approach described before, is meant to provide learning as we implement our work-in-progress models.

Modeling Structure and Behavior

To avoid the problem of confusion between abstraction levels and contexts, we propose to guide people through the model creation process using a framework inherited from system science. The advantage in, to our thinking, is that a framework with clearly defined semantics will enable people to create models where their abstraction levels and contexts are explicit.

The thought framework we propose is based on the modeling of structure and behavior. System theorists generally define a system as a set of objects with interrelations between these objects (Von Bertalanffy 1968, p. 55). These relations change over time and thus constitute the behavior of the system. We can model the behavior of the system by creating models that represent the state of the system in different instants in time. Borrowing a definition from the field of software engineering we can say that a state is (ISO/IEC ITU-T, 1995): "At a given instant in time, the condition of an object or an actor that determines the set of all sequences of actions in which the object can take part." We can understand this as the potentiality for action. In order for this potential to exist, an object must know the other objects it may interact with. Or to say it in a simpler way (D'Souza and Wills, 1999, p. 50): The state of an object is "*the information that is encapsulated in it*"

We can represent the state of the system with a series of *snapshots* (D'Souza and Wills, 1999, p. 50) each snapshot describing the state of the system at a given instant in time. A snapshot will show the set of objects comprising the system and their relationships at that instant.

While it is useful to consider a number of states that are representative of the behavior we are trying to model, in most real world settings it is highly impractical to describe all states. The states we choose to represent depend on what is important to us. As Weinberg put it "The states and structures we choose to observe are purely pragmatic" (Weinberg 1975 p. 61).

Putting together a set of states will give us a view of the activities the objects should engage in, in order to move from one state to the next. This description is either a historical account of what has happened in the system, or a plan for moving the system to a new state. In the latter case, the activities can be viewed as our strategy. Once we have the names of the activities, we can define them by describing their pre-conditions and post-conditions. Pre-conditions are the conditions that must prevail before the activity can be started. Pre-conditions can be seen as the conditions necessary for the post-conditions to be materialized. The pre-conditions of an activity are a subset of the state of the system at the time the activity begins and hence are a subset of post-conditions created by past activities. The post-conditions completely define the state of the system after the activity has taken place.
Our framework requires 3 modeling entities:

- A concept is a representation of a real life object. A concept is displayed as a rounded rectangle
- An activity represents a real world activity involving one or multiple objects. An activity is represented as an ellipse. The pre- and post-conditions that define them can be attached to them.
- A link represents a relationship between two concepts and/or an activity. Links can be named to better convey their meaning.

Figure 2 shows a simple example of a problem. We distinguish between objects and activities. Objects exist in different states. Let's consider an extremely simple example of 3 objects. Object 1 (O1) knows object 3 (O3) and tells object 2 (O2) that O3 exists. We define t1 as the time before O1 tells O2 about O3 and t2 after O1 tells O2 about O3. Figure 2 shows the resulting model.

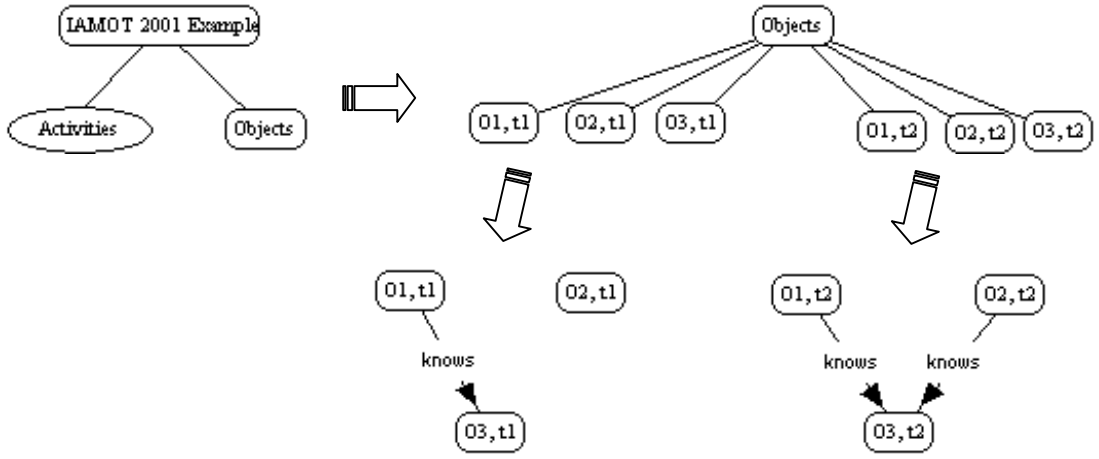

Figure 2. Objects and states

Based on these snapshots, we can describe the activity that must take place in order for object 2 to know object 3. Activity A1 is described by its pre and post conditions, which are in a document attached to the A1 symbol.

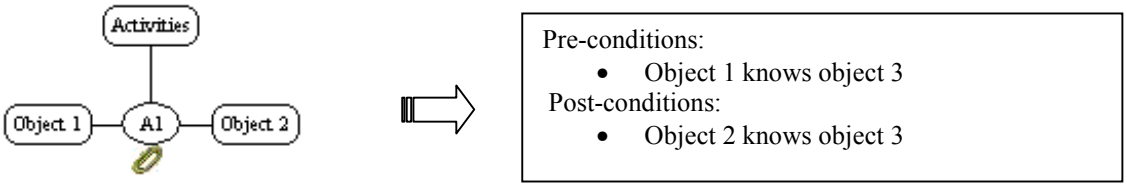

Figure 3. Activities

Collaborative Shared Space

The tool we have built (figure 4) implements a shared space that enables a team of users to collaboratively create models. These models are created using the simple graphical language that we described in our framework. It consists of concepts (rounded rectangles) and activities (ellipses) connected by links (straight lines). For each modeling entity Knoware displays the name of the person who created or modified it and the time of modification. Entities can be private or public. Knoware has a set of awareness tools (Gutwin and Greenberg 1998) for same time and non-same time collaboration. For example, Knoware shows who is connected to the shared space, who logs in or out, and it shows changes made in the model by one person to the other connected people in a slightly deferred real time.

Concepts can contain other concepts. When concepts are "expanded" to reveal the concepts they contain, they expand in the same window as their container concept. This enables users to make connections between concepts contained within different concepts thus breaking down the barriers to linking, very often imposed by categories. Concepts, activities, and links can have attachments. Attachments can be documents or World Wide Web addresses. Comments can be attached to any concept, activity, or link, in a form resembling sticky notes. Each note can contain multiple comments from one or several people. These notes can be seen as mini-chats contextualized by being attached to a concept or relationship.

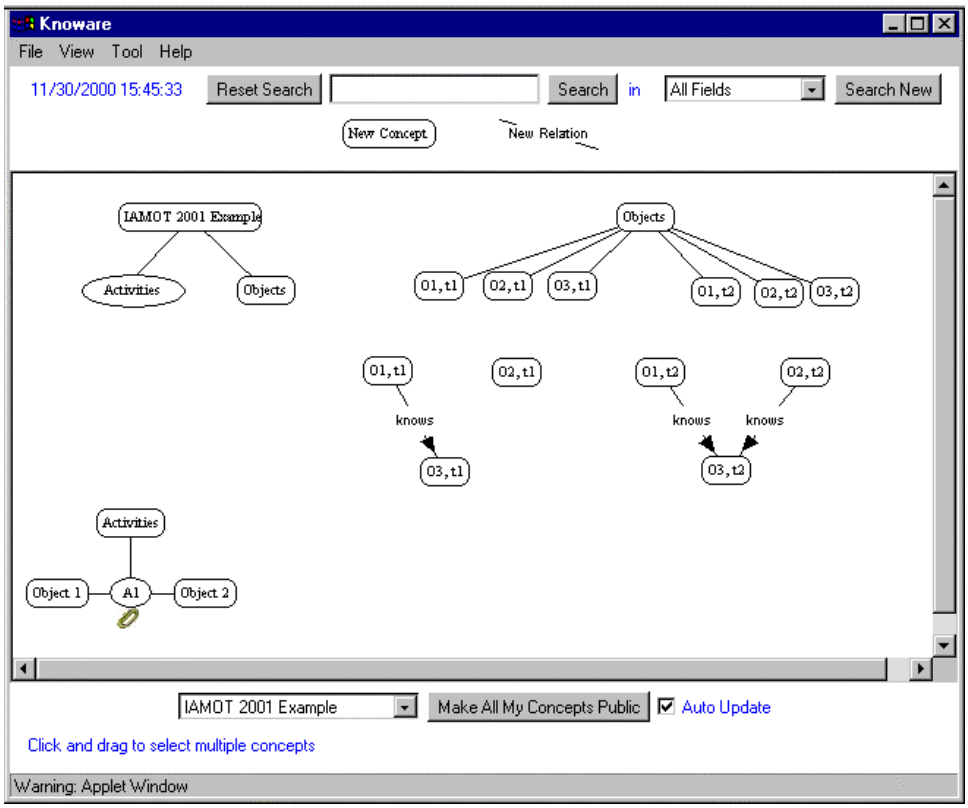

Figure 4. Snapshot of Knoware

Further Research

Our next step is to apply our approach to the business system approach to industry analysis so as to provide a framework that is more specialized for managers. We will then test our framework and tool with EPFL and IMD students in order to draw conclusion as to its usefulness and in order to refine it.

Conclusion

Our approach is inspired by best practices from systems theory, management learning theories, and software engineering. It involves a Web based collaborative-shared space, a lightweight method that we call extreme modeling, and a lightweight framework for modeling structure and behavior.

The collaboration aspects of our tool are very important. As we have showed earlier, different people have different views; they see and model different things. No one can be what Weinberg (Weinberg, 1975) calls the super observer: The observer who sees everything and who is able to arbitrate between the other views. Hence, all our views are necessarily partial. Even combining different views will always yield another partial view. Our hope is that this partial view, created by a group of people, will be useful enough to develop a strategy that leads to action.

The framework we described can be characterized as a lightweight formal framework. It is formal because it imposes specific semantics on the model, objects in various states, actions and relationships. It is lightweight because within this format, it leaves the user a great deal of freedom in what and how to model. It also uses a very small number of modeling entities, which makes it relatively easy to select one or the other without doubting too much whether it is right or wrong. In our view this is important if we want managers to be involved in business modeling.

References

Beck, K, **eXtreme Programming explained – Embrace Change**, Reading MA, Addison-Wesley, 2000.

Buckingham Shum, S., MacLean, A., Bellotti, V. & Hammond, N. **Graphical Argumentation & Design Cognition**. 1996, http://kmi.open.ac.uk/~simonb/csca/graph-arg-design/graph-arg-whole.html.

Conklin, J., Begeman, M. L., **gIBIS: a hypertext tool for exploratory policy discussion,** ACM Transactions on Information Systems, Volume 6 , Issue 4 (1988), Pages 303-331

D'Souza D. F., Wills A. C., **Objects, Components, and Frameworks with UML – The Catalysis Approach**. Reading, MA, Addison-Wesley 1999.

Engelbart D. C., **AUGMENTING HUMAN INTELLECT: A Conceptual Framework**, SRI, AFOSR-3233 Summary Report, Menlo Park, CA, 1962.

Gilbert X, **How managers learn,** Internal IMD Presentation, 2000.

Gilbert X, **Using Collective Learning to Steer the Organization**, Internal IMD presentation, 2000.

Gutwin, C. and Greenberg, S., **Effects of Awareness Support on Groupware Usability**. Proceedings of the CHI'98 Conference on Human Factors in Computing Systems, ACM Press 1998.

ISO/IEC ITU-T: **Open Distributed Processing – Basic Reference Model – Part 2: Founda-tions**. Standard 10746-2, Recommendation X.902 1995 (http://isotc.iso.ch/livelink/livelink/fetch/2000/2489/Ittf_Home/PubliclyAvailableStandards.htm)

Kolb, D., A., **Experiential Learning: Experience as the Resource of Learning and Development**, Prentice-Hall, Englewood Cliffs, 1984.

Lakoff, George, **Women, Fire and Dangerous Things – What Categories Reveal about the Mind**, Chicago Press, 1987

Marchand, Ed., **Competing with Information**, Wiley, 2000.

Meyer B., **Object Oriented Software Construction – Second Edition**, New Jersey, Prentice Hall, 1997.

Morin, E, Le Moigne, J. L., **L'intelligence de la complexité**,Paris, L'Harmattan, 1999.

Nonaka I., Takeuchi, H., **The Knowledge Creating Company**, New York, Oxford University Press, 1995

Novak, J., D., & Gowin, D., B., **Learning how to learn**. New York: Cambridge University Press, 1984.

Novak, J. D., **Learning, Creating, and Using Knowledge: Concept Maps as the Facilitative Tools in Schools and Corporations,** New York: Cambridge University Press, 1998.

OMG, **Object Management Group**, http://www.omg.org, 2000.

Risch, J. D., Troyano-Bermúdez, L., Sterman, J. D., **Designing corporate strategy with system dynamics: a case study in the pulp and paper industry**, System Dynamics Review, Vol. 11, no. 4 (Winter 1995), pp. 249-274.

Schrage, M., **No More Teams! : Mastering the Dynamics of Creative Collaboration**, New York, Currency/Doubleday, 1995.

Schrage, M., **Serious Play – How the world's Best Companies Simulate to Innovate**, Boston, Harvard Business School Press, 1995.

Senge P. et al., **The Dance of Change**, London, Nicolas Brealey, 1999.

Shipman, F. M., Marshal C. C., **Formality considered harmful: Experiences, emerging themes, and directions**. Tech. Rep. CU-CS-648-93., Boulder, CO, Department of Computer Science, University of Colorado at Boulder, 1993.

Shipman, F. M., McCall, R. J., **Supporting Incremental Formalization with the Hyper-Object Substrate**, ACM Transactions on Information Systems, 17, 2 (April 1999), pp. 199-227.

Strebel, Ed., **Focused Energy**, Wiley, 2000.

Von Bertalanffy, L, **General System Theory**, New York, George Braziller, 1969.

Weinberg, G. M., **An Introduction to General Systems Thinking**, New York, Wiley & Sons, 1975.

Authors Biographies

Xavier Gilbert (DBA, Harvard Business School) is Professor of Industry Analysis and Strategy and holds the LEGO Chair in International Business Dynamics. His areas of special interest are competitive analysis, strategy implementation with specific attention to management development and organization learning implications

Alain Wegmann earned his EE degree at EPFL in 1981. He completed his Ph.D. in office automation at INRIA (France) in 1984. He worked for 14 years with Logitech (Switzerland, Taiwan, US) in positions ranging from software developer, VP engineering and VP OEM marketing. He joined EPFL in 1997. His activities are focused on research and teaching in system architecture, software engineering and UML semantics.

Gil Regev earned his CS degree at EPFL in 1988. He worked for 9 years with Logitech in Switzerland and Silicon Valley as software engineer and project manager. He joined EPFL in 1997 and is now pursuing a Ph.D. with EPFL and IMD in the area of collaborative learning in business.