A Formal Model for the Fifth Discipline Part II - Detailed Version

Lourival Paulino da Silva, and Flávio Soares Corrêa da Silva

Department of Computer Science
University of São Paulo
Rua do Matão, 1010
São Paulo - SP - Brazil
E-mail: {loupaul,fcs}@ime.usp.br

Abstract

This paper contains a formal model that we have developed for the Fifth Discipline theory [Sen90]. This model uses the SMART Multi-Agent Systems framework [dL01] as a foundation for the formalization. The formalization of this theory allows us to study its models and underlying assumptions concerning the organization, their members and the interactions among them. The contributions of this article include not only a formal model for Senge's theory, but also analyses that indicate that several individual features play an important role in Senge's theory. Thus, agents must be honest, cooperative, tenacious, and trust is fundamental in the agents' interactions.

1 Introduction

In this paper we present a formalization, from a Multi-Agent Systems (MAS) perspective, for an important and modern Organizational Theory (OT). Based on this model, our goal is to study relevant features of such OT: the models and underlying assumptions about an organization that implements such OT; their members and the interactions among members; and the interactions among the organization as a whole and its members.

For this work, an OT was selected that has gained attention from industry and academia, Senge's Fifth Discipline [Sen90]. Correspondingly, a particular MAS formal framework was also adopted - SMART [dL01]- to serve as the foundation for the formalized OT theory. SMART uses the Z [Spi92] specification language to formally define MAS related concepts and terms.

It is important to note that the formal model presented in this paper corresponds to our interpretation of Senge's work and depicts some characteristics of his theory, thus is not intended to be a full detailed translation of the Fifth Discipline theory into a formal model.

We also note that we have made our best to keep this article self-contained. Nevertheless, to keep it of reasonable size, the presentations of both the Fifth Discipline and the Z formal specification language are extremely condensed and brief. We suggest to the interested reader that some additional texts be consulted for clarification of these topics, e.g. [Sen90, SKR+94] for the Fifth Discipline and [Spi92, Jac97] for Z.

The contributions of this article include not only a formal model for Senge's theory, but also analyses that indicate that several individual features play an important role in Senge's theory. Thus, agents must be honest, cooperative, tenacious, and trust is fundamental in the agents' interactions.

In addition, the applicability of our approach presents new perspectives and reveals new, not yet explored, potentialities concerning the use of formal methods - developed for the area of Software Engineering (SE) - for modeling systems in general. For example, consistency checking of an organization with regard to a specific organizational theory can be investigated.

The structure of this paper is the following: in section 2, we present an overview the Fifth Discipline theory. In section 3, we present some excerpts of a formal model for this theory, including additionally, issues regarding knowledge and interactions among agents in an organization that implements Senge's theory. In section 4, we discuss some properties of the Fifth Discipline theory and properties of the formalized version of this theory. In section 5, we briefly discuss work that relates to our research. Finally, in section 6, we present our concluding remarks and discussion.

2 The Fifth Discipline

Organizational Theory deals with issues related to the structure, design, and performance of organizations. This field describes how organizations are structured and suggests how to build new organizations, or how old ones can change. The main goal of this field is to improve organizational effectiveness.

Different approaches, or schools, were developed and gained attention in the last hundred years of research in OT, influenced by historical, social, cultural, technological, and economic contexts. Starting from the work of Taylor[Tay11] and Fayol[Fay49] where organizations, viewed as machines, were expected to produce at their maximum performance with minimal resource consumption; several schools have emerged: for example the Human Relationship School, Theory of Behavior, Systems Theory and Contingency Theory. In this path, it is noticeable a change from prescriptive to descriptive and explanatory paradigms. Similarly, the concept of man and its individual behavior changes from the simplistic *homo economicus* model of an isolated individual, motivated by material and monetary incentives, to more complex models that also include social motivations [Chi00, MSB99].

Accompanying this trend, and as a consequence of the advent of the knowledge era, the school of Organizational Learning has been growing in the last 40 years, receiving increasing attention from both the academic and business areas, and has become an important field of research.

This school focuses in the processes of learning; innovation; knowledge creation, storage, manipulation and transfer; in order to improve organizational behavior to produce increased organizational performance and adaptability. Differently from other schools, here, greater attention is paid to the management of intangible assets which may award competitive advantage to the organizations

that excel in such endeavor.

There are different views regarding learning organizations: [Arg77] defines that "organizational learning is a process of detecting and correcting error."; [FL85] states that "organizational learning means the process of improving actions through better knowledge and understanding"; [Gar93] describes a learning organization as "an organization skilled at creating, acquiring, and transferring knowledge, and at modifying its behavior to reflect new knowledge and insights.".

One of the first references to the term "organizational learning" is presented in [CD65], where organizational learning is viewed as the result of interactions among adaptations in different levels: subgroup or individual, and organizational. Three types of stresses causes these adaptations: discomfort, performance and disjunctive. Discomfort stress results from the complexity of the environment and from the team's effort to understand it. Performance stress is associated to the organizational pressure to achieve successful individual actions. Finally, disjunctive stress is a consequence of increasing degrees of disagreements and conflicts resulting from different behaviors of teams and individuals. Discomfort and performance stresses cause subgroup and individual adaptations. Organizational learning is a consequence of influences coming from performance and disjunctive stresses.

According to [Kim93] all organizations learn, regardless of specific learning oriented plans. Organizational learning occurs by means of individual learning. As a consequence, this author proposes a model that connects individual learning to organizational learning via mental models. Individual learning involves two levels: operational and conceptual. The first refers to learning the steps associated to the execution of a particular task and is expressed by sequences of routine activities. Not only operational learning affects routines but also routines influence learning. On the other hand, the conceptual level involves reflection about how activities are performed and may cause their revision or removal. Hence, the model of individual learning involves a cycle of conceptual and operational learning that receives information from and provides information to mental models. These cycles influence the organizational shared mental models and, consequently, affect organizational learning.

In [CLW99] organizational learning involves a tension between assimilation of a new learning and utilization of what has been learned, named "tension of strategic renewal". The authors present a framework based on the assumption that the renewal of the entire organization must be the underlying phenomenon to be investigated. This framework takes into account four social and psychological processes: intuiting, interpreting, integrating, and institutionalizing; named *4I's*. Moreover, there are three levels of organizational learning: individual, group, and organization. In this framework the *4I's* are related in feed-forward and feedback processes through these learning levels.

A learning organization is defined in [DTC97] as the one that develops competences so as to, first maintain and improve current performance and, second, adapt the organization to ensure future performance. Therefore, there is a requirement to develop and maintain the entire capacity of management of an

organization, including the development of corporate and personal competences of managers. In fact, such development must become itself an organizational competence, so that it is always applied and incorporated in the organizational systems, structures, values and policies.

An influential work in this field is [Non91], which states that successful organizations consistently create new knowledge, widely disseminate it through the entire organization, and quickly incorporate it in new technologies and products. The author defines two types of knowledge: explicit and tacit. The first is formal and systematic; written in norms, specifications or formulas, it can be communicated and shared. The second has two dimensions: one involves technical skills and professional experience; the other is cognitive and includes beliefs and mental models that influence the way people perceive the environment. These dimensions are hardly articulated, therefore it is difficult to communicate and share this type of knowledge. The classification of knowledge into tacit and explicit lead to the definition of four patterns to create knowledge: from tacit to tacit, from tacit to explicit, from explicit to explicit, and from explicit to tacit. The author defines a dynamic interaction among these patterns: a spiral of knowledge that involves socialization, articulation, combination and internalization. Each of these processes correspond, respectively, to one of the patterns. The knowledge-creating organization keeps this spiral in continuous activity. In addition, articulation and internalization are critical in this spiral as they depend on individual commitment to enable their implementation.

One of the most recent and influential contributions to the school of Organizational Learning is presented by Senge in [Sen90], also referred in this work as the LO theory. According to Senge, a learning organization is the one that is "continually expanding its capacity to create its future".

Senge states that the life cycle of most organizations is relatively short. To exemplify, he cites a research from Royal Dutch/Shell, completed in 1983, which shows that one third of the enterprises mentioned in fortune "500", 1970 edition, had stopped their activities by the time of publication of that research. Further, the same research revealed that the average life time of the largest industrial organizations was less than 40 years. Although in most of the enterprises that fail there are several evidences of problems, they are not able to recognize the imminent threats, perceive its implications, or recommend alternatives. Senge states that these difficulties are related to several learning deficiencies:

- "I am my position".

  Individuals that only concentrate in their positions, or roles, in an organization, lose the sense of responsibility relative to results that are obtained via interactions involving several positions.

- "The enemy is out there".

  In general, this affirmation corresponds to an incomplete view of a given situation, as "inside the organization" or "outside" are just parts of the same system.

- "The illusion of taking charge".

  Often, proactive attitudes just conceal disguised reactive attitudes, attempts to face "the enemy out there". Genuinely proactive attitudes are consequence of a clear perception of the individual's contribution to his/hers own problems.

- "The fixation on events".

  It is a requirement to recognize long term patterns and all the cause-effect connection chain, thus, avoiding concentration on events only.

- "The parable of the boiled frog".

  Organizations, in general, are not prepared to face gradual threats to their existence. It is important to observe both fast and gradual changes.

- "The delusion of learning from experience".

  Direct experience is the most powerful source for learning. However, when dealing with organizational problems, often, it is not possible to correlate the consequences of important decisions with their real causes.

- "The myth of the management team".

  The management team is composed of managers from different functions and specialized areas in the organization. Supposedly, it should face all the deficiencies mentioned above. However, internal political disputes in these teams cause waste of energy and time, while trying to exhibit the impression of coherent team.

2.1 Structure influences behavior

In the context of Senge's theory, the term structure refers to systemic structure, i.e., the relationships among key variables, like natural resources, population and food production. Different people acting on the same systemic structure tend to produce qualitatively similar results, as the observed behavior is a result from the influence of this structure.

It is necessary that the individuals redefine the extent of their influence on the whole system, observing how their decisions influence the others, and taking into account the relationships among key variables. This is a requirement so that the individual performance overcome (or efficiently manage) the constraints and problems imposed by the systemic structure.

Usually, individual behavior result from an approach based on understanding events. As a consequence, this attitude is reactive. In a systemic attitude there are several levels of explanation. as shown in figure 1.

The comprehension of behavioral may supply the individual with suggestions on how to respond in the long term to changes in trends.

The structural level, in its turn, is the most important and powerful. This level focuses on the underlying causes for a given behavior and, as a consequence, behavioral patterns may be changed.

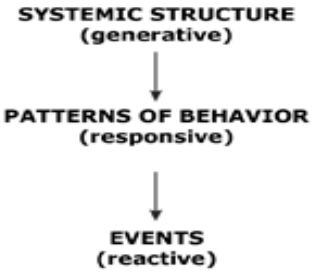

Figure 1: Levels of explanation - a systemic perspective.
Source: adapted from [Sen90, p. 52].

According to Senge, the above mentioned deficiencies permeate human history and culture. Therefore, he presents the Fifth Discipline theory, an antidote against these deficiencies.

Senge's theory comprises the following disciplines: systems thinking, personal mastery, mental models, shared vision and team learning.

The "Learning Organization" follows the principles of the Fifth Discipline and is continually expanding its capacity to create its future. In this type of organization, adaptive learning must be complemented with generative learning, i.e., learning that increments creative capacity.

2.2 Systems thinking

The main concept in this discipline is that an individual can only understand patterns of events in complex systems via a global view. Businesses are also systems in which interrelated actions take some time to totally develop their effects.

An individual involved in such system has difficulties to clearly understand the whole pattern of changes. Therefore, he/she often tends to concentrate on isolated aspects of the problem, aiming at finding a solution that, in fact, should have a global scope.

Systems thinking is a conceptual framework, including knowledge and tools developed in the last 50 years. Its goal is to make complex patterns clearly visible, thus helping an individual to effectively change situations with minimal effort. In Senge's terms, an individual must find the "leverage points of the system".

The basic concepts for systems thinking are delays and two types of feedback processes: reinforcement and balancing. Reinforcement processes operate like an engine of growth (or decline), exhibiting a "snow ball" effect. In the figure 2 we present a reinforcement process for a product: increase in sales causes more positive word of mouth, which results in more sales, and so on.

On the other hand, a balancing process operates when behavior is driven

Figure 2: Reinforcement process.
Source: adapted from [Sen90, p. 81].

by a (implicit or explicit) goal. In figure 3 the individual notices a difference between desired and actual temperature of his/her body, adapts his/her clothing trying to reduce the difference.

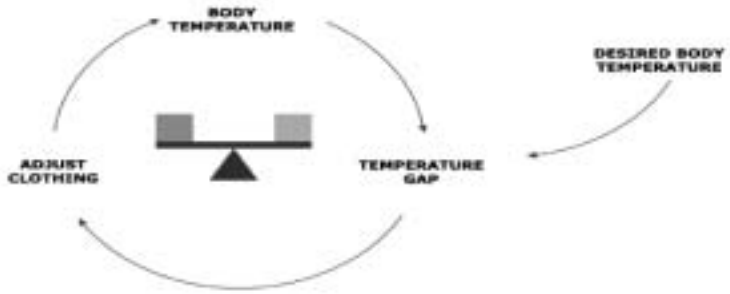

Figure 3: Balancing process.
Source: adapted from [Sen90, p. 84].

Furthermore, frequently feedback processes have delays embedded in their operation. These delays are interruptions in the flow of influences, so that effects related to some actions occur gradually. Figure 4 shows a balancing feedback process with delay: in order to reduce the difference between perceived and desired water temperatures the individual adjusts the volume of water. However, changes in temperature only occur after a given time interval.

Archetypes are generic structures of recurrent patterns, which were developed to facilitate the study of systems dynamics, enabling the identification of the different structures in action and potential leverage points. Archetypes are built on top of reinforcing and balancing processes.

In figure 5 we present the "limits to growth" archetype where a reinforcement

feedback process starts in order to produce a result. A spiral of success is created. However, as a side effect, a balancing process also develops, and this will, eventually, decelerate the success.

Causal loops are flexible diagrams that exhibit the type of relationship between a pair of concepts. They show whether an increase in a variable causes a corresponding increase or decrease in another related variable.

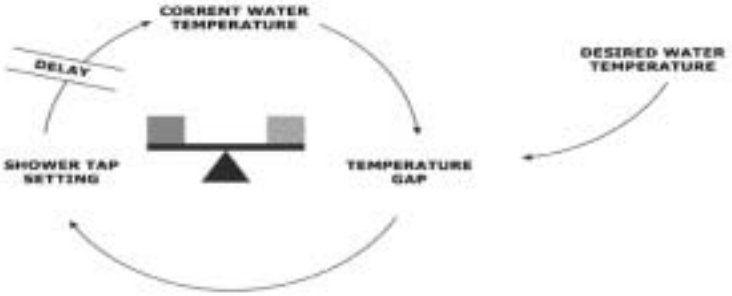

Figure 4: Balancing process with a delay.
Source: adapted from [Sen90, p. 90].

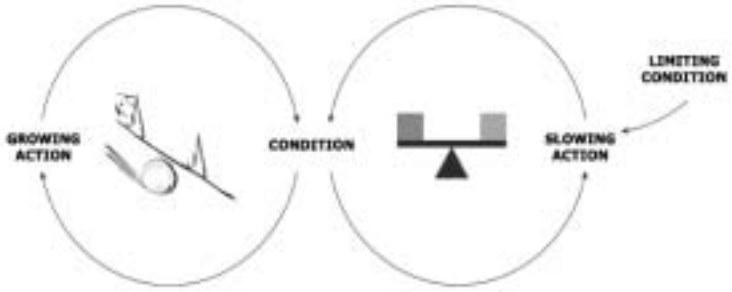

Figure 5: Limits to growth.
Source: adapted from [Sen90, p. 97].

2.3 Personal mastery

Personal mastery consists in continually deepening and clarifying an individual's personal vision, focussing energy, and developing an objective view of reality. This discipline provides the link between organizational and individual learning. Hence, it is essential for organizational learning development.

Senge notes, however, that few organizations support this personal development, resulting in waste of resources. On the other hand, he also observes that few individuals really dedicate their best efforts to develop this mastery. In fact, sustainable learning can only occur when its foundation lies in the individual's commitment with self-development.

The main practice of the personal mastery discipline consists in the development of a capability of maintaining not only a clear view of the current reality, but also a personal vision. This process produces an internal force, named "creative tension". Resolution of this tension involves actions aimed at making the image of reality get closer to the personal vision (figure 6).

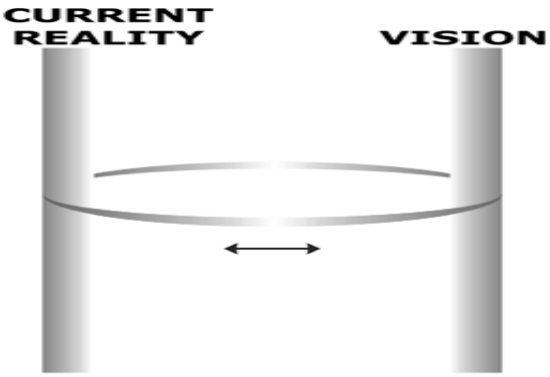

Figure 6: Creative tension.
Source: adapted from [Sen90, p. 151].

2.4 Mental models

Mental models are hypotheses and generalizations that influence both the individual's comprehension of and interaction with the world. Senge states that, often, individuals are not aware of their mental models and the influence they have on their behavior.

Frequently, organizational changes fail to be implemented as a consequence of conflicts they generate with powerful pre-existent mental models.

This discipline involves exposing these models such that they can be criticized and reviewed, if required. In order to achieve this goal, conversations (dialogs) that balance questioning and advocacy are used, so that the members of the organization can both expose their thoughts and become open to influences from others.

Two skills are fundamental to this discipline: reflection and inquiry. The first involves decreasing the speed of the member's thinking processes, so that he/she can recognize how his/her mental models are built. The second consists in conversations among individuals where personal visions are openly shared, and at the same time, they learn each others assumptions on several subjects.

2.5 Shared vision

The existence of goals, values and mission that are profoundly shared through all organization is essential in order to build a successful organization. Such a shared vision constitutes an image of the future that the organization wants to build, and is fundamental so that the organization may unite its members around a shared identity and a sense of destiny.

Once there is a true vision, organizational members want to learn and produce at their best performance. There is no need for a norm or rule that states that members have to behave with such goal in mind.

This discipline aims at translating individual vision into a shared vision by means of the definition of principles and directive lines.

The practice of this discipline involves exposing images of the future which promote the alignment and commitment of the persons with this vision, instead of submission to a vision imposed by the organization.

2.6 Team learning

According to this discipline it is possible to develop learning skills in a team. Once this goal is achieved, this team develops uncommon capabilities for coordinated action, thus producing extraordinary results. This is a state that also permits faster development of the team members than it would be possible in other circumstances.

This discipline involves the development of the dialog technique. This technique concerns the capability of the members to suppress their individual assumptions and enter an authentic state of shared thought. The concept of dialog in this theory involves the free flow of meaning among the members of a team, so that the team can discover insights that could not be discovered individually. Moreover, the practice of dialog also involves learning patterns of interaction among members that hinder or reduce performance in team learning. Another interesting aspect of the dialog technique is that it also includes the development of skills to recognize interaction patterns in teams that hinder team learning, e.g., defensive patterns. Senge also contrasts dialog and discussion: discussion involves competitive exchange of ideas and opinions in order to determine a "winner" idea.

Team learning is also a fundamental discipline for LO theory as teams are the basic collective learning units in organizations and, thus, team learning is a prerequisite for organizational learning.

2.7 Computational models

Computational models provide a concrete vision of the effects resulting from adopting a given set of hypotheses, contrasting with the archetypic world in which systemic elements are vaguely defined and where resulting behavioral patterns are speculative.

These models are utilized for[SKR+94]:

- Exhibiting how systemic structures directly produce behavioral patterns.
- Verifying whether a given structure presents the same performance observed in the real world.
- Investigating what is the change in behavior when several aspects of the structure are modified.
- Exhibiting leverage points that can only be uncovered using such tools.
- Involving teams in-depth systems learning by enabling the investigation of the relationship involving thinking patterns and their consequences.

The main problem in computational modelling is associated to model design: it is difficult to learn how to represent reliably the world. Notwithstanding, computational modelling and simulation is essential for systems thinking, because it is a tool for the development and learning of this discipline.

2.8 All disciplines matter

Senge states that, in the context of his theory, a discipline corresponds to a body of theory and technique, that must be studied and mastered with the objective of practical application. Moreover, practicing a discipline involves continuous learning.

The author also affirms that the five disciplines must be developed in consonance, in a coherent body of theory and practice, which is integrated by the systems thinking discipline. For this reason, he defines systems thinking as the fifth discipline. However, systems thinking also depends on the other four disciplines in order to develop its full potential.

In summary, systems thinking enable understanding of a subtle aspect of organizational learning: a new self-perception for each individual and a new perception of the world. Hence, individuals must observe their connections with the world, continuously creating and changing reality.

In summary, the LO theory places great emphasis in the individual behavior, initiative, and learning capabilities, in order to obtain an emerging organizational behavior. Furthermore, this theory also concentrates on skills that can only be developed collectively, e. g. shared vision and team learning. This focus on both individual and collective aspects, structured into five disciplines; combined with a set of important cases of implementation that include important companies like: Shell, Harley-Davidson, Kyocera, Federal Express, and

Hanover Insurance; makes the LO theory a modern, influential and successful organizational theory and, therefore, a very interesting subject to our study.

3 Modeling the Fifth Discipline as MAS

The area of MAS has gained attention during the last decade. However, many concepts involving agency and agent-based development still lack sound conceptual foundation. In order to provide one such foundation, [dL01] built the SMART conceptual framework.

SMART was chosen in our research because it is a formal, unambiguous, structured, and extensible framework created to serve as a foundation for studying and building different agent oriented architectures. For example, in [BCV98, LyLLd01, MLd03] different models based on SMART are presented. According to [dL01], the Z specification language is used to specify SMART for reasons that include: it enables to formally develop designs of systems; it has close connections to software implementation; and it uses a simple notation, which is expressive, structured, and has acceptance in the Artificial Intelligence (AI) and Software Engineering (SE) communities. While other formalisms could have been used, like VDM [Jon90], CSP [Hoa78], CCS [Mil89], or modal logics [Che80]; none of them offer the same combination of features required by that research work [dL01].

In appendix A we present introductory information concerning Z. However, we note that this basic introduction may not provide sufficient information for the reader with no previous experience with Z. In this case, we suggest [Spi92, Spi89, Jac97] for an introduction to this notation.

3.1 The SMART framework

In SMART, the world is made of entities, which are specified as a collection of attributes. Moreover, entities are classified according to additional features they have: objects are entities with an associated collection of capabilities; agents have a set of goals; and autonomous agents have motivations.

In order to characterize these elements more formally, some basic types must be defined. The types *Attribute*, *Action*, *Goal* and *Motivation* are defined in [dL01] as follows: "attribute is a perceivable feature, action is a discrete event that can change the state of the environment when performed, goal is a state of affairs to be achieved in the environment, and a motivation is any desire or preference that can lead to the generation and adoption of goals and that affects the outcome of the reasoning or behavioral task intended to satisfy those goals". *Action* and *Motivation* are defined as given sets, while *Goal*, *View* and *Actions* are introduced via abbreviation definition as non-empty sets of attributes and a set of actions, respectively. The *Attribute* type corresponds to formulae in first order predicate calculus[1].

[1] Definition ommited here. Refer to [dL01] for this definition.

$[Motivation, Action] \qquad View == \mathbb{P}_1\ Attribute$

$Goal == \mathbb{P}_1\ Attribute \qquad Actions == \mathbb{P}\ Action$

The highest level of abstraction in SMART corresponds to the type *Entity*, which is specified as a non-empty set of attributes, and sets of actions, goals and motivations. In our formalization, however we include several new variables that are important to specify other agent types later. Just as in the original *Entity* type in SMART we only require that the set of attributes is non-empty.

Entity

$attributes, store : \mathbb{P}\ Attribute$
$capableof : \mathbb{P}\ Action$
$goals, allgoals : \mathbb{P}\ Goal$
$motivations : \mathbb{P}\ Motivation$
$ownedresources : \mathbb{P}\ Entity$
$instsreq : Plan \pfun (\text{seq}(Action \times \mathbb{P}\ Entity))$
$resourcesofplan : Plan \pfun \mathbb{P}\ Entity$
$plans, allplans : \mathbb{P}\ Plan$
$planforgoal : Goal \pfun \mathbb{P}\ Plan$
$orgs : \mathbb{P}\ Organization$
$roles : \mathbb{P}\ Role$
$ownguidingideas : GuidingIdea$
$personalMasteryCapabilities, mentalModelsCapabilities : \mathbb{P}\ Action$
$systemsThinkingCapabilities, learningTeamCapabilities : \mathbb{P}\ Action$
$buildingSharedVisionCapabilities : \mathbb{P}\ Action$

$attributes \neq \varnothing$

Entities must be situated in an environment. The type *Env* defines the environment as a set of attributes that depicts all currently true attributes within that environment. The environment state includes all entities inside it.

$Env == \mathbb{P}_1\ Attribute$

EnvironmentState

$env : Env$
$entities : \mathbb{P}\ Entity$

$\bigcup \{e : Entity \mid e \in entities \bullet e.attributes\} \subseteq env$

An object is an entity that is capable of some actions. The object actions will be performed in an environment and will also depend on the state of such environment.

$$Object \mathrel{\widehat=} [Entity \mid capableof \neq \emptyset]$$

ObjectAction

$$Object$$
$$objectact : Env \rightarrow Actions$$

$$\forall\, env : Env \bullet (objectact\ env) \subset capableof$$

ObjectState

$$EnvironmentState$$
$$ObjectAction$$
$$willdo : Actions$$

$$willdo = objectact\ env \land willdo \subset capableof$$

As for object operation, state related variables can change, while other variables are immutable (if they change, instantiation of a new object occurs [dL01]). Therefore, attributes, capabilities, and action selection do not change.

Δ *ObjectState*

$$ObjectState$$
$$ObjectState'$$
$$\Xi ObjectAction$$

Interactions change the state of the environment by adding and removing attributes. The *ObjectInteracts* schema shows individual objects interacting with its environment.

$$effectinteraction : Env \rightarrow \mathbb{P}\, Action \nrightarrow Env$$

ObjectInteracts

$$\Delta ObjectState$$

$$env' = effectinteraction\ env\ willdo$$
$$willdo' = objectact\ env'$$

An agent is an object that has (adopted or generated) goals. The *AgentPerception* schema defines the perception capabilities of an agent. The *canperceive* function defines the attributes that can be perceived while the *agperceives* function defines the attributes actually perceived by the agent. The *AgentAction* schema defines that goals, perceptions and the environment influence the agent's behavior.

$$Agent \mathrel{\widehat=} [Object \mid goals \neq \emptyset]$$

AgentPerception

$$\begin{array}{l} Agent \\ peractions : Actions \\ canperceive : Env \rightarrow Actions \rightarrow Env \\ agperceives : \mathbb{P}\, Goal \rightarrow View \rightarrow View \\ \hline peractions \subset capableof \\ \forall\, env : Env;\ as : Actions \bullet \\ \quad (as \in (\mathrm{dom}(canperceive\ env)) \Rightarrow \\ \quad as = peractions) \\ \mathrm{dom}\, agperceives = \{goals\} \end{array}$$

AgentAction

$$\begin{array}{l} Agent \\ ObjectAction \\ agentact : \mathbb{P}\, Goal \rightarrow View \rightarrow Env \rightarrow Actions \\ \hline \forall\, g : \mathbb{P}\, Goal;\ v : View;\ env : Env \bullet \\ \quad (agentact\ g\ v\ env) \subset capableof \\ \mathrm{dom}\, agentact = \{goals\} \end{array}$$

Applying the same reasoning presented above, more complex agent types may be specified via schema inclusion. For example, agents that are autonomous, have memory, are capable of planning, and perform roles in organizations, are specified in a incremental manner, as successive layers of increasing complexity[2]. Furthermore, interactions among agents can be described. Thus, the formal specification of individual, collective, and organizational aspects can be performed, as we show in the next section.

3.2 Learning Organization Agents

This section contains some excerpts of a formal model for the Fifth Discipline, just to provide an overview of what has been produced. Due to space limitation, some types and concepts are not presented in this work. This is the case for the types: *SAutoAgent*, *PlanningAgent*, *OrgAgent*, *Organization*, and *Role* for example. The Z specification was type checked using ZTC version 2.03 [ZTC03].

Our goal in this incremental definition via refinement of agent types, is to define a more specialized type of agent which corresponds to a formal specification of the requirements described in Senge's theory. Thus, in this section we

[2] In Appendix B, figure 7 presents a structure diagram that displays the structure of part of our specification. This type of diagram was introduced by Luck and D'Inverno [dL01].

introduce the Learning Organization Agent, which is an organizational agent, develops personal mastery, has mental models and develops systems thinking skills. Among the motivations of this agent we have: reducing creative tension and commitment to a clear perception of reality. In order to present this formalization, we divide Senge's disciplines in two groups: intra-personal and inter-personal, according to the type of skills that such disciplines allow the agent to develop. The first group mainly deals with the agent's internal states and representations, and the second deals with interactions among Learning Organization Agents.

3.2.1 Disciplines: Actions, Principles

According to Senge, all disciplines have associated actions and principles, which an agent that develops a given discipline is supposed to perform and know. Thus, below we have *PMActions* and *PMPrinciples* corresponding to personal mastery actions and principles respectively. Similarly, *STActions* and *STPrinciples*; *MMActions* and *MMPrinciples*; *SVActions* and *SVPrinciples*; *LTActions* and *LTPrinciples*; correspond to systems thinking, mental models, shared vision, and team learning, actions and principles respectively.

$$
\begin{array}{|l}
AllDisciplinesActions, PMActions : \mathbb{P}\, Action \\
STActions, MMActions, SVActions, LTActions : \mathbb{P}\, Action \\
\hline
PMActions \neq \varnothing \\
STActions \neq \varnothing \\
MMActions \neq \varnothing \\
SVActions \neq \varnothing \\
LTActions \neq \varnothing \\
AllDisciplinesActions = PMActions \cup STActions \cup \\
\quad MMActions \cup SVActions \cup LTActions
\end{array}
$$

$$
\begin{array}{|l}
AllDisciplinesPrinciples, PMPrinciples : \mathbb{P}\, Attribute \\
STPrinciples, MMPrinciples : \mathbb{P}\, Attribute \\
SVPrinciples, LTPrinciples : \mathbb{P}\, Attribute \\
\hline
PMPrinciples \neq \varnothing \\
STPrinciples \neq \varnothing \\
MMPrinciples \neq \varnothing \\
SVPrinciples \neq \varnothing \\
LTPrinciples \neq \varnothing \\
AllDisciplinesPrinciples = PMPrinciples \cup STPrinciples \\
\quad \cup MMPrinciples \cup SVPrinciples \cup LTPrinciples
\end{array}
$$

The set of actions and principles of the five disciplines is specified as follows.

$$
\begin{array}{|l}
\textit{DisciplinesActionsAndPrinciples} \\
\hline
actions : \mathbb{P}\, Action \\
principles : \mathbb{P}\, Attribute \\
\hline
actions \subseteq AllDisciplinesActions \\
principles \subseteq AllDisciplinesPrinciples \\
actions \neq \varnothing \\
principles \neq \varnothing \\
\hline
\end{array}
$$

Below, we specify a function that formalizes that the development of the disciplines by an agent involves the instantiation of a new entity, which is capable of developing the five disciplines. The function takes an entity and the set of actions and principles of the disciplines, and creates a new entity with the same values for each variable defined in the former entity, but with new capabilities and additional principles, related to the principles, in its store.

$$
\begin{array}{|l}
entityLearnDisciplinesActionsAndPrinciples : (Entity \\
\quad \times DisciplinesActionsAndPrinciples) \rightarrow Entity \\
\hline
\forall\, old, new : Entity;\ actionAndPrinciple : \\
\quad DisciplinesActionsAndPrinciples \bullet \\
\quad (entityLearnDisciplinesActionsAndPrinciples(old, \\
\quad actionAndPrinciple) = new \Leftrightarrow \\
\quad (new.goals = old.goals \land \\
\qquad new.attributes = old.attributes \land \\
\qquad new.motivations = old.motivations \land \\
\qquad new.ownedresources = old.ownedresources \land \\
\qquad new.instsreq = old.instsreq \land \\
\qquad new.resourcesofplan = old.resourcesofplan \land \\
\qquad new.plans = old.plans \land \\
\qquad new.planforgoal = old.planforgoal \land \\
\qquad new.orgs = old.orgs \land \\
\qquad new.roles = old.roles \land \\
\qquad new.allplans = old.allplans \land \\
\qquad new.allgoals = old.allgoals \land \\
\qquad new.capableof = old.capableof \cup \\
\qquad\quad actionAndPrinciple.actions \land \\
\qquad new.store = old.store \cup \\
\qquad actionAndPrinciple.principles \land \\
\qquad actionAndPrinciple.principles \neq \varnothing \land \\
\qquad actionAndPrinciple.actions \neq \varnothing)) \\
\end{array}
$$

3.2.2 Learning Organization Agent

Finally, we define the *LearningOrgAgent* type. The Learning Organization Agent is capable of developing the disciplines defined by Senge.

LearningOrgAgent

OrgAgent

$personalMasteryCapabilities \neq \varnothing$
$mentalModelsCapabilities \neq \varnothing$
$systemsThinkingCapabilities \neq \varnothing$
$learningTeamCapabilities \neq \varnothing$
$buildingSharedVisionCapabilities \neq \varnothing$
$personalMasteryCapabilities \subset capableof$
$mentalModelsCapabilities \subset capableof$
$systemsThinkingCapabilities \subset capableof$
$learningTeamCapabilities \subset capableof$
$buildingSharedVisionCapabilities \subset capableof$

3.2.3 Mental Model

Mental models include reflection and inquiry skills, so as to, first, make the agent aware of the hypotheses and generalizations that influence both the agent's comprehension of and interaction with the world; and, second, interact with other agents to share personal visions and learn each others assumptions on several subjects.

Initially, we introduce a formalization of beliefs that follows [dL01]:

$$
\begin{aligned}
AGLiteral &::= pos\langle\langle Atom^{3}\rangle\rangle \\
&\mid\ negate\langle\langle Atom\rangle\rangle \\
AGBelief &::= agliteral\langle\langle AGLiteral\rangle\rangle \\
&\mid\ conjunct\langle\langle AGBelief \times AGBelief\rangle\rangle \\
AgentBelief &::= agbeliefs\langle\langle AGBelief\rangle\rangle \\
ExposedBelief &::= expbeliefs\langle\langle AGBelief\rangle\rangle
\end{aligned}
$$

The reasoning process corresponds to the exhibition of the "ladder of inference - a common mental pathway of increasing abstraction, often leading to misguided beliefs" [4], as mentioned by Senge.

In addition, in [dL01] the modeling capabilities of agents are represented in a series of schemes that depict the fact that an agent is capable of modeling entities, objects, agents, etc. In this work, we define that a Learning Organization Agent is capable of modeling all these entities.

First, we follow [dL01] and define representations of models of entities, objects, agents, neutral objects, server agents, and store agents, and include models for *SAutoAgent*, *PlanningAgent*, and *OrgAgent*.

[3] The type *Atom* contains predicates and terms of first order logic. Please, refer to [dL01] for its definition.

[4] In [SKR+94, p. 243]

$$ReasoningProcess == \text{seq}_1\ AGBelief$$

$$ReflectedBeliefReasoning == AGBelief \times ReasoningProcess$$

$$ExposedBeliefReasoning == AGBelief \times ReasoningProcess$$

$$BeliefAndReasoning == AGBelief \times ReasoningProcess$$

$$EntityModel == Entity$$

$$ObjectModel == Object$$

$$NeutralObjectModel == NeutralObject$$

$$AgentModel == Agent$$

$$ServerAgentModel == ServerAgent$$

$$StoreAgentModel == StoreAgent$$

$$SAutoAgentModel == SAutoAgent$$

$$PlanningAgentModel == PlanningAgent$$

$$OrgAgentModel == OrgAgent$$

The mental models must also include the relationships among the various components present in the environment.

$$\begin{aligned} Component ::= & \ compEntity\langle\langle Entity\rangle\rangle \\ | & \ compObject\langle\langle Object\rangle\rangle \\ | & \ compNeutralObject\langle\langle NeutralObject\rangle\rangle \\ | & \ compAgent\langle\langle Agent\rangle\rangle \\ | & \ compServerAgent\langle\langle ServerAgent\rangle\rangle \\ | & \ compStoreAgent\langle\langle StoreAgent\rangle\rangle \\ | & \ compSAutoAgent\langle\langle SAutoAgent\rangle\rangle \\ | & \ compPlanningAgent\langle\langle PlanningAgent\rangle\rangle \\ | & \ compOrgAgent\langle\langle OrgAgent\rangle\rangle \end{aligned}$$

$$ComponentRelationship ::= compRelationship\langle\langle Component \times Component\rangle\rangle$$

$$ComponentRelationshipModel == ComponentRelationship$$

The *BelModel* schema presents the specification of components present in the environment and also the relationships involving them.

BelModel

$models : \mathbb{P}\ EntityModel$
$agentbeliefs : \mathbb{P}\ AGBelief$
$modelentities : \mathbb{P}\ EntityModel$
$modelobjects : \mathbb{P}\ ObjectModel$
$modelneutralobjects : \mathbb{P}\ NeutralObjectModel$
$modelagents : \mathbb{P}\ AgentModel$
$modelserveragents : \mathbb{P}\ ServerAgentModel$
$modelstoreagents : \mathbb{P}\ StoreAgentModel$
$modelsautoagents : \mathbb{P}\ SAutoAgentModel$
$modelplanningagents : \mathbb{P}\ PlanningAgentModel$
$modelorgagents : \mathbb{P}\ OrgAgentModel$
$modelcomprelationship : \mathbb{P}\ ComponentRelationshipModel$

$models = \bigcup\{modelentities, modelobjects,$
$\quad modelneutralobjects, modelagents, modelserveragents,$
$\quad modelstoreagents, modelsautoagents,$
$\quad modelplanningagents, modelorgagents\}$
$\#models > 1 \Rightarrow modelcomprelationship \neq \emptyset$
$modelsautoagents \subseteq modelstoreagents \subseteq$
$\quad modelagents \subseteq modelobjects \subseteq modelentities$
$modelobjects = modelagents \cup modelneutralobjects$
$modelserveragents = modelagents \setminus modelsautoagents$

Hence, the *IntraPersonalMentalModel* schema specifies the agent's mental models that do not depend on interactions among agents of type *LearningOrgAgent*.

IntraPersonalMentalModel

$agentmodels : BelModel$
$developReflectionSkillsActions : \mathbb{P}\ Action$
$developAdvocacySkillsActions : \mathbb{P}\ Action$
$intraPersonalMentalModelPrinciples : \mathbb{P}\ Attribute$
$reflection : \mathbb{P}\ Motivation \times \mathbb{P}\ Action \times$
$\quad BeliefAndReasoning \nrightarrow ReflectedBeliefReasoning$
$advocacy : \mathbb{P}\ Motivation \times \mathbb{P}\ Action \times$
$\quad BeliefAndReasoning \nrightarrow ExposedBeliefReasoning$

$developReflectionSkillsActions \cup$
$\quad developAdvocacySkillsActions = MMActions$

The *InterPersonalMentalModel* schema specifies the agent's mental models that depend on interactions among agents of type *LearningOrgAgent*. *The agent must inquiry other agents in order to learn their mental models*.

InterPersonalMentalModel

$inquiry : LearningOrgAgent \nrightarrow ExposedBelief$

In the same way that other models of components were specified, we define that a model of a *LearningOrgAgent* corresponds to an abbreviation to the type *LearningOrgAgent*. In addition, we extend both the representation of the components in the environment to include the *LearningOrgAgent* and the possible relationships among the various components present in the environment.

$$LearningOrgAgentModel == LearningOrgAgent$$

$$LearnComponent ::= compLearningOrgAgent\langle\langle LearningOrgAgent\rangle\rangle \mid compGeneral\langle\langle Component\rangle\rangle$$

$$LearnComponentRelationship ::= compLearnRelationship \langle\langle LearnComponent \times LearnComponent\rangle\rangle$$

$$LearnComponentRelationshipModel == LearnComponentRelationship$$

$$[LOrgAgentInteractions, TrustLOrgAgentInter, NonTrustLOrgAgentInter]$$

$$LOrgAgentInteractionsModel == LOrgAgentInteractions$$

$$TrustLOrgAgentInterModel == TrustLOrgAgentInter$$

$$NonTrustLOrgAgentInterModel == NonTrustLOrgAgentInter$$

The *LearnBelModel* schema corresponds to the specification of components present in the environment and also the relationships involving them. Moreover, we have inserted in this schema the *modelbelmodel* function which is similar to the *modelsociologicalagents* function presented in [dL01, p. 106]. This function represents the agent's capability to model the models of other agents.

LearnBelModel

$$BelModel$$
$$modellearningorgagents : \mathbb{P}_1\ LearningOrgAgentModel$$
$$modelbelmodel : Agent \pfun LearningOrgAgent$$
$$modellearncomprelationship : \mathbb{P}\ LearnComponentRelationshipModel$$
$$modelLOrgAgInts : \mathbb{P}\ LOrgAgentInteractionsModel$$
$$modelTrustLOrgAgInts : \mathbb{P}\ TrustLOrgAgentInterModel$$
$$modelNonTrustLOrgAgInts : \mathbb{P}\ NonTrustLOrgAgentInterModel$$

$$models = modelentities \cup modelobjects \cup modelneutralobjects \cup modelagents \cup modelserveragents \cup modelstoreagents \cup modelsautoagents \cup modelplanningagents \cup modelorgagents \cup modellearningorgagents$$
$$\#models > 1 \Rightarrow modellearncomprelationship \neq \varnothing$$
$$\mathrm{dom}\ modelbelmodel \subseteq modelagents$$

Thus, in *PeerLOAgMentalModel* we have the specification of all models that a *LearningOrgAgent* may have, including models of peer agents of the same type.

PeerLOAgMentalModel
LearnBelModel
InterPersonalMentalModel

Finally, the mental model includes all types of mental models that the agent may have.

MentalModel
IntraPersonalMentalModel
PeerLOAgMentalModel

3.2.4 Personal Mastery

Personal mastery is a process that is based on two factors: a continuous and clear evaluation of reality, and a personal vision. The vision, on its turn, corresponds to individual goals which represent an image of the future that the agent wants to produce. This process generates an internal force in the agent, named creative tension according to Senge. The resolution of this tension means: to make the image of reality get closer to the personal vision [Sen90]. In addition, the behavior of this agent is also guided by its values. Values are "deeply held views of what we find worthwhile"[5]. Some examples of personal values include honesty, reputation, status, and loyalty[6]. Moreover, Senge states that a vision can only be understood in association to a purpose. Purpose corresponds to a "direction, a general heading"[7].

In this formalization, we consider that both vision (*Vision*) and purpose (*Purpose*) correspond to sets of goals, and that values (*Value*) corresponds to a set of *BehavioralConstraint*, that we specify in this work as a set of beliefs. Additionally, we also define the type *RealityVision* as an abbreviation for a set of *View*.

$$RealityVision == \mathbb{P}\ View$$

$$Consistency ::= yes \mid no$$

$$BehavioralConstraint == \mathbb{P}\ AGBelief$$

$$Vision \mathrel{\widehat=} [visions : \mathbb{P}\ Goal]$$

$$Purpose \mathrel{\widehat=} [purposes : \mathbb{P}\ Goal]$$

$$Value \mathrel{\widehat=} [values : \mathbb{P}\ BehavioralConstraint]$$

According to Senge, guiding (or governing) ideas correspond to a set that includes vision, purpose and values. In our model we define both individual and collective guiding ideas. The elements of guiding ideas must be consistent. Thus, in this work, we define the functions *isVisionPurposeConsistent*,

[5] [SKR+94, p. 209].
[6] [SKR+94, p. 210].
[7] [Sen90, p. 148].

isValueVisionConsistent, and *isValuePurposeConsistent* that map to a *Consistency* value of *yes*, if, respectively, the formulae set corresponding to vision and purpose are logically consistent; no formula in the vision formulae set contradicts any formulae in the value set; and no formula in the purpose formulae set contradicts any formulae in the value set.

$$
\begin{array}{l}
isVisionPurposeConsistent : (\mathbb{P}\ Vision \times \mathbb{P}\ Purpose) \rightarrow Consistency \\
isValueVisionConsistent : (\mathbb{P}\ Value \times \mathbb{P}\ Vision) \rightarrow Consistency \\
isValuePurposeConsistent : (\mathbb{P}\ Value \times \mathbb{P}\ Purpose) \rightarrow Consistency
\end{array}
$$

The *GuidingIdea* schema includes consistent vision, purpose and values.

GuidingIdea

$$
\begin{array}{l}
vision : \mathbb{P}\ Vision \\
purpose : \mathbb{P}\ Purpose \\
value : \mathbb{P}\ Value \\
\hline
isVisionPurposeConsistent(vision, purpose) = yes \land \\
\quad isValueVisionConsistent(value, vision) = yes \land \\
\quad isValuePurposeConsistent(value, purpose) = yes
\end{array}
$$

The agent's personal vision corresponds to a set of goals that is influenced by its motivations, goals related to its roles, mental models, and guiding ideas.

$$
\begin{array}{l}
PersonalVision \mathrel{\widehat{=}} [personalvision : \mathbb{P}\ Motivation \nrightarrow \mathbb{P}\ Goal \nrightarrow \\
\quad \mathbb{P}\ IntraPersonalMentalModel \nrightarrow \mathbb{P}\ GuidingIdea \nrightarrow \mathbb{P}\ Goal]
\end{array}
$$

In summary, the personal mastery discipline involves a set of actions and principles that aim at developing personal visions (*clarifypersonalvisions*), guiding ideas (*developguidingideas*), and a clear vision of the current reality (*enhancerealityvisions*). In addition, the creative tension (*creativetension*) has to produce goals (*ResolutionGoal*) with the objective of reducing such tension.

$$ResolutionGoal == \mathbb{P}\ Goal$$

PersonalMastery

$developguidingideasactions : \mathbb{P}\, Action$
$enhancerealityvisionsactions : \mathbb{P}\, Action$
$clarifypersonalvisionsactions : \mathbb{P}\, Action$
$personalMasteryPrinciples : \mathbb{P}\, Attribute$
$developguidingideas : \mathbb{P}\, Motivation \times \mathbb{P}\, Action \pfun GuidingIdea$
$enhancerealityvisions : \mathbb{P}\, Motivation \times \mathbb{P}\, Action \pfun \mathbb{P}\, RealityVision$
$clarifypersonalvisions : \mathbb{P}\, Motivation \times \mathbb{P}\, Action \pfun \mathbb{P}\, PersonalVision$
$creativetension : \mathbb{P}\, View \times \mathbb{P}\, PersonalVision \pfun \mathbb{P}\, ResolutionGoal$
$personalvisions : \mathbb{P}\, PersonalVision$
$guidingidea : GuidingIdea$

$developguidingideasactions \cup enhancerealityvisionsactions \cup clarifypersonalvisionsactions = PMActions$

3.2.5 Team Learning

The goal of this discipline is to develop learning skills in a team, so that the team develops uncommon capabilities for the coordinated action, producing extraordinary results and faster development of its members than it would be possible in other circumstances.

This discipline involves the development of the dialog and discussion techniques. Dialog concerns the capability of the members of a team to suppress their individual assumptions and enter an authentic state of shared thought. On the other hand, discussion involves the presentation of the members' particular viewpoints (in a given context) with the goal of producing a resulting goal or decision [Sen90].

In this work, we specify that dialog, discussion, inquiry, and advocacy correspond to protocols that we introduce as a given set, as the detailed definition of each one of these is out of the scope of this work. We also define that there are actions associated with each protocol that an agent must be capable of performing and that the development of dialog and inquiry actions by a given agent is also a function of its motivations.

$[Protocol]$

TeamLearning

$$
\begin{array}{l}
dialogdiscussactions, inquiryadvocacyactions : \mathbb{P}\, Action \\
teamLearningPrinciples : \mathbb{P}\, Attribute \\
dialogprotocols, discussionprotocols : \mathbb{P}\, Protocol \\
inquiryprotocols, advocacyprotocols : \mathbb{P}\, Protocol \\
actionsofdialogprotocols : Protocol \nrightarrow \mathbb{P}\, Action \\
actionsofdiscussionprotocols : Protocol \nrightarrow \mathbb{P}\, Action \\
actionsofinquiryprotocols : Protocol \nrightarrow \mathbb{P}\, Action \\
actionsofadvocacyprotocols : Protocol \nrightarrow \mathbb{P}\, Action \\
developdialogdiscussactions : \mathbb{P}\, Motivation \times \mathbb{P}\, Action \nrightarrow \mathbb{P}\, Action \\
developinquiryadvocacyactions : \mathbb{P}\, Motivation \times \mathbb{P}\, Action \nrightarrow \mathbb{P}\, Action \\
\hline
dialogdiscussactions \cup inquiryadvocacyactions = LTActions
\end{array}
$$

3.2.6 Systems Thinking

Systems thinking reveals a variety of potential actions that produce the desired results and also, unintended consequences. It involves the recognition of four levels operating simultaneously: events, patterns of behavior, systemic structures and mental models [SKR+94].

In order to build a model for this discipline we first introduce a few types as given sets, related to a time instant, a position in space, links, events and computational models.

$$[Time, Space, Links, Event, CompModel]$$

In this work, *Context* refers to information related to time, location and a set of entities involved in some modification occurred in the environment.

$$Context == Time \times Space \times \mathbb{P}\, Entity$$

Behavioral patterns (*BehavioralPattern*) and scenarios (*Scenario*) are both defined as sequences of events (*Event*). Finally, we also define a generic structure library (*GenericStructure*) that includes: causal loops and archetypes. The loops and archetypes declared below correspond to the several types presented by Senge in [Sen90].

$$BehavioralPattern == \mathrm{seq}_1\, Event$$

$$Scenario == \mathrm{seq}_1\, Event$$

$$Loops ::= ReinforcingLoops \mid BalancingLoops \mid DelayedReinforcingLoops \mid DelayedBalancingLoops$$

$$SystemsArchetypes ::= Fixes\_that\_backfire \mid Limits\_to\_growth \mid Shifting\_the\_burden \mid Tragedy\_of\_the\_commons \mid Accidental\_adversaries$$

$$GenericStructure ::= links\langle\langle Links\rangle\rangle \mid loops\langle\langle Loops\rangle\rangle \mid systemsarchetypes\langle\langle SystemsArchetypes\rangle\rangle$$

This discipline also involves simulations based on computational models. We assume in this model that there is a library of computational models (*CompModelLib*) that the agents can query and update. In this library are stored records that relate a particular computational model, behavioral patterns and events. Hence, based on behavioral patterns, an agent might try to select a suitable model to perform simulations. As a result of such simulations some potential states (*PotentialState*) in the environment can be investigated. In addtion, we also define in our model a plan library (*PlanLib*) that agents can read and update. These plans are associated with the production of a number of the potential states.

$$PotentialState == \mathbb{P}_1\, Attribute$$

$$PlanLib == \mathbb{P}\, Plan$$

$$CompModelLib == \mathbb{P}(\mathbb{P}\, CompModel \times \mathbb{P}\, BehavioralPattern \times \mathbb{P}\, Event)$$

The *SystemsThinking* schema includes a library of generic structures and a set of inference engines, possibly specialized, which allow the agent to reason about these structures. In the schema, *planlib* corresponds to a library of plans available to the agent. In addition, we include a library of computational models that is accessible to all agents, a set of events, and several functions.

The *contextassessment* function maps the agent's perceptions to contexts. These contexts define for a given event: time, space, and the associated entities related to this percept. The *eventanalyer* function maps contexts to events. The *behavioralpatternanalyser* function maps a set of events to a behavioral pattern. The *compmodelsdesign* function represents the agent's capability to design a computational model that corresponds to its interpretation of the relationship involving a set of patterns of behavior and a set of events, using a library of generic structures and a library of computational models to help it in this process. The computational model produced is then used in simulations in different scenarios, resulting in a set of potential states, as specified in the simulation function. Finally, considering the agent's goals, the potential state that is most likely to reduce the agent's creative tension is then selected, and a set of plans designed to achieve the desired potential state is produced, based on information contained in the plan library. The restrictions associated to all these variables are introduced in the $\Delta LearningOrgAgentState$ schema so that they are defined in the scope of the type *LearningOrgAgent*.

SystemsThinking

$events : \mathbb{P}\, Event$
$behavioralpatterns : \mathbb{P}\, BehavioralPattern$
$compmodels : \mathbb{P}\, CompModel$
$simulatedpotentialstates : \mathbb{P}\, PotentialState$
$selectedpotentialstates : \mathbb{P}\, PotentialState$
$scenarios : \mathbb{P}\, Scenario$
$planlib : PlanLib$
$compmodellib : CompModelLib$
$genericstructures : \mathbb{P}\, GenericStructure$
$contextassessment : View \pfun \mathbb{P}\, Context$
$eventanalyser : \mathbb{P}\, Context \pfun \mathbb{P}\, Event$
$behavioralpatternanalyser : \mathbb{P}\, Event \pfun \mathbb{P}\, BehavioralPattern$
$compmodelsdesign : \mathbb{P}\, BehavioralPattern \times \mathbb{P}\, Event \times$
$\quad \mathbb{P}\, GenericStructure \times CompModelLib \pfun \mathbb{P}\, CompModel$
$behavioralpatternfromcompmodel : \mathbb{P}\, CompModel \pfun$
$\quad \mathbb{P}\, BehavioralPattern$
$simulation : \mathbb{P}\, CompModel \times \mathbb{P}\, Scenario \pfun \mathbb{P}\, PotentialState$
$potentialstatesanalyser : \mathbb{P}\, PersonalVision \times$
$\quad \mathbb{P}\, PotentialState \pfun \mathbb{P}\, PotentialState$
$planfrompotentialstate : \mathbb{P}\, PotentialState \times \mathbb{P}\, Plan \pfun \mathbb{P}\, Plan$
$modelscenariofrompotstate : \mathbb{P}\, PotentialState \pfun$
$\quad \mathbb{P}\, CompModel \times \mathbb{P}\, Scenario \times \mathbb{P}\, PotentialState$
$resultingplans : \mathbb{P}\, Plan$

The $\Delta SystemsThinking$ schema presents an operation on the *SystemsThinking* schema and shows that the introduction of new perceptions (*systemsactualpercepts*?) produces a new set of resulting plans (*resultingplans*!). Moreover, the library of computational models is updated accordingly. We assume, also, that there may be different scenarios to be considered.

$\Delta SystemsThinking$

$\Delta LearningOrgAgentState$
$SystemsThinking$
$SystemsThinking'$
$resultingplans! : \mathbb{P}\ Plan$
$systemsactualpercepts? : View$
$modelscenariostate : \mathbb{P}\ CompModel \times$
$\quad \mathbb{P}\ Scenario \times \mathbb{P}\ PotentialState$
$compmodeltoadd : \mathbb{P}\ CompModel$
$behavioralpatterntoadd : \mathbb{P}\ BehavioralPattern$

$genericstructures' = genericstructures$
$planlib' = planlib$
$events' = eventanalyser(contextassessment$
$\quad (systemsactualpercepts?))$
$behavioralpatterns' = behavioralpatternanalyser(events')$
$compmodels' = compmodelsdesign(behavioralpatterns',$
$\quad events', genericstructures, compmodellib)$
$simulatedpotentialstates' = simulation(compmodels',$
$\quad scenarios')$
$selectedpotentialstates' = potentialstatesanalyser$
$\quad (personalmastery.personalvisions,$
$\quad simulatedpotentialstates')$
$modelscenariostate = modelscenariofrompotstate$
$\quad selectedpotentialstates'$
$compmodeltoadd = trifirst\ modelscenariostate$
$behavioralpatterntoadd = behavioralpatternfromcompmodel$
$\quad compmodeltoadd$
$compmodellib' = compmodellib \cup \{(compmodeltoadd,$
$\quad behavioralpatterntoadd, events')\}$
$resultingplans! = planfrompotentialstate$
$\quad (selectedpotentialstates', planlib)$

We abstract the disciplines presented so far, which mainly relate to the agent's internal states and representations, in the schema *IntraPersonalDisciplines*. Later, we group the remaining disciplines, shared vision and team learning, in the *InterPersonalDisciplines* schema.

$IntraPersonalDisciplines$

$PersonalMastery$
$SystemsThinking$
$IntraPersonalMentalModel$

3.2.7 Learning Team Development

In this section, we specify the process of the development of a team into a learning team. The process is based on the type *TLearningTeam*, which is a team where common plans and goals are developed by agents of type *LearningOrgAgent*.

TLearningTeam

$$Team$$

$$\text{dom}\, developcommonplans = \{learningmembers\}$$

The *TLearningTeamIni* schema represents the initial state of the *TLearningTeam*, with empty common plans and team guiding ideas.

TLearningTeamIni

$$\Delta\, TLearningTeam$$

$$developcommonplans' = \varnothing$$
$$teamguidingideas'.visions = \varnothing$$
$$teamguidingideas'.purposes = \varnothing$$
$$teamguidingideas'.values = \varnothing$$
$$learningmembers' = \varnothing$$
$$commonplans' = \varnothing$$

In fact, we assume in this model that the team development process involves the individual development of each team member into an agent of type *LearningOrgAgent*. This assumption is based on the fact that, in our model, only this type of agent is required to develop the actions and know the principles related to the team learning discipline. Therefore, when all agent members become *LearningOrgAgent* agents the team is fully developed as specified in the *TLearningTeamDeveloped* schema.

TLearningTeamDeveloped

$$TLearningTeam$$

$$membersgroup = learningmembers$$

In the *TLearningTeamMembersChange* schema we specify that a set of agents of type *PlanningAgent* (*pas*?), which is a subset of the members of the team as defined for a *TLearningTeam*, develop new capabilities and learn the principles related to Senge's disciplines. This process, defined by the *entityLearnDisciplinesActionsAndPrinciples* function, causes the instantiation of a new *LearningOrgAgent* for each corresponding *PlanningAgent*. Each new *LearningOrgAgent* becomes a member of the *learningmembers* set of the *TLearningTeam*.

$$
\begin{array}{|l}
\hline
\textit{TLearningTeamMembersChange} \\
\Delta\, \textit{TLearningTeam} \\
\textit{pas}? : \mathbb{P}\, \textit{PlanningAgent} \\
\textit{actionsAndPrinciples}? : \mathbb{P}\, \textit{DisciplinesActionsAndPrinciples} \\
\hline
\textit{resources}' = \textit{resources} \\
\textit{commongoals}' = \textit{commongoals} \\
\textit{developcommonplans}' = \textit{developcommonplans} \\
\textit{teamguidingideas}' = \textit{teamguidingideas} \\
\textit{commonplans}' = \textit{commonplans} \\
\textit{actionsAndPrinciples}? \neq \varnothing \\
\#\textit{actionsAndPrinciples}? = \#\textit{pas}? \\
\textit{pas}? \neq \varnothing \\
\textit{pas}? \subseteq \textit{membersgroup} \\
\forall\, \textit{pa} : \textit{LearningOrgAgent} \mid \textit{pa} \in \textit{pas}? \bullet \\
\quad (\exists_1\, \textit{actsAndPrinciples} : \textit{DisciplinesActionsAndPrinciples} \mid \\
\quad \textit{actsAndPrinciples} \in \textit{actionsAndPrinciples}? \bullet \\
\qquad \textit{learningmembers}' = \textit{learningmembers} \cup \\
\qquad \{\textit{entityLearnDisciplinesActionsAndPrinciples}(\textit{pa}, \\
\qquad\quad \textit{actsAndPrinciples})\}) \\
\textit{membersgroup}' = \textit{membersgroup} \\
\hline
\end{array}
$$

The relation specified below associates two *TLearningTeam* states, before and after the operation *TLearningTeamMembersChange*.

$$
\begin{array}{l}
\textit{RelationTLearningTeam} == \mathbb{P}\{\, \textit{TLearningTeamMembersChange} \bullet \\
\quad (\theta\, \textit{TLearningTeam}, \theta\, \textit{TLearningTeam}')\}
\end{array}
$$

Based on the *RelationTLearningTeam* relation we can define a history of state changes in a team. This history shows that, at each state, a certain number of agents of type *LearningOrgAgent* are members of the team. At the beginning of this history, the state of the *TLearningTeam* corresponds to the *TLearningTeamIni* state. This constraint is imposed by the first predicate in *histTLearningTeamDevelopment*, which defines that the first element in the history sequence must be the one represented by the *TLearningTeam* initial state. At the end of the history the state of the *TLearningTeam* corresponds to the *TLearningTeamDeveloped* state. The third predicate in *histTLearningTeamDevelopment* defines that in this history two contiguous states are related by the *RelationTLearningTeam*. We assume, as a simplification hypothesis, that the length of the history is equal to the number of team members, so that the end of the history occurs when all agents are of type *LearningOrgAgent*.

opTLearningTeam

$TLearningTeam$
$nextChangeLTeamState : RelationTLearningTeam$

$TLearningTeam \in \{\text{dom}\, nextChangeLTeamState\}$

histTLearningTeamDevelopment

$hist : \text{seq}\, opTLearningTeam$

$\exists\, teamIni : TLearningTeam \bullet (\exists\, opLTeam :$
$\quad opTLearningTeam \mid opLTeam = head(\{1\} \upharpoonright hist) \bullet$
$\quad (teamIni = (\lambda\, opTLearningTeam \bullet$
$\quad \theta\, TLearningTeam)\, opLTeam \wedge$
$\quad teamIni \in \{\, TLearningTeamIni\}))$
$\exists\, teamFin : TLearningTeam \bullet (\exists\, opLTeam : opTLearningTeam \mid$
$\quad opLTeam = head(\{\#hist\} \upharpoonright hist) \bullet$
$\quad (teamFin = (\lambda\, opTLearningTeam \bullet$
$\quad \theta\, TLearningTeam)\, opLTeam \wedge$
$\quad teamFin \in \{\, TLearningTeamDeveloped\}))$
$\forall\, i : \mathbb{N}_1 \mid i \in \text{dom}\, hist \setminus \{1\} \bullet$
$\quad (\exists\, lteamStateBefore, lteamStateAfter : TLearningTeam;$
$\quad lTeamRelation : RelationTLearningTeam \bullet$
$\quad (\exists\, opLTeamBefore, opLTeamAfter : opTLearningTeam \mid$
$\quad opLTeamBefore = head(\{i-1\} \upharpoonright hist) \wedge$
$\quad opLTeamAfter = head(\{i\} \upharpoonright hist) \wedge$
$\quad lTeamRelation = opLTeamBefore.nextChangeLTeamState \bullet$
$\quad (lteamStateBefore = (\lambda\, opTLearningTeam \bullet$
$\quad \theta\, TLearningTeam)\, opLTeamBefore \wedge$
$\quad lteamStateAfter = (\lambda\, opTLearningTeam \bullet$
$\quad\quad \theta\, TLearningTeam)\, opLTeamAfter \wedge$
$\quad lteamStateBefore\ \underline{lTeamRelation}\ lteamStateAfter)))$

The process evolves: at each new *TLearningTeam* state there are more agent members of type *LearningOrgAgent*. This is true for all states except for the final state when all members are learning members.

histTLearningTeamDevelopmentProgress

$histTLearningTeamDevelopment$
$lTeamRelation : RelationTLearningTeam$

$\forall\, i, j : \mathbb{N}_1 \mid i \in \text{dom}\, hist \wedge j \in \text{dom}\, hist \wedge j = i + 1 \bullet$
$\quad (\exists\, lteamStateBefore, lteamStateAfter : TLearningTeam \mid$
$\quad lteamStateBefore\ \underline{lTeamRelation}\ lteamStateAfter \bullet$
$\quad (\#(lteamStateBefore.learningmembers) <$
$\quad \#(lteamStateAfter.learningmembers) \wedge$
$\quad \#hist \leq \#lteamStateAfter.membersgroup))$

We consider in this model that the learning team must develop its guiding ideas. A *LearningOrgAgent* adapts his guiding ideas to the ones that emerge from the process of dialog and discussion. These ideas must be consistent with the values, purposes and vision maintained by the agent before the process of adaptation. The resulting guiding ideas become the agent's guiding ideas.

$$
\begin{array}{l}
agentAdaptGuidingIdeas : (LearningOrgAgent \times \\
\quad GuidingIdea) \rightarrow GuidingIdea \\
\hline
\forall\, la : LearningOrgAgent;\ gi : GuidingIdea \bullet (\exists_1\, lagBefore, \\
\quad lagAfter : GuidingIdea \mid lagBefore = la.ownguidingideas \bullet \\
\quad (agentAdaptGuidingIdeas(la, gi) = lagAfter \Leftrightarrow \\
\quad (isVisionPurposeConsistent(lagBefore.visions, gi.purposes) = yes \\
\qquad \wedge\ isValueVisionConsistent(lagBefore.values, gi.visions) = yes \\
\qquad \wedge\ isValuePurposeConsistent(lagBefore.values, gi.purposes) = \\
\qquad\quad yes \wedge isVisionPurposeConsistent(lagBefore.visions, \\
\qquad\quad lagAfter.purposes) = yes \wedge \\
\qquad isValueVisionConsistent(lagBefore.values, \\
\qquad\quad lagAfter.visions) = yes \wedge \\
\qquad isValuePurposeConsistent(lagBefore.values, \\
\qquad\quad lagAfter.purposes) = yes \wedge \\
\qquad isVisionPurposeConsistent(lagAfter.visions, \\
\qquad\quad gi.purposes) = yes \wedge \\
\qquad isValueVisionConsistent(lagAfter.values, gi.visions) = yes \wedge \\
\qquad isValuePurposeConsistent(lagAfter.values, gi.purposes) \\
\qquad\quad = yes \wedge la.ownguidingideas = lagAfter)))
\end{array}
$$

Here, we consider that a group's guiding ideas have been produced if all agents members have adapted their own guiding ideas during the process, and if at the end of this process, the shared guiding ideas become a subset of the adapted guiding ideas of each participant agent. Thus, the shared guiding ideas are consistent with the guiding ideas of each agent member.

$$
\begin{array}{l}
dialogdiscuss : \mathbb{P}\, LearningOrgAgent \rightarrow GuidingIdea \\
\hline
\forall\, las : \mathbb{P}\, LearningOrgAgent;\ gi : GuidingIdea \bullet \\
\quad (\forall\, la : LearningOrgAgent \mid la \in las \bullet \\
\quad (\exists_1\, lag : GuidingIdea \bullet (dialogdiscuss\ (las) = gi \Leftrightarrow \\
\quad (agentAdaptGuidingIdeas(la, gi) = lag \wedge \\
\qquad la.ownguidingideas = lag \wedge gi.visions \subseteq lag.visions \wedge \\
\qquad gi.purposes \subseteq lag.purposes \wedge gi.values \subseteq lag.values))))
\end{array}
$$

Finally, in a *LearningTeam*, all members are of type *LearningOrgAgent*. The team guiding ideas (*teamguidingideas*) result from a process involving dialog and discussion (*dialogdiscuss*). We also define that there are resources and goals that are shared among the team members (*learningmembers*).

LearningTeam

TLearningTeamDeveloped

$$
\begin{array}{l}
teamguidingideas = dialogdiscuss\,(learningmembers) \\
\exists\, r : Resource \mid r \in resources \bullet (\forall\, a : LearningOrgAgent \mid \\
\quad a \in learningmembers \bullet (\exists\, p : Plan \mid \\
\quad p \in a.plans \bullet r \in \{a.resourcesofplan\ p\})) \\
\forall\, g : Goal \mid g \in commongoals \bullet (\forall\, a : LearningOrgAgent \mid \\
\quad a \in learningmembers \bullet g \in a.goals)
\end{array}
$$

3.2.8 Shared Vision

As mentioned above, team guiding ideas result from *dialogdiscuss* processes. Similarly, the shared vision of a set of teams results from *dialogdiscuss* processes involving all these teams. The basic assumption here is that for every team in the organization, at least one of its members is also a member of another team[8]. Therefore, *dialogdiscuss* processes should allow the development of shared guiding ideas that reflect the influence of all teams in a particular organization. It is also important to note that guiding ideas are composed of visions, values and purposes. Hence, in the *SharedVision* schema the set of shared visions (*sharedvisions*) is obtained from the *visionsFromGuidingIdea* function. This function, in its turn, extracts shared visions from the guiding ideas associated with *interTeamsGuidingIdeas*.

SharedVision

$$
\begin{array}{l}
learningteams : \mathbb{P}_1\, LearningTeam \\
interTeamsGuidingIdeas : \mathbb{P}\, LearningTeam \rightarrow GuidingIdea \\
sharedvisions : \mathbb{P}\, Vision \\
visionsFromGuidingIdea : \mathbb{P}\, GuidingIdea \rightarrow \mathbb{P}\, Vision \\
\hline
sharedvisions = visionsFromGuidingIdea(\text{ran}\ interTeamsGuidingIdeas) \\
\forall\, gt : \mathbb{P}\, learningteams;\ gi : GuidingIdea \bullet (\forall\, t_1, t_2 : LearningTeam \mid \\
\quad t_1 \in gt \wedge t_2 \in gt \bullet (\forall\, la_1, la_2 : LearningOrgAgent \mid \\
\quad la_1 \in t_1.learningmembers \wedge la_2 \in t_2.learningmembers \wedge \\
\quad (la_1 \in t_1.learningmembers \cap t_2.learningmembers \vee \\
\qquad la_2 \in t_1.learningmembers \cap t_2.learningmembers) \bullet \\
\quad (interTeamsGuidingIdeas\ gt = gi \Leftrightarrow dialogdiscuss\{la_1, la_2\} = \\
\quad t_1.teamguidingideas = t_2.teamguidingideas)))
\end{array}
$$

Senge recognizes basically three types of agent's behaviors towards shared vision:

- Enrollment: process of becoming part of something by choice.

[8] For example, suppose that in a particular organization there is a software development team with two developers and one supervisor, and that there is also a team of supervisors. In this case, the software development supervisor is a member of both teams.

- Commitment: involves enrollment and, in addition, a feeling of responsibility towards the realization of the vision.
- Compliance: the agent cooperates with the vision, supporting it and doing what is expected of him. They are not enrolled or committed, though. Among the different levels of compliance, Senge presents: genuine, formal, grudging, noncompliance and apathy.

Senge also recognizes that, based only on the agent's behavior, it is hard to distinguish between genuinely compliant, committed or enrolled agents. Commitment, however, brings energy, passion and excitement, according to Senge. [Sen90, p. 218]

In an ideal Learning Organization the agents develop commitment to shared vision. The agents not only play by the rules of the game, but also feel responsible by the game and change rules as required in order to accomplish the vision.

Thus, in a Learning Organization the collaboration among agents occur at the level of commitment to joint goals. However, such commitment results from the fact that the reasons underlying the joint action are rooted in the motivations of the agent. Agents collaborate because their motivations drive their actions.

In this sense, the organization affects the minds of the agents, not only their behavior. This presents a close relationship to a basic problem of sociology and Distributed Artificial Intelligence: the micro-macro link [CC96]. This problem deals with the interaction involving individual behavior and structural rules in sets of agents. In our case, it is individual behavior, driven by motivations and personal mastery, that leads to the emergence of a shared vision. However, Systems Thinking, Mental Models, and Team Learning are also important in this process. The first provides the skills that enable each individual agent to perceive reality using a systemic perspective, in which a given agent is affected by and affects the environment state. The second enables each agent to recognize its underlying assumptions while interacting with the environment. Finally, the third allows an agent to develop the perception of the existence of different individual perspectives, regarding a given context, thus enabling the production of a collective perspective.

At this point it is also convenient to group the learning team and shared vision. Therefore, the *InterPersonalDisciplines* schema includes disciplines that require interaction among agents of type *LearningOrgAgent*.

InterPersonalDisciplines

InterPersonalMentalModel
TeamLearning
sharedvision : *SharedVision*

3.2.9 Learning Organization Agent and Disciplines

Complementing what has been modelled so far, we define that a *LearningOrgAgent* develops all of Senge's disciplines. Based on its motivations,

on the principles and actions related to each discipline, a given agent develops new capabilities to act. Moreover, the agent's guiding ideas are consistent with all of its goals.

LearningOrgAgentAndDisciplines

$$
\begin{array}{l}
LearningOrgAgent \\
improveIntraPersonal : \mathbb{P}\, Motivation \times \mathbb{P}\, Action \times \\
\quad IntraPersonalDisciplines \rightarrow \mathbb{P}\, Action \\
improveInterPersonal : \mathbb{P}\, Motivation \times \mathbb{P}\, Action \times \\
\quad InterPersonalDisciplines \rightarrow \mathbb{P}\, Action \\
personalmastery : PersonalMastery \\
mentalmodels : MentalModel \\
systemsthinkingskill : SystemsThinking \\
\hline
\exists_1\, gi : GuidingIdea \mid gi = ownguidingideas \bullet \\
\quad (isVisionGoalConsistent(gi.visions, allgoals) = yes \land \\
\quad isPurposeGoalConsistent(gi.purposes, allgoals) = yes \land \\
\quad isValueGoalConsistent(gi.values, allgoals) = yes) \\
(mapset\ trifirst)(\mathrm{dom}\ improveIntraPersonal) = \{motivations\} \\
(mapset\ trisecond)(\mathrm{dom}\ improveIntraPersonal) = \{capableof\} \\
(mapset\ trifirst)(\mathrm{dom}\ improveInterPersonal) = \{motivations\} \\
(mapset\ trisecond)(\mathrm{dom}\ improveInterPersonal) = \{capableof\} \\
\mathrm{ran}\ improveIntraPersonal \neq \varnothing \land \mathrm{ran}\ improveInterPersonal \neq \varnothing
\end{array}
$$

3.2.10 Perception, Action and State

The Learning Organization Agent's perception depends on its motivations, mental models, guiding ideas and goals. The agent's goals and guiding ideas are influenced by the development of the personal mastery discipline. Moreover, the goals that result from the creative tension are contingent on the agent's personal visions.

LearningOrgAgentPerception

$$
\begin{array}{l}
LearningOrgAgentAndDisciplines \\
OrgAgentPerception \\
learnorgperceives : \mathbb{P}\, Motivation \rightarrow MentalModel \rightarrow \\
\quad GuidingIdea \rightarrow \mathbb{P}\, Role \rightarrow \mathbb{P}\, Goal \rightarrow Environment \rightarrow View \\
\hline
\mathrm{dom}\ learnorgperceives = \{motivations\} \\
\mathrm{dom}(learnorgperceives\ motivations) = \{mentalmodels\} \\
\mathrm{dom}(learnorgperceives\ motivations\ mentalmodels) = \\
\quad \{personalmastery.guidingidea\} \\
\mathrm{dom}(learnorgperceives\ motivations\ mentalmodels \\
\quad personalmastery.guidingidea) = \{roles\}
\end{array}
$$

The action of this agent is influenced by its motivations, mental models, guiding ideas, goals and plans. Also, as in the *LearningOrgAgentPerception*

schema, the agent's goals and guiding ideas are influenced by the development of the personal mastery discipline as the goals taken into account are restricted to the *ResolutionGoal* set obtained from the *creativetension* function which is defined in the *PersonalMastery* schema; and its plans are influenced by the development of the systems thinking skill.

LearningOrgAgentAction

$$
\begin{aligned}
&\mathit{LearningOrgAgentAndDisciplines}\\
&\mathit{OrgAgentAction}\\
&\mathit{learnorgact} : \mathbb{P}\,\mathit{Motivation} \rightarrow \mathit{MentalModel} \rightarrow \mathit{GuidingIdea} \rightarrow\\
&\qquad \mathbb{P}\,\mathit{ResolutionGoal} \rightarrow \mathbb{P}\,\mathit{Plan} \rightarrow \mathit{View} \rightarrow \mathit{Environment} \rightarrow \mathit{Actions}\\
\hline
&\operatorname{dom} \mathit{learnorgact} = \{\mathit{motivations}\}\\
&\operatorname{dom}(\mathit{learnorgact}\ \mathit{motivations}) = \{\mathit{mentalmodels}\}\\
&\operatorname{dom}(\mathit{learnorgact}\ \mathit{motivations}\ \mathit{mentalmodels}) =\\
&\qquad \{\mathit{personalmastery.guidingidea}\}
\end{aligned}
$$

The state of this agent depends on its perception, action and derives from the organizational agent state as defined in the *LOAgStateIncludes* schema. Moreover, some constraints related to the agent's perception and action are introduced in the *LOAgStatePerceptionActionConstraints* schema. Hence, the goals that result from the creative tension depend on the agent's personal vision and influence the *learnorgperceives* and *learnorgact* functions. Furthermore, in the *learnorgact* function, plans are influenced by the development of capabilities related to systems thinking.

LOAgStateIncludes

$$
\begin{aligned}
&\mathit{LearningOrgAgentPerception}\\
&\mathit{LearningOrgAgentAction}\\
&\mathit{OrgAgentState}\\
&\mathit{SystemsThinking}
\end{aligned}
$$

LOAgStatePerceptionActionConstraints
LOAgStateIncludes

dom(*learnorgperceives motivations mentalmodels*
personalmastery.guidingidea roles) =
personalmastery.creativetension({*actualpercepts*},
personalmastery.personalvisions)
dom(*learnorgact motivations mentalmodels*
personalmastery.guidingidea) =
{*personalmastery.creativetension*({*actualpercepts*},
personalmastery.personalvisions)}
dom((*learnorgact motivations mentalmodels*
personalmastery.guidingidea)
(*personalmastery.creativetension*({*actualpercepts*},
personalmastery.personalvisions))) =
{*systemsthinkingskill.resultingplans*}

Similarly, the *LOAgStateSystemsThinkingConstraints* schema present several constraints related to the functions defined in the *SystemsThinking* schema. For example, the function *contextassessment* is applied on *actualpercepts*, so that the construction of contexts depends on the perception of the environment; the events produced by *eventanalyser* are obtained from these contexts; *behavioralpatterns* result from analysis of the events; the construction or reuse of computational models depends not only on these patterns and events, but also on previously available generic structures and computational models. Simulations performed on a set of scenarios using the selected computational models produce *simulatedpotentialstates*. Then, based on the agent's vision, a set of states is selected, and *resultingplans* corresponds to the set of plans selected to be performed so that the agent achieves the *selectedpotentialstates*.

LOAgStateSystemsThinkingConstraints

LOAgStateIncludes

$$
\begin{array}{l}
allplans \subseteq planlib \\
\text{dom}\ contextassessment = \{actualpercepts\} \\
\text{dom}\ eventanalyser = \text{ran}\ contextassessment \\
events = eventanalyser\ (contextassessment\ (actualpercepts)) \\
\text{dom}\ behavioralpatternanalyser = \text{ran}\ eventanalyser \\
(mapset\ tetrafirst)(\text{dom}\ compmodelsdesign) = \\
\qquad \text{ran}\ behavioralpatternanalyser \\
behavioralpatterns = behavioralpatternanalyser\ (events) \\
compmodels = compmodelsdesign(behavioralpatterns, events, \\
\qquad genericstructures, compmodellib) \\
simulatedpotentialstates = simulation(compmodels, scenarios) \\
selectedpotentialstates = potentialstatesanalyser(\\
\qquad personalmastery.personalvisions, \\
\qquad simulatedpotentialstates) \\
(mapset\ tetrasecond)(\text{dom}\ compmodelsdesign) = \text{ran}\ eventanalyser \\
(mapset\ first)(\text{dom}\ simulation) \subseteq \text{ran}\ compmodelsdesign \\
(mapset\ second)(\text{dom}\ potentialstatesanalyser) \subseteq \text{ran}\ simulation \\
(mapset\ second)(\text{dom}\ planfrompotentialstate) \subseteq \{planlib\} \\
resultingplans = planfrompotentialstate(selectedpotentialstates, planlib)
\end{array}
$$

Finally, the *LearningOrgAgentState* schema includes the agent's perception, action and also the constraints presented above. Moreover, the *willdo* function depends on plans (*resultingplans*) as produced by the agent's systems thinking (*systemsthinkingskill*). The agent's actual perception are influenced by its motivations, guiding ideas, mental models, goals and roles performed by the agent.

LearningOrgAgentState

LOAgStateIncludes

$$
\begin{array}{l}
LOAgStateSystemsThinkingConstraints \\
LOAgStatePerceptionActionConstraints \\
actualpercepts = learnorgperceives\ motivations\ mentalmodels \\
\qquad personalmastery.guidingidea\ roles\ allgoals\ posspercepts \\
willdo = learnorgact\ motivations\ mentalmodels \\
\qquad personalmastery.guidingidea(\bigcup(\text{ran} \\
\qquad personalmastery.creativetension)) \\
\qquad systemsthinkingskill.resultingplans\ actualpercepts\ env
\end{array}
$$

3.2.11 Operations

Considering operations related to learning organization agents, we have the $\Delta LearningOrgAgentState$ and *LearningOrgAgentInteracts* schemes.

$\Delta LearningOrgAgentState$

$LearningOrgAgentState$
$LearningOrgAgentState'$

$\Delta OrgAgentState$
$learnorgperceives' = learnorgperceives$
$learnorgact' = learnorgact$

The *LearningOrgAgentInteracts* schema specifies that agent interaction with the environment (including interaction with other agents) is influenced by its perceptions. These perceptions serve as input to the systems thinking skill and are influenced by the mental models and guiding ideas developed by the agent. Finally, the actions that will be performed by the agent, depend on its new motivations, mental models, guiding ideas, goals and new plans produced by the agent's systems thinking capabilities.

$LearningOrgAgentInteracts$

$\Delta LearningOrgAgentState$
$DeltaSystemsThinking$
$OrgAgentInteracts$

$actualpercepts' = learnorgperceives\ motivations'$
$\quad mentalmodels'personalmastery'.guidingidea\ roles'$
$\quad allgoals'\ posspercepts'$
$systemsactualpercepts? = actualpercepts'$
$resultingplans' = resultingplans!$
$willdo' = learnorgact\ motivations'\ mentalmodels'$
$\quad personalmastery'.guidingidea\ \{allgoals'\}$
$\quad resultingplans'\ actualpercepts'\ env'$

3.3 The Learning Organization Model

Initially, we discuss how some basic types used in the definition of a Learning Organization, e.g.: groups, teams organizations and formal organizations, are defined in our model[9]. A group is a set of individuals that share a set of resources. Thus, in such a group there is no need for a joint commitment towards a joint goal. A team is a group where there is a set of goals that is shared among its members. In a learning team all members are agents of the type *LearningOrgAgent* and it is possible to develop team's guiding ideas that are shared among its members. An organization comprises an organizational structure, sets of teams, roles, norms, and organizational goals. The structure of an organization is specified as a connected graph in which the node set corresponds to teams and the edge set corresponds to relationships among the teams. Roles are specified as sets of actions that must be performed, sets of goals, sets of resources that an agent, while performing this role, has permission to manage,

[9] These types are omitted in this paper. The interested reader should refer to [Sil04]

and a level of organizational autonomy. This level defines whether the agent performing such role has some freedom to define new tasks in order to achieve the goals associated with the role (as stipulated in norms), or even to change part of the goals, or no freedom at all. We also define that an organization is formal if there are defined organizational positions and norms defined in its context, and each member in the organization performs at least one role.

We specify a Learning Organization as a type of formal organization in which all teams are learning teams. In summary, the *LearningOrg* schema specifies, via schema inclusion, that a Learning Organization is such that it has structure, roles, norms that permit that all teams develop the team learning discipline. Moreover, every agent in the organization develops its personal mastery, reflects on their mental models, have systems thinking skills, collectively develop shared visions, and perform roles in the organization.

The reference to *sharedvision.sharedvisions* gives access to the organizational shared vision. We require that this set is non-empty, ie, a Learning Organization must develop shared visions. In addition, *sharedvision.learningteams* is restricted so that the teams that produce such visions are exactly the teams that constitute the organization.

LearningOrg

$$
\begin{array}{l}
FormalOrg \\
learningteams : \mathbb{P}\, LearningTeam \\
sharedvision : SharedVision \\
\hline
sharedvision.learningteams = learningteams \\
sharedvision.sharedvisions \neq \varnothing \\
\forall\, lt : LearningTeam \mid lt \in learningteams \bullet \\
\quad (\exists_1\, te : Team \mid te \in teams \bullet (\#(te.membersgroup) = 0 \land \\
\quad te.resources = lt.resources \land \\
\quad te.commongoals = lt.commongoals \land \\
\quad te.teamguidingideas = lt.teamguidingideas \land \\
\quad te.commonplans = lt.commonplans \land \\
\quad te = (\lambda\, LearningTeam \bullet \theta\, Team)\, lt)) \\
\forall\, lt : LearningTeam \mid lt \in learningteams \bullet \\
\quad (\forall\, la : LearningOrgAgent \mid la \in lt.learningmembers \bullet \\
\quad (\exists\, r : Role \mid r \in roles \bullet perform(la, r, regset) = pos)) \\
\#teams = \#learningteams
\end{array}
$$

3.4 Goal Generation, Assessment, Adoption and Removal

In this section we consider situations involving the generation, assessment, adoption and removal of goals by agents of type *LearningOrgAgent*. The process that we describe here is inspired by the one presented in [dL01]. In that work, the *AssessGoals* schema presents an autonomous agent's goal assessment process, where the *goallibrary* variable represents a repository of goals known by the agent. The *motiveEffectGenerate* and *motiveEffectDestroy* functions return an

integer that represents, respectively, the effect of adopting or removing a set of goals on the agent's motivations. Those functions are influenced by the agent's motivations, goals and current perceptions. Hence, based on *AssessGoals*, we specify that in the goal assessment process it is relevant the role of the agent's guiding ideas: the agent has to consider the relevancy of the goals in relation to the agent's personal vision, purpose, and their consistency against the agent's values.

LearningOrgAgentAssessGoals

$$
\begin{array}{l}
LearningOrgAgentState \\
AssessGoals \\
lorgAgMotiveEffectGenerate : \mathbb{P}\, Motivation \rightarrow GuidingIdea \rightarrow \\
\quad \mathbb{P}\, Role \rightarrow \mathbb{P}\, Goal \rightarrow View \rightarrow \mathbb{P}\, Goal \rightarrow \mathbb{Z} \\
lorgAgMotiveEffectDestroy : \mathbb{P}\, Motivation \rightarrow GuidingIdea \rightarrow \\
\quad \mathbb{P}\, Role \rightarrow \mathbb{P}\, Goal \rightarrow View \rightarrow \mathbb{P}\, Goal \rightarrow \mathbb{Z} \\
\hline
goals \subseteq goallibrary \\
\forall\, gs : \mathbb{P}\, goallibrary \bullet (satisfyGenerate\ gs = \\
\quad lorgAgMotiveEffectGenerate\ motivations \\
\quad personalmastery.guidingidea\ roles\ goals\ actualpercepts\ gs \land \\
\quad satisfyDestroy\ gs = lorgAgMotiveEffectGenerate \\
\qquad motivations\ personalmastery.guidingidea\ roles\ goals \\
\qquad actualpercepts\ gs)
\end{array}
$$

In addtion, [dL01] define operations related to goal generation and destruction by an autonomous agent. Therefore, in the *GenerateGoals* schema an agent generates a set of goals in *goallibrary* if this set presents the greatest motivational effect, assessed by the *satisfyGenerate* function. Similarly, in *DestroyGoals* an agent destroys a subset of its goals if the destruction of this set presents the greatest motivational effect, assessed by the *satisfyDestroy* function. In the *LearningOrgAgentGenerateGoals* and *LearningOrgAgentDestroyGoals* schemes, respectively, we present the goal generation and destruction for the *LearningOrgAgent*. Here, the definitions of the *satisfyGenerate* and *satisfyDestroy* functions were adapted so that the agent's guiding ideas are also taken into account, as a consequence of the inclusion of the *LearningOrgAgentAssessGoals* schema.

LearningOrgAgentGenerateGoals

$$
\begin{array}{l}
\Delta LearningOrgAgentState \\
LearningOrgAgentAssessGoals \\
\hline
\exists\, gs : \mathbb{P}\, Goal \mid gs \subseteq goallibrary \bullet (\forall\, os : \mathbb{P}\, Goal \mid os \in (\mathbb{P}\, goallibrary) \bullet \\
\quad ((satisfyGenerate\ gs \geq satisfyGenerate\ os) \land goals' = goals \cup gs))
\end{array}
$$

$$
\begin{array}{|l}
\textit{LearningOrgAgentDestroyGoals} \\
\Delta \textit{LearningOrgAgentState} \\
\textit{LearningOrgAgentAssessGoals} \\
\hline
\exists\, gs : \mathbb{P}\, \textit{Goal} \mid gs \subseteq \textit{goallibrary} \bullet (\forall\, os : \mathbb{P}\, \textit{Goal} \mid os \in (\mathbb{P}\, \textit{goallibrary}) \bullet \\
\quad ((\textit{satisfyDestroy}\ gs \geq \textit{satisfyDestroy}\ os) \land \textit{goals}' = \textit{goals} \setminus gs)) \\
\hline
\end{array}
$$

Moreover, [dL01] define the *EntityAdoptGoals* and *EntityRemoveGoals* functions. The first takes an entity and a set of goals and instantiates a new entity that inherits the original entity's features with the addition of the new goals. Similarly, the second function instantiates a new entity that is identical to the original one, except for the removed goals. In our work, those functions were extended to include the new variables that we have added to the specification of an entity in our model.

$$
\begin{array}{|l}
\textit{EntityAdoptGoals} : (\textit{Entity} \times \mathbb{P}\, \textit{Goal}) \rightarrow \textit{Entity} \\
\hline
\forall\, gs : \mathbb{P}\, \textit{Goal};\ \textit{old}, \textit{new} : \textit{Entity} \bullet \\
\quad (\textit{EntityAdoptGoals}(\textit{old}, gs) = \textit{new} \Leftrightarrow \\
\quad (\textit{new.goals} = \textit{old.goals} \cup gs \land \\
\qquad \textit{new.allgoals} = \textit{old.allgoals} \cup gs \land \\
\qquad \textit{new.capableof} = \textit{old.capableof} \land \\
\qquad \textit{new.attributes} = \textit{old.attributes} \land \\
\qquad \textit{new.store} = \textit{old.store} \land \\
\qquad \textit{new.motivations} = \textit{old.motivations} \land \\
\qquad \textit{new.ownedresources} = \textit{old.ownedresources} \land \\
\qquad \textit{new.instsreq} = \textit{old.instsreq} \land \\
\qquad \textit{new.resourcesofplan} = \textit{old.resourcesofplan} \land \\
\qquad \textit{new.plans} = \textit{old.plans} \land \\
\qquad \textit{new.planforgoal} = \textit{old.planforgoal} \land \\
\qquad \textit{new.orgs} = \textit{old.orgs} \land \\
\qquad \textit{new.roles} = \textit{old.roles} \land \\
\qquad \textit{new.allplans} = \textit{old.allplans})) \\
\end{array}
$$

$$EntityRemoveGoals : (Entity \times \mathbb{P}\, Goal) \rightarrow Entity$$

$$\begin{aligned}
&\forall\, gs : \mathbb{P}\, Goal;\ old, new : Entity \bullet \\
&\quad (EntityRemoveGoals(old, gs) = new \Leftrightarrow \\
&\quad (new.goals = old.goals \setminus gs \wedge \\
&\qquad new.allgoals = old.allgoals \setminus gs \wedge \\
&\qquad new.capableof = old.capableof \wedge \\
&\qquad new.attributes = old.attributes \wedge \\
&\qquad new.store = old.store \wedge \\
&\qquad new.motivations = old.motivations \wedge \\
&\qquad new.ownedresources = old.ownedresources \wedge \\
&\qquad new.instsreq = old.instsreq \wedge \\
&\qquad new.resourcesofplan = old.resourcesofplan \wedge \\
&\qquad new.plans = old.plans \wedge \\
&\qquad new.planforgoal = old.planforgoal \wedge \\
&\qquad new.orgs = old.orgs \wedge \\
&\qquad new.roles = old.roles \wedge \\
&\qquad new.allplans = old.allplans))
\end{aligned}$$

At this point we consider goal adoption and removal by a *LearningOrgAgent*: the agent has to consider its motivations and guiding ideas to assess its interest in adopting (or removing) a given set of goals. The *getLOrgAgFromGenerateGoals* auxiliary function is used to extract the definitions concerning a *LearningOrgAgent* based on the type *LearningOrgAgentGenerateGoals*.

$$\begin{aligned}
&getLOrgAgFromGenerateGoals == (\lambda \\
&\quad LearningOrgAgentGenerateGoals \bullet LearningOrgAgent)
\end{aligned}$$

The specification of the *lOrgAgentAdoptGoals* and *lOrgAgentRemoveGoals* functions is based, respectively, on the *EntityAdoptGoals* and *EntityRemoveGoals*, defined above.

$$\begin{aligned}
&lOrgAgentAdoptGoals : (LearningOrgAgent \times \mathbb{P} \\
&\quad Goal) \rightarrow LearningOrgAgent \\
&\forall\, gs : \mathbb{P}\, Goal;\ oldAg, newAg : LearningOrgAgent \bullet \\
&\quad (\exists_1\, lag : GuidingIdea \mid lag = oldAg.owngguidingideas \bullet \\
&\quad (\exists_1\, oldAgGGoals : LearningOrgAgentGenerateGoals \mid \\
&\quad oldAg \in getLOrgAgFromGenerateGoals\,(oldAgGGoals) \bullet \\
&\quad (lOrgAgentAdoptGoals(oldAg, gs) = newAg \Leftrightarrow \\
&\quad (isVisionGoalConsistent(lag.visions, gs) = yes \wedge \\
&\qquad isPurposeGoalConsistent(lag.purposes, gs) = yes \wedge \\
&\qquad isValueGoalConsistent(lag.values, gs) = yes \wedge \\
&\qquad gs \subseteq oldAgGGoals.goallibrary \wedge \\
&\qquad (\forall\, os : \mathbb{P}\, Goal \mid os \in (\mathbb{P}\, oldAgGGoals.goallibrary) \bullet \\
&\qquad\quad oldAgGGoals.satisfyGenerate\ gs \geq \\
&\qquad\quad oldAgGGoals.satisfyGenerate\ os) \wedge \\
&\qquad EntityAdoptGoals(oldAg, gs) = newAg))))
\end{aligned}$$

$$
\begin{array}{l}
lOrgAgentRemoveGoals : (LearningOrgAgent \times \mathbb{P} \\
\quad Goal) \rightarrow LearningOrgAgent \\
\hline
\forall\, gs : \mathbb{P}\, Goal;\ oldAg, newAg : LearningOrgAgent \\
\quad \bullet\ (\exists_1\, lag : GuidingIdea \mid lag = \\
\quad oldAg.ownguidingideas \bullet (\exists_1\, oldAgGGoals : \\
\quad LearningOrgAgentGenerateGoals \mid oldAg \in \\
\quad getLOrgAgFromGenerateGoals\,(oldAgGGoals) \bullet (\exists_1 \\
\quad newGoalSet : \mathbb{P}\, Goal \mid newGoalSet = oldAg.goals \setminus gs \bullet \\
\quad (lOrgAgentRemoveGoals(oldAg, gs) = newAg \Leftrightarrow \\
\quad (isVisionGoalConsistent(lag.visions, newGoalSet) = yes \wedge \\
\qquad isPurposeGoalConsistent(lag.purposes, newGoalSet) = yes \wedge \\
\qquad isValueGoalConsistent(lag.values, newGoalSet) = yes \wedge \\
\qquad gs \subseteq oldAgGGoals.goallibrary \wedge \\
\qquad (\forall\, os : \mathbb{P}\, Goal \mid os \in (\mathbb{P}\, oldAgGGoals.goallibrary) \bullet \\
\qquad\quad oldAgGGoals.satisfyGenerate\ gs \geq \\
\qquad\quad oldAgGGoals.satisfyGenerate\ os) \wedge \\
\qquad EntityRemoveGoals(oldAg, gs) = newAg)))))
\end{array}
$$

In the *LearningOrgAgentAdoptGoals* and *LearningOrgAgentRemoveGoals* schemes we define the goal adoption and removal by a *LearningOrgAgent*. This specification is based on the *lOrgAgentAdoptGoals* and *lOrgAgentRemoveGoals* functions, respectively, and therefore goal adoption and removal depend on the motivational effect associated to this process and to the consistency of these goals with the agent's guiding ideas. Furthermore, we use the *LearningOrgAgentGenerateGoals* schema to define that an agent adopts a goal just as if it had generated such goal [10]. Hence, considering a set of available goals (*goallibrary*) the *LearningOrgAgent* adopts those goals that present the greatest return for the *satisfyGenerate* function

$$
\begin{array}{l}
LearningOrgAgentAdoptGoals \\
\hline
\Delta LearningTeam \\
aa? : LearningOrgAgent \\
gs? : \mathbb{P}\, Goal \\
\hline
aa? \in learningmembers \\
learningmembers' = (learningmembers \setminus \{aa?\}) \cup \\
\quad \{lOrgAgentAdoptGoals(aa?, gs?)\} \\
membersgroup' = membersgroup \\
resources' = resources \\
teamguidingideas' = teamguidingideas \\
commonplans' = commonplans \\
developcommonplans' = developcommonplans \\
commongoals' = \textbf{if}\ (\forall\, a : LearningOrgAgent \mid a \in learningmembers' \bullet \\
\quad gs? \subseteq a.goals)\ \textbf{then}\ commongoals \cup (gs? \setminus (gs? \cap commongoals)) \\
\qquad\qquad \textbf{else}\ commongoals
\end{array}
$$

[10] This strategy is inspired by the one adopted in [dL01].

LearningOrgAgentRemoveGoals

$\Delta LearningTeam$
$aa? : LearningOrgAgent$
$gs? : \mathbb{P}\ Goal$

$aa? \in learningmembers$
$gs? \subseteq aa?.goals$
$learningmembers' = (learningmembers \setminus \{aa?\}) \cup$
$\quad \{lOrgAgentRemoveGoals(aa?, gs?)\}$
$membersgroup' = membersgroup$
$resources' = resources$
$teamguidingideas' = teamguidingideas$
$commonplans' = commonplans$
$developcommonplans' = developcommonplans$
$commongoals' = \textbf{if}\ ((gs? \cap commongoals) \subset commongoals)$
$\quad \textbf{then}\ commongoals \setminus (gs? \cap commongoals)$
$\quad \textbf{else}\ developCommonGoals\ (learningmembers')$

3.5 Interactions, Knowledge and Mental Models

In this section we present a formalization for some of the processes involving interactions among agents of type *LearningOrgAgent* and how these interactions affect their knowledge and mental models. This section is divided in four subsections. The first deals with aspects related to communication among agents. In the second we present a model for the agent's knowledge and mental models. The third presents a model for interactions and conversations among agents. In the last subsection two specific conversation types are presented: dialog and discussion, based on the team learning discipline. We note that in this section the term "agent" refers to a *LearningOrgAgent*. Moreover, in this work the distinction between "beliefs" and "knowledge" is not relevant, hence, they are considered to be equivalent.

3.5.1 Communication

Initially, we define that a *Message* corresponds to a set of *Attribute*. Although there are more detailed definitions for message and agent communication[FFMM94, fIPA00], this level of abstraction is adequate for our work. Hence, messages may carry information about some specific entity or about the environment in general.

$$Message == \mathbb{P}\ Attribute$$

We assume, also, that all communication occur under a given conversational context that we formalize as a subject context. These subject contexts may involve agents' beliefs or goals.

$$\begin{aligned} SubjectContext ::= & \; bels\langle\langle AGBelief \rangle\rangle \\ | & \; subjgoals\langle\langle \mathbb{P}\, Goal \rangle\rangle \end{aligned}$$

The agent's perceptions include several types of information obtained from the environment. This is also the case for messages destined to a particular agent. In the *MessageExtraction* schema we define that the extraction of a message from percepts involves specific actions (*messageExtractionActions*). Clearly, the agent must be capable of performing such actions. The *getMessage* function takes sets of actions and percepts and returns a message. Moreover, message extraction only occurs if the agent selects (*learnorgact*) the actions required to perform this activity.

MessageExtraction

$$\begin{aligned} & LearningOrgAgentState \\ & messageExtractionActions : \mathbb{P}\, Action \\ & getMessage : (\mathbb{P}\, Action \times View) \rightarrow Message \\ \hline & messageExtractionActions \subset internalperactions \\ & (mapset\ first)(\mathrm{dom}\ getMessage) = \{messageExtractionActions\} \\ & \forall\, v : View;\ acts : \mathbb{P}\, Action;\ m : Message \bullet \\ & \quad (getMessage(acts, v) = m \Leftrightarrow \\ & \quad (acts = messageExtractionActions \land \\ & \qquad messageExtractionActions \neq \varnothing \land \\ & \qquad messageExtractionActions = learnorgact\ motivations \\ & \qquad\quad mentalmodels\ personalmastery.guidingidea\ \{allgoals\} \\ & \qquad\quad resultingplans\ actualpercepts\ env \land \\ & \qquad v = learnorgperceives\ motivations\ mentalmodels \\ & \qquad\quad personalmastery.guidingidea\ roles\ allgoals\ posspercepts)) \end{aligned}$$

In the *ExtractMessage* schema we present an abstraction for the extraction process. This involves the reception of new percepts (*newPercepts*?) and the extraction of a given message (*extractedMessage*!).

ExtractMessage

$$\begin{aligned} & LearningOrgAgentInteracts \\ & MessageExtraction \\ & newPercepts? : View \\ & extractedMessage! : Message \\ \hline & willdo' = messageExtractionActions \\ & extractedMessage! = getMessage(messageExtractionActions, \\ & \quad newPercepts?) \end{aligned}$$

The agent interprets messages. In the *MessageInterpretation* schema we define that the interpretation process involves actions

(*messageInterpretationActions*) that the agent must be capable of performing. The function *interpretMessage* takes a set of actions and a message and returns an interpretation (*View*). In addition, interpretation only occurs if the required actions are selected by the agent in its action selection (*learnorgact*) function.

MessageInterpretation

$$
\begin{array}{l}
LearningOrgAgentState \\
messageInterpretationActions : \mathbb{P}\, Action \\
interpretMessage : (\mathbb{P}\, Action \times Message) \rightarrowtail\!\!\!\!\!\!\rightarrow View \\
\hline
messageInterpretationActions \subset capableof \\
(mapset\ first)(\mathrm{dom}\ interpretMessage) = \\
\quad \{messageInterpretationActions\} \\
\forall\, v : View;\ acts : \mathbb{P}\, Action;\ m : Message \bullet \\
\quad (interpretMessage(acts, m) = v \Leftrightarrow \\
\quad (acts = messageInterpretationActions \land \\
\qquad messageInterpretationActions \neq \varnothing \land \\
\qquad messageInterpretationActions = learnorgact\ motivations \\
\qquad\quad mentalmodels\ personalmastery.guidingidea\ \{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env))
\end{array}
$$

Thus, given an extracted message, the *InterpretMessage* operation produces an interpreted message. The interpretation is influenced by the agent's mental models.

InterpretMessage

$$
\begin{array}{l}
LearningOrgAgentInteracts \\
MessageInterpretation \\
extractedMessage? : Message \\
interpretedMessage! : Message \\
\hline
willdo' = messageInterpretationActions \\
interpretedMessage! = interpretMessage \\
\quad (messageInterpretationActions, extractedMessage?) \\
store' = store \cup interpretedMessage!
\end{array}
$$

Internal views, such as the ones produced by the interpretation process, can be recalled from the agent's store.

ViewRecalling

$$
\begin{array}{l}
LearningOrgAgentState \\
recallActions : \mathbb{P}\, Action \\
recallView : (\mathbb{P}\, Action \times View) \rightarrowtail\!\!\!\!\!\!\rightarrow View \\
\hline
recallActions \subset internalperactions \\
(mapset\ first)(\mathrm{dom}\, recallView) = \{recallActions\} \\
\forall\, v1, v2 : View;\ acts : \mathbb{P}\, Action \bullet (recallView \\
\quad (acts, v1) = v2 \Leftrightarrow \\
\quad (acts = recallActions \land \\
\qquad recallActions \neq \varnothing \land \\
\qquad v1 \subset store \land \\
\qquad recallActions = learnorgact\ motivations\ mentalmodels \\
\qquad\quad personalmastery.guidingidea\ \{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env))
\end{array}
$$

Recalling of an internal concept involves the reception of a given internal image and, if that image is stored in the agent's memory, returning it (*recalledView*!).

RecallView

$$
\begin{array}{l}
LearningOrgAgentInteracts \\
ViewRecalling \\
whatView? : View \\
recalledView! : View \\
\hline
willdo' = recallActions \\
recalledView! = recallView(recallActions, whatView?)
\end{array}
$$

Alternatively, the agent may be able to infer views that were not previously stored in its store.

ViewInfer

$$
\begin{array}{l}
LearningOrgAgentState \\
inferActions : \mathbb{P}\, Action \\
inferView : (\mathbb{P}\, Action \times View) \rightarrowtail\!\!\!\!\!\!\rightarrow View \\
\hline
inferActions \subset capableof \\
(mapset\ first)(\mathrm{dom}\, inferView) = \{inferActions\} \\
\forall\, v1, v2 : View;\ acts : \mathbb{P}\, Action \bullet \\
\quad (inferView\ (acts, v1) = v2 \Leftrightarrow \\
\quad (acts = inferActions \land \\
\qquad inferActions \neq \varnothing \land \\
\qquad v1 \cap store = \varnothing \land \\
\qquad inferActions = learnorgact\ motivations\ mentalmodels \\
\qquad\quad personalmastery.guidingidea\ \{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env))
\end{array}
$$

InferView

$$
\begin{array}{l}
LearningOrgAgentInteracts \\
ViewInfer \\
whatView? : View \\
inferredView! : View \\
\hline
willdo' = inferActions \\
inferredView! = inferView(inferActions, whatView?) \\
store' = store \cup inferredView!
\end{array}
$$

The agent is also capable of producing messages for future transmission. These messages are also influenced by the agent's motivations and mental models.

MessageProduction

$$
\begin{array}{l}
LearningOrgAgentState \\
messageProductionActions : \mathbb{P}\, Action \\
produceMessage : \mathbb{P}\, Action \rightarrowtail\!\!\!\!\!\rightarrow Message \\
\hline
messageProductionActions \subset capableof \\
\mathrm{dom}\, produceMessage = \{messageProductionActions\} \\
\forall\, m : Message;\ acts : \mathbb{P}\, Action \bullet \\
\quad (produceMessage\ acts = m \Leftrightarrow \\
\quad (acts = messageProductionActions \land \\
\qquad messageProductionActions \neq \varnothing \land \\
\qquad messageProductionActions = learnorgact\ motivations \\
\qquad\quad mentalmodels\ personalmastery.guidingidea\ \{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env))
\end{array}
$$

ProduceMessage

$$
\begin{array}{l}
LearningOrgAgentInteracts \\
MessageProduction \\
newEnv? : \mathbb{P}\, Attribute \\
producedMessage! : Message \\
\hline
env' = newEnv? \\
willdo' = messageProductionActions \\
producedMessage! = produceMessage\ messageProductionActions
\end{array}
$$

An agent transmits messages using actions that are a subset of the agent's capabilities. As we are interested in specifying sequences of interactions, in the function *expose* we also use an index in its domain.

MessageExposition

$$
\begin{array}{l}
LearningOrgAgentState \\
exposeActions : \mathbb{P}\, Action \\
expose : (\mathbb{P}\, Action \times Message \times \mathbb{N}_1) \rightarrowtail\!\!\!\!\!\!\rightarrow Message \\
\hline
exposeActions \subset learningTeamCapabilities \\
(mapset\ trifirst)(\mathrm{dom}\ expose) = \{exposeActions\} \\
\forall\, mint, mexp : Message;\ acts : \mathbb{P}\, Action;\ ind : \mathbb{N}_1 \bullet \\
\quad (expose(acts, mint, ind) = mexp \Leftrightarrow \\
\quad (acts = exposeActions \land \\
\qquad exposeActions \neq \varnothing \land \\
\qquad exposeActions = learnorgact\ motivations\ mentalmodels \\
\qquad\quad personalmastery.guidingidea\ \{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env))
\end{array}
$$

A message internally produced using the *ProduceMessage* operation may be transmitted by the agent.

ExposeMessage

$$
\begin{array}{l}
LearningOrgAgentInteracts \\
MessageExposition \\
ind? : \mathbb{N}_1 \\
producedMessage? : Message \\
exposedMessage! : Message \\
\hline
env' = env \\
willdo' = exposeActions \\
exposedMessage! = expose(exposeActions, producedMessage?, ind?)
\end{array}
$$

3.5.2 Interpretation, Knowledge and Mental Models

In this section we introduce models for the knowledge and mental models of a given agent. We assume here that the agent's mental model includes its knowledge and beliefs.

An agent knows a view if it can be recalled from its store or if it can infer it from the views contained in its store.

LOrgAgentKnowledgeIncludes

$$
\begin{array}{l}
ViewInfer \\
ViewRecalling \\
knowingActions : \mathbb{P}\, Action \\
knowsAView : (\mathbb{P}\, Action \times View) \rightarrow Consistency \\
knowsAViewState : (\mathbb{P}\, Action \times View) \rightarrow \mathbb{P}\, Attribute \\
knows, doesNotKnow : Attribute \\
\hline
knowingActions \subset capableof \\
(mapset\ first)(\mathrm{dom}\ knowsAView) = \{knowingActions\} \\
(mapset\ first)(\mathrm{dom}\ knowsAViewState) = \{knowingActions\} \\
\mathrm{ran}\ knowsAViewState = \{\{knows\}, \{doesNotKnow\}\}
\end{array}
$$

The *knowsAView* function returns *yes* when the received concept is already stored in the agent's store, or if its possible to infer this concept from the knowledge present in this store. Otherwise, this function returns *no*. The *LOrgAgentKnowledgeKnows* and *LOrgAgentKnowledgeDoesNotKnow* schemes describe, respectively, the first and second cases.

LOrgAgentKnowledgeKnows

$$
\begin{array}{l}
LOrgAgentKnowledgeIncludes \\
\hline
\forall\, v : View;\ acts : \mathbb{P}\, Action \bullet (\exists\, v1 : \\
\quad View \mid v1 \subset store \bullet (knowsAView(acts, v) = yes \Leftrightarrow \\
\quad (acts = knowingActions \wedge \\
\qquad knowingActions \neq \varnothing \wedge \\
\qquad knowingActions = learnorgact\ motivations\ mentalmodels \\
\qquad\quad personalmastery.guidingidea\{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env \wedge \\
\qquad (recallView(acts, v1) = v \vee \\
\qquad\quad inferView(acts, v1) = v))))
\end{array}
$$

LOrgAgentKnowledgeDoesNotKnow

$$
\begin{array}{l}
LOrgAgentKnowledgeIncludes \\
\hline
\forall\, v : View;\ acts : \mathbb{P}\, Action \bullet (\forall\, v1 : \\
\quad View \mid v1 \subset store \bullet (knowsAView(acts, v) \neq yes \Leftrightarrow \\
\quad (acts = knowingActions \wedge \\
\qquad knowingActions \neq \varnothing \wedge \\
\qquad knowingActions = learnorgact\ motivations\ mentalmodels \\
\qquad\quad personalmastery.guidingidea\{allgoals\} \\
\qquad\quad resultingplans\ actualpercepts\ env \wedge \\
\qquad (recallView(acts, v1) \neq v \wedge \\
\qquad\quad inferView(acts, v1) \neq v))))
\end{array}
$$

Similarly, in the *LOrgAgentKnowledgeState* schema the *knowsAViewState* function returns the *knows* attribute when the received concept is already stored

in the agent's store or if it can be infered from this store contents. Otherwise, it returns *doesNotKnow*.

LOrgAgentKnowledgeState

$$
\begin{array}{l}
LOrgAgentKnowledgeIncludes \\
\hline
\forall\, v : View;\ acts : \mathbb{P}\, Action;\ c : Consistency \bullet \\
\quad (knowsAViewState(acts, v) = \{knows\} \Leftrightarrow \\
\quad (knowsAView(acts, v) = yes \land c = yes)) \\
\forall\, v : View;\ acts : \mathbb{P}\, Action;\ c : Consistency \bullet \\
\quad (knowsAViewState(acts, v) = \{doesNotKnow\} \Leftrightarrow \\
\quad (knowsAView(acts, v) \neq yes \land c \neq yes))
\end{array}
$$

Finally, the *LearningOrgAgentKnowledge* schema describes that there are known and unknown concepts for a given agent.

$$
\begin{aligned}
LearningOrgAgentKnowledge \mathrel{\widehat{=}}\ & LOrgAgentKnowledgeKnows \land \\
& LOrgAgentKnowledgeDoesNotKnow \land \\
& LOrgAgentKnowledgeState
\end{aligned}
$$

The operation described in the *LearningOrgAgentKnowsView* schema receives a specific concept as input (*whatView*?) and returns an answer (*answer*!). This answer may be *yes* if the agent knows this concept, otherwise this answer will be *no*.

LearningOrgAgentKnowsView

$$
\begin{array}{l}
LearningOrgAgentInteracts \\
LearningOrgAgentKnowledge \\
whatView? : View \\
answer! : Consistency \\
\hline
willdo' = knowingActions \\
answer! = knowsAView(knowingActions, whatView?)
\end{array}
$$

An agent capable of interacting has knowledge, extracts, interprets, produces, and exposes messages.

$$
\begin{aligned}
LOrgAgentInts \mathrel{\widehat{=}}\ & LearningOrgAgentKnowsView \land ExtractMessage \land \\
& InterpretMessage \land RecallView \land ProduceMessage \land ExposeMessage \\
& \land InferView
\end{aligned}
$$

Now, we specify that an agent performing interactions may have models of these interactions, and also, may change these models.

LOrgAgentIntsModels

$$
\begin{array}{l}
LOrgAgentInts \\
modelOfAgentInteractions : LOrgAgentInteracting \rightarrow \\
\quad LOrgAgentInteractingModel \\
changeModelOfAgentInteractions : (LOrgAgentInteracting \\
\quad \times LOrgAgentInteractingModel) \rightarrow LOrgAgentInteractingModel \\
\hline
\mathrm{ran}\, modelOfAgentInteractions = mentalmodels.modelLOrgAgInts \\
\mathrm{ran}\, changeModelOfAgentInteractions = \\
\quad mentalmodels.modelLOrgAgInts
\end{array}
$$

$$LOrgAgentInteracting \mathrel{\widehat=} [LOrgAgentIntsModels]$$

What is the agent capable of knowing ? The schema *WhatKnow* shows a few examples of what is possible to represent concerning a given agent's knowledge. The agent a_i knows its own vision v_1. It also knows its interpretation of view v_1. The agent a_i also knows its interpretation for the message sent by a_j.

WhatKnow

$$
\begin{array}{l}
interactionPool : \mathbb{P}\, LOrgAgentInteracting \\
\hline
\forall\, a_i, a_j : LOrgAgentInteracting \mid a_i \in interactionPool \land \\
\quad a_j \in interactionPool \land a_i \neq a_j \bullet (\exists\, v_1, v_2 : View;\ m_j : \\
\quad Message;\ ind_j : \mathbb{N}_1 \bullet (a_i.knowsAView \\
\quad (a_i.knowingActions, v_1) = yes \land \\
\quad a_i.knowsAView(a_i.knowingActions, \\
\qquad a_i.interpretMessage \\
\qquad (a_i.messageInterpretationActions, v_1)) = yes \land \\
\quad a_i.knowsAView(a_i.knowingActions, \\
\qquad a_i.interpretMessage \\
\qquad (a_i.messageInterpretationActions, a_j.expose \\
\qquad (a_j.exposeActions, m_j, ind_j))) = yes))
\end{array}
$$

3.5.3 Agent Interactions

The *SendMessage* schema formalizes the process of a specific agent sending a message to a group of agents. Here, we suppose that the sender agent broadcasts its *messageSent* message.

SendMessage

$$
\begin{array}{l}
sender : LOrgAgentInteracting \\
receivers : \mathbb{P}\, LOrgAgentInteracting \\
messageSent : Message \\
\hline
sender \notin receivers \\
receivers \neq \varnothing
\end{array}
$$

Every agent in the *receivers* group receives its respective message (in *messagesReceived*).

ReceiveMessage

$$
\begin{array}{l}
sender : LOrgAgentInteracting \\
receivers : \mathbb{P}\, LOrgAgentInteracting \\
messagesReceived : \mathbb{P}\, Message \\
\hline
sender \notin receivers \\
receivers \neq \varnothing \\
\#messagesReceived = \#receivers
\end{array}
$$

A complete interaction involves sending and receiving a message. The presence of noise in communication defines to what degree the sent and received messages are similar. Each receiver agent has its own interpretation (*interpretedMessage*) of its respective received message (a member of *messagesReceived*). However, every receiver agent knows that the received message is influenced by the exposition performed by the sender agent, and this exposition is influenced by the sender's mental models. Each agent in the group knows that the other agents have their own interpretation (*interpretedMessage*) of their respective member of *messagesReceived*.

Interaction

$$
\begin{array}{l}
SendMessage \\
ReceiveMessage \\
agents : \mathbb{P}\, LOrgAgentInteracting \\
interactionIndex : \mathbb{N}_1 \\
\hline
agents = \{sender\} \cup receivers \\
\exists_1\, internalMessage : Message \bullet messageSent = \\
\quad sender.expose(sender.exposeActions, internalMessage, \\
\quad interactionIndex) \\
\forall\, ag : LOrgAgentInteracting \mid ag \in receivers \bullet \\
\quad (\exists\, messRec : Message \mid messRec \in \\
\quad messagesReceived \bullet (messRec = ag.interpretMessage \\
\quad (ag.messageInterpretationActions, messageSent) \wedge \\
\quad ag.knowsAView(ag.knowingActions, messRec) = yes)) \\
\forall\, a\_k, a\_j : LOrgAgentInteracting \mid a\_k \in \\
\quad receivers \wedge \\
\quad a\_j \in receivers \bullet a\_k.knowsAView \\
\quad (a\_k.knowingActions, a\_j.interpretMessage \\
\quad (a\_j.messageInterpretationActions, messageSent)) = yes
\end{array}
$$

In the initial state of an interaction the *messageSent* was sent but has not been received by the receivers.

InteractionIni

$$Interaction$$

$$interactionIndex = 1$$
$$messagesReceived = \varnothing$$

After receiving a message, each receiver has its own copy of the received message which is a member of *messagesReceived*. The presence of noise in communication defines to what degree the sent and received messages are similar. Each receiver agent has its own interpretation (*interpretedMessage*) of its respective received message (contained in *messageReceived*). However, every receiver agent knows that the received message is influenced by the exposition performed by the sender agent, and this exposition is influenced by the sender's mental models. Each agent in the group knows that the other agents have their own interpretation *interpretedMessage* of their respective received message (member of *messagesReceived*).

RecMsgIncreaseIndex

$$\Delta Interaction$$

$$interactionIndex' = interactionIndex + 1$$
$$sender' = sender$$
$$receivers' = receivers$$
$$agents' = agents$$
$$\#messagesReceived' = \#receivers'$$

Each agent that receives a message, knows that it is subject to the influence of the sender's mental models.

RecMsgSenderInternalView

$$\Delta Interaction$$

$$\exists\, senderInternalViewOfMessage : Message \bullet messageSent' = sender.expose(sender.exposeActions, senderInternalViewOfMessage, interactionIndex')$$

Each receiver agent has knowledge of his own interpretation of the received message.

RecMsgInterpret

$$\Delta Interaction$$

$$\forall\, ag : LOrgAgentInteracting \mid ag \in receivers' \bullet (\exists\, messRec : Message \mid messRec \in messagesReceived' \bullet (messRec = ag.interpretMessage(ag.messageInterpretationActions, messageSent') \land ag.knowsAView(ag.knowingActions, messRec) = yes))$$

In addition, each agent in the group knows that the others have their own interpretation of their messages.

RecMsgAllInterpret

$\Delta Interaction$

$$\forall\, a\_k, a\_j : LOrgAgentInteracting \mid a\_k \in receivers \land a\_j \in receivers \bullet$$
$$a\_k.knowsAView(a\_k.knowingActions, a\_j.interpretMessage$$
$$(a\_j.messageInterpretationActions, messageSent')) = yes$$

Furthermore, after receiving their messages, each receiver agent can change its models regarding the sender agent. This model can be of a honest of dishonest agent.

RecMsgUpdateModels

$\Delta Interaction$

$$\forall\, ag : LOrgAgentInteracting \mid ag \in receivers' \bullet$$
$$(\exists\, senderOldModel,$$
$$senderNewModel : LOrgAgentInteractingModel \mid$$
$$senderOldModel \in ag.mentalmodels.modelLOrgAgInts \land$$
$$senderOldModel = sender.modelOfAgentInteractions\,(ag) \bullet$$
$$(ag.changeModelOfAgentInteractions($$
$$sender, senderOldModel) = senderNewModel \land$$
$$ag.mentalmodels.modelLOrgAgInts = \{senderNewModel\} \cup$$
$$ag.mentalmodels.modelLOrgAgInts \setminus \{senderOldModel\}))$$

Finally, message reception (*ReceivingMessage*) involves all the aspects described above.

$$ReceivingMessage \mathrel{\widehat{=}} RecMsgIncreaseIndex \land RecMsgSenderInternalView$$
$$\land\, RecMsgInterpret \land RecMsgAllInterpret$$
$$\land\, RecMsgUpdateModels$$

In some interactions the same sender agent may send more than one message.

SameSenderNewMessage

$\Delta Interaction$

$$interactionIndex' = interactionIndex + 1$$
$$sender' = sender$$
$$receivers' = receivers$$
$$agents' = agents$$
$$messagesReceived' = \varnothing$$
$$\exists\, senderInternalViewOfMessage : Message \bullet$$
$$messageSent' = sender.expose(sender.exposeActions,$$
$$senderInternalViewOfMessage, interactionIndex')$$

Alternatively, a different agent may act as a sender. In this case, we assume, as a simplification hypothesis, that the set of agents interacting does not change. However, one agent among the members of the previous receivers set becomes the sender agent. Simultaneously, the previous sender agent join the receivers group. The new message is stored in $messageSent'$ and the $messagesReceived'$ set becomes empty.

ChangeSender

$$\Delta Interaction$$
$$newSender? : LOrgAgentInteracting$$

$$interactionIndex' = interactionIndex + 1$$
$$agents' = agents$$
$$messagesReceived' = \varnothing$$
$$newSender? \in receivers$$
$$receivers' = \{sender\} \cup (receivers \setminus \{ newSender?\})$$
$$sender' = newSender?$$
$$\exists\, senderInternalViewOfMessage : Message \bullet messageSent' = sender'.expose\,(sender'.exposeActions, senderInternalViewOfMessage, interactionIndex')$$

Finally, when the interaction ends there are no more messages nor agents involved in interactions.

InteractionEnd

$$Interaction$$

$$interactionIndex > 0$$
$$agents = \varnothing$$

Now we define a relation *InteractRel* that maps from one *Interaction* state to the next one, based in [DS89]. *InteractRel* corresponds to the set that contains all relations of this type.

$$RelationRecMsg == \mathbb{P}\{ReceivingMessage \bullet (\theta Interaction, \theta Interaction')\}$$
$$RelationNewMsg == \mathbb{P}\{SameSenderNewMessage \bullet (\theta Interaction, \theta Interaction')\}$$
$$RelationChgSnd == \mathbb{P}\{ChangeSender \bullet (\theta Interaction, \theta Interaction')\}$$
$$InteractRel == RelationRecMsg \cup RelationNewMsg \cup RelationChgSnd$$

Operations on a given state *opInteraction* define a relation between states before and after. For example, let $rel : InteractRel$ e $s1, s2 : opInteraction$, then $s1 \underline{rel}\, s2$ if $s1 \in \mathrm{dom}\, rel$, and after applying *op* on $s1$ the system will be in state $s2$ [DS89].

opInteraction

$$Interaction$$
$$nextInt : InteractRel$$
$$Interaction \in \{\mathrm{dom}\, nextInt\}$$

Next, we define that a history is a sequence of interactions where the first element of the sequence corresponds to the *InteractionIni* state, and that every state in the sequence is related to the next one via a *nextInt* operation.

History

$$hist : \mathrm{seq}\, opInteraction$$
$$\begin{aligned}
&\exists\, interactIni : Interaction \bullet (\exists\, opint : opInteraction \mid \\
&\quad opint = head(\{1\} \upharpoonright hist) \bullet \\
&\quad (interactIni = (\lambda\, opInteraction \bullet \theta Interaction)\, opint \wedge \\
&\quad interactIni \in \{InteractionIni\})) \\
&\forall\, i : \mathbb{N}_1 \mid i \in \mathrm{dom}\, hist \setminus \{1\} \bullet \\
&\quad (\exists\, interactILess1, interactI : Interaction;\ nextIntILess1 : \\
&\quad InteractRel \bullet (\exists\, opintILess1, opintI : opInteraction \mid \\
&\quad opintILess1 = head(\{i-1\} \upharpoonright hist) \wedge \\
&\quad opintI = head(\{i\} \upharpoonright hist) \wedge \\
&\quad nextIntILess1 = opintILess1.nextInt \bullet \\
&\quad (interactILess1 = (\lambda\, opInteraction \bullet \theta Interaction)\, opintILess1 \wedge \\
&\quad interactI = (\lambda\, opInteraction \bullet \theta Interaction)\, opintI \wedge \\
&\quad interactILess1\ \underline{nextIntILess1}\ interactI)))
\end{aligned}$$

Our goal is to specify sequences where agents continuously exchange messages. Consequently, we specify a particular sequence of operations. In the sketch presented below the numbers correspond to indexes for the interactions and `ini` corresponds to *InteractionIni* state, `changesender` corresponds to *ChangeSender* state, `receivingmsg` corresponds to *ReceivingMessage* state, `samesender` corresponds to *SameSenderNewMessage* state, and `end` corresponds to *InteractionEnd* state.

```
ini(1) - changesender(1) - Interaction(2) - receivingmsg(2) -
                                            Interaction(3)
or changesender(3) - Interaction(4) - receivingmsg(4) -
                                            Interaction(5)
or samesender(3)   - Interaction(4) - receivingmsg(4) -
                                            Interaction(5)
or end
```

Therefore, the first operation in such sequence must be a change of sender agent. In fact, in this type of sequences, operations with an even index must be message receiving operations. Analagously, operations with an odd index must be either a change in sender role, or another send operation executed by the same agent already performing the sender role. Hence, we specify the *Progress*

schema, that presents a sequence of operations that follows the rules regarding interaction indexes discussed above. Initially, we present a simple specification for odd numbers.

$$Odd == \{int : \mathbb{Z} \bullet 2 * int + 1\}$$

Progress

$History$
$rrmsg : RelationRecMsg$
$rnmsg : RelationNewMsg$
$rchgs : RelationChgSnd$

$\forall i : \mathbb{N}_1 \mid i \in \operatorname{dom} hist \wedge i = 1 \bullet$
$(\exists interactI : Interaction;\ nextIntI : InteractRel \bullet$
$(\exists opintI : opInteraction \mid opintI = head(\{i\} \upharpoonright hist) \wedge$
$nextIntI = opintI.nextInt \bullet nextIntI = rchgs))$
$\forall i : \mathbb{N}_1 \mid i \in \operatorname{dom} hist \wedge i \notin Odd \bullet$
$(\exists interactI : Interaction;\ nextIntI : InteractRel \bullet$
$(\exists opintI : opInteraction \mid opintI = head(\{i\} \upharpoonright hist) \wedge$
$nextIntI = opintI.nextInt \bullet nextIntI = rrmsg))$
$\forall i : \mathbb{N}_1 \mid i \in \operatorname{dom} hist \wedge i > 2 \wedge i \in Odd \bullet$
$(\exists interactI : Interaction;\ nextIntI : InteractRel \bullet$
$(\exists opintI : opInteraction \mid opintI = head(\{i\} \upharpoonright hist) \wedge$
$nextIntI = opintI.nextInt \bullet ((nextIntI = rchgs) \vee$
$(nextIntI = rnmsg))))$

3.5.4 Conversations: Dialog and Discussion

At this point, we specify that a conversation session involves sequences of interactions among agents.

Conversation

$session : Progress$

The start of a conversation coincides with the start of the history of interactions.

ConversationInit

$Conversation$

$\#session.hist = 1$

In a conversation, when the sender changes, the history cumulatively reflects the operation. Moreover, the operation of the last state in the history is set to change sender (*RelationChgSnd*), and the operation of the new state is set to

receiving message (*RelationRecMsg*). In order to present the change in conversation roles, the new sender agent is removed from the receiving agents set while the former sender is added to this set.

$$
\begin{array}{|l}
\textit{ConversationChangeSender} \\
\hline
\Delta \textit{Conversation} \\
\textit{sender?} : \textit{LOrgAgentInteracting} \\
\textit{receivers!} : \mathbb{P}\, \textit{LOrgAgentInteracting} \\
\textit{rchgs} : \textit{RelationChgSnd} \\
\textit{rrmsg} : \textit{RelationRecMsg} \\
\hline
\textit{sender?} \in (\textit{lastsession.hist}).\textit{agents} \\
\exists\, \textit{newState} : \textit{opInteraction}; \\
\quad \textit{senderInternalViewOfMessage} : \textit{Message} \mid \\
\quad \textit{newState.interactionIndex} = \\
\quad (\textit{last session.hist}).\textit{interactionIndex} + 1 \land \\
\quad \textit{newState.agents} = (\textit{lastsession.hist}).\textit{agents} \land \\
\quad \textit{newState.sender} = \textit{sender?} \land \\
\quad \textit{newState.sender} \neq (\textit{lastsession.hist}).\textit{sender} \land \\
\quad \textit{receivers!} = \{(\textit{lastsession.hist}).\textit{sender}\} \cup \\
\qquad (\textit{lastsession.hist}).\textit{receivers} \setminus \{\textit{sender?}\} \land \\
\quad \textit{newState.receivers} = \textit{receivers!} \land \\
\quad \textit{newState.messageSent} = \textit{sender?}.\textit{expose} \\
\qquad (\textit{sender?}.\textit{exposeActions}, \textit{senderInternalViewOfMessage}, \\
\qquad \textit{newState.interactionIndex}) \land \\
\quad \textit{newState.messagesReceived} = \varnothing \land \\
\quad \textit{newState.nextInt} = \textit{rrmsg} \bullet \\
\qquad \textit{session}'.\textit{hist} = \textit{session.hist} \oplus \{(\#\textit{session.hist} + 1, \textit{newState})\} \\
(\textit{lastsession.hist}).\textit{nextInt} = \textit{rchgs} \\
\hline
\end{array}
$$

In a conversation the same sender can send several messages. Moreover, the operation of the last state in the history is set to "new message" (*RelationNewMsg*), and the operation of the new state is set to "receiving message" (*RelationRecMsg*).

ConversationNewMessage

$\Delta Conversation$
$messageSent! : Message$
$rnmsg : RelationNewMsg$
$rrmsg : RelationRecMsg$

$\exists\, newState : opInteraction;$
$\quad senderInternalViewOfMessage : Message \mid$
$\quad newState.interactionIndex =$
$\quad (last\ session.hist).interactionIndex + 1 \land$
$\quad newState.agents = (lastsession.hist).agents \land$
$\quad newState.sender = (lastsession.hist).sender \land$
$\quad newState.receivers = (lastsession.hist).receivers \land$
$\quad newState.messageSent = newState.sender.expose$
$\quad\quad (newState.sender.exposeActions,$
$\quad\quad senderInternalViewOfMessage, newState.interactionIndex)$
$\quad \land$
$\quad messageSent! = newState.messageSent \land$
$\quad newState.messagesReceived = \varnothing \land$
$\quad newState.nextInt = rrmsg \bullet$
$\quad\quad session'.hist = session.hist \oplus \{(\#session.hist + 1, newState)\}$
$(lastsession.hist).nextInt = rnmsg$

The *ConversationReceivingMessage* schema presents the process of receiving messages in a conversation. Similarly to the case presented above, there are no changes in sender and receiver roles. The operation associated to the new state is set to "new message" (*RelationNewMsg*).

ConversationReceivingMessage

$\Delta Conversation$
$messageSent? : Message$
$rnmsg : RelationNewMsg$
$rrmsg : RelationRecMsg$

$\exists\, newState : opInteraction \mid$
$\quad newState.interactionIndex =$
$\quad (last\ session.hist).interactionIndex + 1 \land$
$\quad newState.agents = (lastsession.hist).agents \land$
$\quad newState.sender = (lastsession.hist).sender \land$
$\quad newState.receivers = (lastsession.hist).receivers \land$
$\quad newState.messageSent = messageSent? \land$
$\quad newState.messagesReceived = \{a : LOrgAgentInteracting \mid$
$\quad\quad a \in newState.receivers \bullet$
$\quad\quad a.interpretMessage(a.messageInterpretationActions,$
$\quad\quad messageSent?)\} \land newState.nextInt = rnmsg \bullet$
$\quad session'.hist = session.hist \oplus \{(\#session.hist + 1, newState)\}$

In this section we are interested in two specific types of conversation: dialog and discussion. In [SKR+94], a few characteristics of these conversation types are presented. Here, we define that these characteristics correspond to protocols. These protocols have several associated restrictions (*ProtocolConstraint*).

$$ProtocolConstraint == \mathbb{P}_1\ Attribute$$

$$ProtocolMode ::= DialogProtocol \mid DiscussProtocol$$

Protocol

$$constraints : \mathbb{P}\ ProtocolConstraint$$
$$protocolmode : ProtocolMode$$

DialogDiscussProtocol

$$dialogProt : Protocol$$
$$discussProt : Protocol$$

$$dialogProt.protocolmode = DialogProtocol$$
$$discussProt.protocolmode = DiscussProtocol$$

In addition, *SwitchProtocol* defines whether there is a change (*Switch_yes*) in protocol types or not (*Switch_no*) in a conversation.

$$SwitchProtocol ::= Switch\_yes \mid Switch\_no$$

Senge also states that in order to manage conversation sessions that follow these protocols and balance dialogs and discussions, a particular type of agent is required: the *facilitator* (*FacilitatorLOrgAg*). Initially, the variables needed to define this type are declared in the *FacLOrgAgIncludes* schema.

FacLOrgAgIncludes

$$LearningOrgAgent$$
$$enforceCoherencyWithSubjectContext : Conversation \times SubjectContext \rightarrow Actions$$
$$enforceCoherencyWithProtocol : Conversation \times DialogDiscussProtocol \rightarrow Actions$$
$$enforceCoherencyOfMessage : Message \times SubjectContext \rightarrow Consistency$$
$$enforceAdherenceOfMessage : Message \times \mathbb{P}\ ProtocolConstraint \rightarrow Consistency$$
$$currentSubjectContext : SubjectContext$$
$$currentProtocolMode : ProtocolMode$$
$$switchProtocolDecision : SwitchProtocol$$

Here this agent type is modelled as a *LearningOrgAgent*, with additional capabilities (*FacLOrgAgCapabilities*) so that the agent is capable of enforcing that all exchanged messages are coherent with the conversation context (*FacLOrgAgEnforceCoherency*), and also that all rules associated to the particular protocol type in use in a given instant are observed (*FacLOrgAgEnforceAdherence*). The underlying assumption here is that all agents involved in these sessions know these protocols. Furthermore, a session may be changed from dialog to discussion mode, and vice-versa.

FacLOrgAgCapabilities

FacLOrgAgIncludes

$$\exists\, r : Role \mid r \in roles \bullet r.ident = facilitator$$
$$\mathrm{ran}\ enforceCoherencyWithSubjectContext \subset \{capableof\}$$
$$\mathrm{ran}\ enforceCoherencyWithProtocol \subset \{capableof\}$$
$$(mapset\ first)(\mathrm{dom}\ enforceCoherencyWithSubjectContext) = (mapset\ first)(\mathrm{dom}\ enforceCoherencyWithProtocol)$$
$$currentProtocolMode = DialogProtocol \lor currentProtocolMode = DiscussProtocol$$
$$(mapset\ second)(\mathrm{dom}\ enforceCoherencyWithSubjectContext) = \{currentSubjectContext\}$$

FacLOrgAgEnforceCoherency

FacLOrgAgIncludes

$$\forall\, conv : Conversation \mid conv \in (mapset\ first)(\mathrm{dom}\ enforceCoherencyWithSubjectContext) \bullet (\forall\, subj : SubjectContext \mid subj \in (mapset\ second)(\mathrm{dom}\ enforceCoherencyWithSubjectContext) \bullet (\forall\, int : Interaction \mid \langle int \rangle \ \mathsf{in}\ conv.session.hist \bullet (\exists\, msg : Message;\ ac : \mathbb{P}\, Action \mid msg = int.messageSent \bullet (((enforceCoherencyOfMessage(msg, subj) = yes) \Rightarrow (ac = \varnothing)) \land ((enforceCoherencyOfMessage(msg, subj) = no)) \Rightarrow (ac \neq \varnothing))))))$$

FacLOrgAgEnforceAdherence

FacLOrgAgIncludes

$$
\begin{array}{l}
\forall\, conv : Conversation \mid conv \in (mapset\, first)(\\
\quad \text{dom}\, enforceCoherencyWithProtocol) \bullet \\
\quad (\forall\, dgp : DialogDiscussProtocol \mid dgp \in (mapset\, second) \\
\qquad (\text{dom}\, enforceCoherencyWithProtocol) \bullet \\
\qquad\quad (\forall\, int : Interaction \mid \langle int \rangle \ \mathsf{in}\ conv.session.hist \bullet \\
\qquad\qquad\quad (\exists\, msg : Message;\ ac : \mathbb{P}\, Action \mid msg = \\
\quad int.messageSent \bullet (\\
\qquad\qquad\qquad ((enforceAdherenceOfMessage(msg, \\
\qquad\qquad\qquad\quad dgp.dialogProt.constraints) = yes \Rightarrow \\
\qquad\qquad\qquad\quad (ac = \varnothing)) \wedge \\
\qquad\qquad\qquad ((enforceAdherenceOfMessage(msg, \\
\qquad\qquad\qquad\quad dgp.dialogProt.constraints) = no) \Rightarrow \\
\qquad\qquad\qquad\quad (ac \neq \varnothing)) \wedge \\
\qquad\qquad\qquad ((enforceAdherenceOfMessage(msg, \\
\qquad\qquad\qquad\quad dgp.discussProt.constraints) = yes) \Rightarrow \\
\qquad\qquad\qquad\quad (ac = \varnothing)) \wedge \\
\qquad\qquad\qquad ((enforceAdherenceOfMessage(msg, \\
\qquad\qquad\qquad\quad dgp.discussProt.constraints) = no) \Rightarrow \\
\qquad\qquad\qquad\quad (ac \neq \varnothing))))))
\end{array}
$$

Based on the schemes presented above, we now specify the *facilitator* agent. We note that coherency and adherence violations require the *facilitator* agent to take some corrective actions, which we do not present in detail in this work.

$$
\begin{array}{l}
FacilitatorLOrgAg \mathrel{\widehat{=}} FacLOrgAgCapabilities \wedge \\
\quad FacLOrgAgEnforceCoherency \wedge FacLOrgAgEnforceAdherence
\end{array}
$$

Now that we have defined the *facilitator* agent, we can specify dialogs and discussions. A dialog is a type of conversation, with a specific subject and participation of a *facilitator* agent. Similarly, a discussion is a type of conversation, with a specific subject and a *facilitator* agent.

Dialog

$$
\begin{array}{l}
Conversation \\
subjectContext : SubjectContext \\
facilitatorAgent : FacilitatorLOrgAg \\
\hline
facilitatorAgent.currentSubjectContext = subjectContext \\
facilitatorAgent.currentProtocolMode = DialogProtocol
\end{array}
$$

Discussion

$$
\begin{array}{l}
Conversation \\
subjectContext : SubjectContext \\
facilitatorAgent : FacilitatorLOrgAg \\
\hline
facilitatorAgent.currentSubjectContext = subjectContext \\
facilitatorAgent.currentProtocolMode = DiscussProtocol
\end{array}
$$

A meeting may involve dialog or discussion, or both. Furthermore, there is an upper bound on the number of times that an agent may act as sender (*individualSenderLimit*).

DialogDiscussionSession

$$
\begin{array}{l}
Dialog \\
Discussion \\
currentProtocolMode : ProtocolMode \\
switchProtocol : SwitchProtocol \\
durationLimit : \mathbb{N}_1 \\
individualSenderLimit : \mathbb{N}_1 \\
\hline
currentProtocolMode = facilitatorAgent.currentProtocolMode \\
switchProtocol = Switch\_yes \Rightarrow \\
\quad currentProtocolMode = \textbf{if } currentProtocolMode = DialogProtocol \\
\qquad \textbf{then } DiscussProtocol \\
\qquad \textbf{else } DialogProtocol
\end{array}
$$

At any stage, after the start of interactions, the group may define the duration of the session. Here, we assume that the duration corresponds to an upper bound on the number of interactions and that this bound must be set to a value greater than the number of participants.

$$
\begin{array}{l}
groupDefineDurationLimit : (\mathbb{P}\, LOrgAgentInteracting) \\
\quad \pfun \mathbb{N}_1 \\
\hline
\forall\, g : \mathbb{P}\, LOrgAgentInteracting \bullet \\
\quad groupDefineDurationLimit\,(g) \geq \#g
\end{array}
$$

DialogDiscussionSessionDefineDurationLimit

$\Delta DialogDiscussionSession$

$session' = session$
$subjectContext' = subjectContext$
$facilitatorAgent' = facilitatorAgent$
$currentProtocolMode' = currentProtocolMode$
$switchProtocol' = switchProtocol$
$\#session.hist \geq 1$
let $ags == (head\ session.hist).agents \bullet (\#ags \neq 0 \land$
$durationLimit' = groupDefineDurationLimit\,(ags) \land$
$individualSenderLimit' = durationLimit'\ \mathsf{div}\ \#ags)$

In a complete session involving dialog and discussion, the number of interactions must be greater than or equal to the number of participants in order to allow each agent to perform the sender role. In addition, all participants agree in finishing the session. We assume also, as a simplification hypothesis, that the group remains the same during all the session.

DialogDiscussionComplete

$\Delta DialogDiscussionSession$
$\Delta FacilitatorLOrgAg$

$facilitatorAgent' = facilitatorAgent$
$subjectContext' = subjectContext$
$currentProtocolMode' = facilitatorAgent'.currentProtocolMode$
$\#session.hist > 1$
let $ags == (head\ session.hist).agents \bullet \#session.hist \geq \#ags$
let $ags == (head\ session.hist).agents \bullet (\forall\, a1 : LOrgAgentInteracting \mid$
$a1 \in ags \bullet (\exists\, i : \mathbb{N} \mid i \in \mathrm{dom}\, session.hist \bullet$
$(\exists\, int : opInteraction \mid int = head(\{i\} \rhd session.hist) \bullet$
$int.sender = a1)))$

At any stage after the start of interactions the group may define that the session is finished.

$DDSessionUnfolding ::= DDSessionEnd \mid DDSessionAdvance$

$groupDefineSessionEnd : (\mathbb{P}\, LOrgAgentInteracting) \pfun$
$DDSessionUnfolding$

DialogDiscussionSessionTermination

$\Delta DialogDiscussionSession$

$session' = session$
$subjectContext' = subjectContext$
$facilitatorAgent' = facilitatorAgent$
$currentProtocolMode' = currentProtocolMode$
$switchProtocol' = switchProtocol$
$\#session.hist \geq 1$
let $ags == (headsession.hist).agents \bullet groupDefineSessionEnd\,(ags) =$
$DDSessionEnd$

After the process of protocol change, the *facilitator* informs the new protocol.

$facilitatorChgProt : FacilitatorLOrgAg \pfun ProtocolMode$

$\forall\, af : FacilitatorLOrgAg \mid af \in (\mathrm{dom}\, facilitatorChgProt) \land$
$af.switchProtocolDecision = Switch\_yes \bullet$
$((af.currentProtocolMode =$ **if** $af.currentProtocolMode =$
$DialogProtocol$
then $DiscussProtocol$
else $DialogProtocol) \land$
$\mathrm{ran}\, facilitatorChgProt = \{af.currentProtocolMode\})$

A change in the protocol mode requires the participation of the *facilitator* agent.

DialogDiscussionProtocolChange

$\Delta DialogDiscussionSession$

$session' = session$
$subjectContext' = subjectContext$
$facilitatorAgent' = facilitatorAgent$
$switchProtocol' = switchProtocol$
$\#session.hist \geq 1$
$currentProtocolMode' = facilitatorChgProt\ facilitatorAgent$

As the dialog advances, the *facilitator* tries to ensure that no agent performs the sender role all the time. In order to do this, we consider here that the quotient between duration and the number of agents in the group is an upper bound to the number of times a given agent may perform the sender role. If a given agent tries to exceed this limit, the group may decide to change sender or to allow the same agent to send its message. In this case the group may increase the duration of the session, otherwise some agent may not be able to send its messages before the end of the session.

$$groupDefineSender : (\mathbb{P}\, LOrgAgentInteracting) \nrightarrow LOrgAgentInteracting$$

$$\forall\, g : \mathbb{P}\, LOrgAgentInteracting \bullet (\exists\, ag : LOrgAgentInteracting \mid ag \in g \bullet groupDefineSender(g) = ag)$$

Finally, the development of a conversation session involving dialog and discussion, includes changes in agents performing the sender role, and also some of the sender agents sending several messages.

DialogDiscussionDevelopment

$$\Delta DialogDiscussionSession$$
$$ConversationChangeSender$$
$$ConversationReceivingMessage$$
$$ConversationNewMessage$$

$$facilitatorAgent' = facilitatorAgent$$
$$subjectContext' = subjectContext$$
$$session' = session$$
$$subjectContext' = subjectContext$$
$$currentProtocolMode' = currentProtocolMode$$
$$switchProtocol' = switchProtocol$$
$$\#session.hist \geq 1$$
$$\textbf{let}\ ags == (lastsession.hist).agents \bullet (\#ags \neq 0 \land sender? = groupDefineSender\,(ags) \land (((last\ session.hist).sender \neq sender? \land ConversationChangeSender) \lor ConversationNewMessage) \land durationLimit' = groupDefineDurationLimit\,(ags) \land individualSenderLimit' = durationLimit'\ \mathsf{div}\ \#ags)$$

4 Properties of the Formal Model

In this section we investigate the characteristics of the model presented in the previous sections.

4.1 Definitions

Initially we introduce several definitions that allow us to investigate what are the features of the model.

Let a_j and a_i be agents that interact with each other.

$$a_i, a_j : LOrgAgentInteractions$$

Let m_i and m_j be messages that contain information regarding some context (current state of affairs) or proposals to change this state (goals).

$$m_i, m_j : Message$$

Let v_i and v_j be of type *View*. These views result from interpretations performed by agents a_i and a_j for each message m_i and m_j, respectively.

$$v_i, v_j : View$$

Let sm_i and sm_j be sequences of messages.

$$sm_i, sm_j : \text{seq}\, Message$$

Let ind_i and ind_j be indexes of interactions.

$$ind_i, ind_j : \mathbb{N}_1$$

Let $goal_s$ be a set of goals.

$$goal_s : \mathbb{P}\, Goal$$

1. A message m_i is consistent with the mental models of an agent a_i if it does not contain predicates that are logically inconsistent with the predicates contained in the mental models of a_i.

$$\begin{array}{l} consistentMessage : LOrgAgentInteractions \times Message \rightarrow \\ \quad Consistency \\ \hline \forall\, ag : LOrgAgentInteractions;\ m_i : Message \bullet \\ \quad (\exists\, internalViewOfMessage : Message \bullet \\ \quad (consistentMessage(ag, m_i) = yes \Leftrightarrow \\ \quad (internalViewOfMessage = (ag.produceMessage \\ \qquad ag.messageProductionActions) \wedge \\ \qquad ag.knowsAView(ag.knowingActions, \\ \qquad\quad internalViewOfMessage) = yes \wedge \\ \qquad m_i = ag.expose(ag.exposeActions, \\ \qquad\quad internalViewOfMessage, ind_i) \wedge \\ \qquad ag.knowsAView(ag.knowingActions, m_i) = yes))) \end{array}$$

2. An agent a_i is *trustAgent* if it exposes only messages that are consistent with its mental models, ie, a_i exposes only information that it believes or knows. Otherwise, the agent a_i is *nonTrustAgent*.

$$\begin{array}{l} trustAgent : LOrgAgentInteractions \rightarrow Consistency \\ nonTrustAgent : LOrgAgentInteractions \rightarrow Consistency \\ \hline \forall\, ag : LOrgAgentInteractions;\ m_i : Message \bullet \\ \quad (trustAgent\ ag = yes \Leftrightarrow consistentMessage(ag, m_i) = yes) \\ \forall\, ag : LOrgAgentInteractions \bullet (\exists\, m_i : Message \bullet \\ \quad (nonTrustAgent\ ag = yes \Leftrightarrow \\ \quad (\neg\, (consistentMessage(ag, m_i) = yes)))) \end{array}$$

3. An interaction among agents a_i and a_j is *TRUST* if during this interaction a_i has in its mental models a *trustAgent* model of a_j and, reciprocally, if a_j has in its mental models a *trustAgent* model of a_i. Conversely, an interaction is *NOTTRUST* if one of either a_i or a_j, or both do not have in its mental models a *trustAgent* model of its respective partner.

TrustInteraction

Interaction

$$
\begin{array}{l}
\#agents = 2 \\
\#receivers = 1 \\
\exists\, model1, model2 : LOrgAgentInteractionsModel; \\
\quad trustModel1, trustModel2 : TrustLOrgAgentInterModel \bullet \\
\quad (\exists_1\, ag : LOrgAgentInteractions \bullet \\
\quad (ag.modelOfAgentInteractions\,(sender) = model1 \land \\
\quad sender.modelOfAgentInteractions\,(ag) = model2 \land \\
\quad model1 = trustModel1 \land \\
\quad model2 = trustModel2 \land \\
\quad trustModel2 \in \\
\quad sender.mentalmodels.modelTrustLOrgAgInts \land \\
\quad trustModel1 \in ag.mentalmodels.modelTrustLOrgAgInts))
\end{array}
$$

NonTrustInteraction

Interaction

$$
\begin{array}{l}
\#agents = 2 \\
\#receivers = 1 \\
\exists\, model1, model2 : LOrgAgentInteractionsModel; \\
\quad trustModel1, trustModel2 : NonTrustLOrgAgentInterModel \\
\quad \bullet\, (\exists_1\, ag : LOrgAgentInteractions \bullet \\
\quad (ag.modelOfAgentInteractions\,(sender) = model1 \land \\
\quad sender.modelOfAgentInteractions\,(ag) = model2 \land \\
\quad ((model1 = trustModel1 \land \\
\qquad trustModel1 \in \\
\qquad ag.mentalmodels.modelNonTrustLOrgAgInts) \lor \\
\qquad (model2 = trustModel2 \land \\
\qquad\quad trustModel2 \in \\
\qquad\quad sender.mentalmodels.modelNonTrustLOrgAgInts))))
\end{array}
$$

4. An agent a_i is *believer* if, in the absence of previous interactions with a particular a_j, a_i assumes by default a *trustAgent* model of a_j.

BelieverLOrgAgentInter

LOrgAgentInteractions

$$\forall\, ag : LOrgAgentInteractions \mid ag \notin \text{dom}\ modelOfAgentInteractions \bullet (\exists_1\, trustModel : TrustLOrgAgentInterModel \bullet (modelOfAgentInteractions = modelOfAgentInteractions \cup \{(ag \mapsto trustModel)\} \land trustModel \in ag.mentalmodels.modelTrustLOrgAgInts))$$

5. An agent a_i is *skeptical* if,in the absence of previous interactions with a particular a_j, a_i assumes by default a *nonTrustAgent* model of a_j.

SkepticalLOrgAgentInter

LOrgAgentInteractions

$$\forall\, ag : LOrgAgentInteractions \mid ag \notin \text{dom}\ modelOfAgentInteractions \bullet (\exists_1\, trustModel : NonTrustLOrgAgentInterModel \bullet (modelOfAgentInteractions = modelOfAgentInteractions \cup \{(ag \mapsto trustModel)\} \land trustModel \in ag.mentalmodels.modelNonTrustLOrgAgInts))$$

6. An agent a_i revises its *trustAgent* model of a_j to a *nonTrustAgent* model of a_j if a_i is able to perceive that there is no consistency between a_j exposed mental models, proposals (messages), agreements, and a_j actions.
7. An agent a_i revises its *nonTrustAgent* model of a_j to a *trustAgent* model of a_j, if a_j is able to justify to a_i any perceived inconsistency between its exposed mental models, proposals (messages), agreements, and a_j actions.
8. A dialog/discussion session involving agents a_i and a_j successfully produces a set $goal_s$ of shared goals if at the end of the session all these holds:
   (a) a_i believes in $goal_s$ and
   (b) a_j believes in $goal_s$ and
   (c) a_i believes that a_j believes in a_j's interpretation of $goal_s$ and
   (d) a_j believes that a_i believes in a_i's interpretation of $goal_s$

4.2 Properties

4.2.1 *TRUST* Is Necessary for the Formalized Fifth Discipline

We argue that it is not possible to build a shared set of goals (similarly for visions, values, purposes) if the agents involved in the process do not trust each other.

The context of analysis involves two agents a_i and a_j and an initial sequence of sending and receiving messages performed by both agents in order to formulate a set of goals which both agree upon. Our initial hypotheses include:

($h0$). We suppose that there are no records of previous interactions among a_i and a_j.

($h1$). We suppose that both a_i and a_j are *believer* agents.

Let us consider the following process:

Agent a_i sends message m_i. We follow the *ReceivingMessage* schema to investigate what agent a_j knows after receiving m_i.

First, a_i has exposed message m_i.

$$\exists\, senderInternalViewOfMessage : Message \bullet \\ m_i = a_i.expose(a_i.exposeActions, \\ senderInternalViewOfMessage, ind_i)$$

Then, the message received is interpreted by a_j.

$$\exists\, messRec : Message \bullet \\ (messRec = a_j.interpretMessage \\ (a_j.messageInterpretationActions, m_i) \wedge \\ a_j.knowsAView(a_j.knowingActions, messRec) = yes$$

According to ($h1$), a_j is a *believer* agent, therefore it believes that a_i believes in m_i.

$$a_j.knowsAView(a_j.knowingActions, \\ a_i.knowsAViewState(a_i.knowingActions, m_i)) = yes$$

Thus, a_j believes that a_i believes in m_i, ie, that a_i believes that the state of affairs in the environment corresponds to the information contained in m_i and a_j may proceed and drive its reasoning based on this belief. After this step, a_j may send message m_j regarding this dialog/discussion session. Hence, in a similar way, a_i receives m_j and knows:

$$\exists\, messRec : Message \bullet \\ (messRec = a_i.interpretMessage \\ (a_i.messageInterpretationActions, m_j) \wedge \\ a_i.knowsAView(a_i.knowingActions, messRec) = yes \wedge \\ a_i.knowsAView(a_i.knowingActions, \\ a_j.knowsAViewState(a_j.knowingActions, m_j)) = yes$$

Again, a_i believes that a_j believes in what is stated in m_j. We may assume that this process proceed through several interactions involving negotiation/argumentation, so that when the session is considered finished by both a_i and a_j, each agent will have a richer model about the beliefs of the corresponding partner.

Regarding the set of shared goals, there are two possibilities at the end of the session: a set was built, or there was no agreement about shared goals.

If we suppose that there was an agreement about a set $goal_s$ of goals, then:

$$a_i.knowsAView(a_i.knowingActions, (\bigcup(goal_s))) = yes$$
$$a_j.knowsAView(a_j.knowingActions, (\bigcup(goal_s))) = yes$$

Let ind_{if} and ind_{jf} be, respectively the indexes of the final interaction among agents a_i and a_j; $msgoal_i$ and $msgoal_j$ messages that confirm that a_i and a_j agree with $goal_s$; and $vsgoal_i$ and $vsgoal_j$ interpretations that a_i and a_j perform on the respective messages.

$$\begin{array}{l}
msgoal_i, msgoal_j : Message \\
vsgoal_i, vsgoal_j : View \\
ind_{if}, ind_{jf} : \mathbb{N}_1 \\
\hline
\exists\, senderInternalViewOfMessage : Message \bullet \\
\quad msgoal_i = a_i.expose(a_i.exposeActions, \\
\quad senderInternalViewOfMessage, ind_{if}) \land \\
\quad (vsgoal_j = a_j.interpretMessage \\
\quad (a_j.messageInterpretationActions, msgoal_i) \land \\
\quad a_j.knowsAView(a_j.knowingActions, vsgoal_j) = yes \land \\
\quad a_j.knowsAView(a_j.knowingActions, \\
\qquad a_i.knowsAViewState(a_i.knowingActions, msgoal_i)) = yes \\
\exists\, senderInternalViewOfMessage : Message \bullet \\
\quad msgoal_j = a_j.expose(a_j.exposeActions, \\
\quad senderInternalViewOfMessage, ind_{jf}) \land \\
\quad (vsgoal_i = a_i.interpretMessage \\
\quad (a_i.messageInterpretationActions, msgoal_j) \land \\
\quad a_i.knowsAView(a_i.knowingActions, vsgoal_i) = yes \land \\
\quad a_i.knowsAView(a_i.knowingActions, \\
\qquad a_j.knowsAViewState(a_j.knowingActions, msgoal_j)) = yes
\end{array}$$

Then, what is known to each agent at this stage corresponds to:

$$\begin{array}{ll}
(k1) & a_j.knowsAView(a_j.knowingActions, \\
& a_i.knowsAViewState(a_i.knowingActions, \\
& \quad a_i.expose(a_i.exposeActions, senderInternalViewOfMessage_i, \\
& \quad ind_{if}))) = yes
\end{array}$$

$$\begin{array}{ll}
(k2) & a_i.knowsAView(a_i.knowingActions, \\
& a_j.knowsAViewState(a_j.knowingActions, \\
& \quad a_j.expose(a_j.exposeActions, senderInternalViewOfMessage_j, \\
& \quad ind_{jf}))) = yes
\end{array}$$

If a_i and a_j at the end of the session still model the corresponding partner as *trustAgent*, then we can affirm that a_i believes that a_j believes in its interpretation $vsgoal_j$ of the goal $goal_s$. If, however, at the end of the session, at least one of a_i or a_j model the respective partner as *nonTrustAgent*, then one of $(k1)$ or $(k2)$ will not hold, and therefore it will not be possible to build a $goal_s$ set.

4.2.2 Agents Must Be *trustAgent*

Initially, we have to refer back to the type *ExposedBelief*, which is a type of the agent's beliefs. These beliefs are stored in the agent's *store*. Thus, following the function *advocacy*, defined in the *IntraPersonalMentalModel* schema, the agent has motivations and performs actions so as to expose its beliefs and reasoning. Therefore, the agent in our model is a *trustAgent*, ie, it exposes only information that it believes or knows.

4.2.3 Agents' Motivations and Disciplines Must Be Consistent

Here we show that the agents' development of the disciplines depends on their motivations.

First, in our formalization of mental models, the *reflection* and *advocacy* functions depend on the agent's motivations. In the case of the personal mastery discipline, the agent's *personalvision*, *developguidingideas*, *enhancerealityvisions*, *clarifypersonalvisions*, and *creativetension* functions are all dependent on the agent's motivations.

The formalization of the team learning discipline introduces a set of actions and protocols, among them *inquiry* and *advocacy* which, as showed above in the study of mental models, are influenced by the agent's motivations. However, there is also the function *developdialogdiscussactions*, which also depends on the motivations of the agent.

As for the systems thinking discipline, we have the *potentialstatesanalyser* function which depends on *PersonalVision* which, in its turn, depends on the agent's motivations.

Finally, the shared vision discipline is developed by teams of agents that construct agreements via *dialogdiscuss* function, which uses *dialogdiscussactions* actions. The development of these actions depend on the agents' motivations. Thus, in summary, an agent that develops the five disciplines must have motivations that are consistent with the actions and protocols defined in each of the mentioned disciplines.

4.2.4 Agents Must Be *tenacious*

Following the specification of the *LearningOrgAgent* schema, such type of agent develops the five disciplines independently of any temporal considerations, ie, the disciplines determines the way the agent thinks, acts and interacts. This is a consequence of the fact that for every interaction (with the environment or with another agent) the *learningorgperceives* function is used to produce the agent's percepts. This function is affected by the agent's mental models and guiding ideas. Guiding ideas result from the agent's personal mastery and include vision, purpose and values. Accordingly, the agent's action selection function, *learnorgact*, is affected by mental models, personal mastery, and systems thinking. In addition, the learning organization model, presented in *LearnOrg*, requires that every team in the organization is a learning team and the organi-

zation must develop a set of shared visions. Therefore, all disciplines must be continuously developed.

4.2.5 Agents Must Be *cooperative*

According to [Wei01], a cooperation is a type of coordination among non antagonistic agents in which the participants succeed or fail together. In contrast, in a competition the success of one participant implies the failure of others.

In our formalization of the Mental Models discipline we require that the agent exposes its beliefs so that in a team of agents, each agent learn the others' mental models. This is also true for building shared vision.

If we consider that an "index" of success of a learning organization is to what degree it is able to develop shared visions, and that this is a collective achievement, then we conclude, that in fact, the learning organization is a cooperative scenario.

In a competitive scenario an agent exposes its mental models only if it believes that by doing so it will have some type of benefit.

4.2.6 Organizational Change Emerges from Individual's Motivations

According to our specification, all agents in the organization develop the five disciplines and the disciplines determine how the agents think, act and interact. We also know that the development of the disciplines by a given agent is influenced by its motivations. Moreover, we have shown that shared goals, values, and visions, may emerge from interactions among agents and also depend on the agents development of the disciplines. If we consider that one of the indicators of a given organizational state is its current set of shared goals, values, and visions; and that the outcomes produced by the organization result from agent's actions, we conclude that organizational change emerge from individual motivations.

4.2.7 Low Turn-Over Is Required in the Organization

The interaction process described in the previous section presents a situation where there are no changes in the agents that are members of the team during all the process. However, if we take changes in team membership into account, it is possible to observe that the interaction process underlying the construction of shared vision and shared mental models is affected as follows.

Let us consider a team with n members.

We have shown that at least n rounds of interactions are required so that each agent has the opportunity to play the role of sender.

Suppose that in round k, with $k < n$, k agents have already played the sender role when a new agent enters the group. Now the group has $n + 1$ members and at least $n + 1$ rounds would be required to allow the new member to play the sender role. However, in this case the new member also needs to play the receiver role for all the k presentations that occurred before he was a member

of the team. Thus, at least $k + n + 1$ rounds are required for each new agent that becomes a member of the team in a given round k.

Now, considering the case where the new agent substitutes a former member of the team, we notice that $k + n$ rounds are required for each new agent that substitutes a member of the team in a given round k.

Hence, the more frequent inclusions or substitutions of members of a team, the less efficient is the process of producing common knowledge via interactions.

In fact, for each agent in the team a certain amount of time is spent in modeling the others mental models and visions. If there is a high frequency of turn-over in the teams, all this effort may be wasted.

4.2.8 Bounded Number of Members in Learning Teams

In order to improve the efficiency of the interaction process, the number of members in a given team must be bounded. Otherwise, teams with a large number of members will have to spend too much time deliberating, even if the agent population is constant[11]. Thus, in order to cope with this complexity, the learning organization must be subdivided in a number of teams and must also define an upper limit for the size of its teams.

5 Related Work

Formalizations for the organizational theory Organization in Action (OA) [Tho67] are presented in [KP99] and [MH96]. Similarly to the work reported in this paper, parts of a discursive theory are revised using formal methods. In [KP99], first order predicate logic is used to study the underlying argumentation structure for the propositions of the OA theory. On the other hand, a multi-agent modal logic developed by the authors in [MH96], is used to achieve goals that are similar to the ones mentioned above, and, additionally, to investigate the expressive power and applicability of this logic for the formalization of a discursive theory. The formal model presented in this paper uses the Z notation to build a structured framework that can be used to study Senge's theory and also to investigate hybrid organizations, involving both human and computational agents.

In addition, in [PCG98] and [CP94], several works related to organizational computational simulation are reported. All these works involve the construction of computational models related to organizations or organizational theories. Below, some of these works are mentioned.

The focus of the work presented in [KWW98] is the decision process in a team. A team corresponds to a group that have a goal or purpose in common. The team is made of different and inter-dependent members with shared leadership. Also, members have individual and collective responsibilities and assignment of activities. In the team decision process the different capabilities

[11] The population is constant if turn-over is equal to zero and new members are not admitted.

of its members are used, often involving the division of the problem into smaller subproblems which later will be integrated to construct the team's solution. In comparison, in the Fifth Discipline Model, we define three types of collections of agents: group, team and learning team. In a group resources are shared among its members. In a team, which corresponds to a refinement for the group type, resources and goals are shared. In a learning team, these features are also present, and there are also guiding ideas and plans that are shared. Shared guiding ideas include shared vision, shared purpose, and shared values. In Senge's theory [Sen90] collective decision processes are not presented. Therefore, the formal model introduced in our work does not specify decision processes. In the model, what is specified for each agent is the action selection function. Each member of a learning team is an agent of type *LearningOrgAgent* and the function *learnorgact* defines the next action selected by this type of agent. This function is influenced by the agent's goals. However, the goal adoption process defined in the formal model (*LearningOrgAgentAdoptGoals*) shows that the agent only adopt the goals that satisfy the agent's motivations and that are consistent with its vision, purpose and values. Moreover, the shared guiding ideas developed in a learning team corresponds to a set involving vision, purpose and values that is consistent with each member's vision, purpose and values. As a consequence, agents in a learning team adopt goals that are consistent with the shared vision and select their next actions based on such goals.

In [Lin98] the investigation and design of organizations that require a high level of reliability is presented. In that work, simulation models are tools to examine different possibilities in the process of structuring organizations. The effect of external conditions in the decision process performance is studied. The approach presented in that work is a generalization of the Contingency Theory [LL67] of administration: the effect of the environment on the organizational performance and the corresponding implications on its structure. A computational model is used to analyze several factors that affect organizational performance in dynamic environments. The investigation of organizational structures is beyond the scope of this work, as Senge does not specify them in detail in his theory. In the model presented here, we decided to specify the organizational structure as a graph in which each team is represented by a node, and relationships among teams are represented by edges. However, in this model different organizational structures could have been specified instead.

The social dilemma involving voluntary cooperation among individuals confronted with conflicting time and effort options is the focus of the work reported in [HG98]. The individual can contribute to build a common good or, alternatively, it may decide to take advantage of the efforts of the other individuals. This is a fundamental question for the study of cooperative behavior in organizations. In the model presented in this paper, the shared vision discipline involves a process in which shared high level goals have to emerge. In this process the motivation of each agent underlies its decision to adopt a set of shared goals. The existence of conflicting time and effort options for a given agent could imply the absence of a motivation for this agent to adopt a shared goal or shared vision. However, in our formal model a learning organization has to develop a

shared vision, in a process that involves all agents in the organization. Therefore, confronted with such dilemmas, in our model for the Fifth Discipline the agents choose to cooperate.

In [CP98] a type of agent, called *WebBot* is investigated. This agent performs tasks autonomously, acting as an assistant for humans or other *WebBots*). That work studies the effect of the honesty of the *WebBot* concerning the individual and collective organizational behavior. The tests reported in that paper present two types of organizations. In the first type all *WebBots* are honest. Conversely, in the second all *WebBots* are dishonest. The results of these tests show that, after some time, honest *WebBots* ask more questions and learn more than dishonest *WebBots*. Similarly, in our model one of the features that plays an important role is the honesty of the agent in its interactions in a learning organization. In fact, we show that in this model trust interactions among agents are necessary to build a learning organization.

The last paper in [PCG98] that we mention here is the work of [SD98]. Organizational design and re-design in organizations that include human and computational agents is the main subject of that work. Two perspectives inspired by organizational theories are used to define a framework that describes the problem involving the structuring of organizations: Contingency Theory [LL67] and Socio-Technical Theory [Tri81]. Analogously, in our model both human and computational agents can be represented. However, dynamic organizational restructuring is beyond the scope of our work.

The goal of the work presented in [YS99] is to study a role based agent-oriented conceptual framework in order to model workflow. In that case, business processes are viewed as a collection of problem solvers autonomous agents that interact when faced with interdependencies. Furthermore, a workflow is modeled as a set of related roles. Role definition is based on several attributes, like: goals, skills, obligations, permissions, and protocols. Protocols define the interactions among roles. Roles are assigned to agents based on an assessment of the agent's qualifications and skills. The agent's behavior is a consequence of its mental states, like intentions, beliefs, goals, skills, etc. Roles are defined as collections of duties, modeled as obligations, and rights, modeled as permissions. When a given role is assigned to a specific agent, that agent inherits the particular obligations and permissions related to that role. Workflow coordination is achieved via agent communication. In that work, the organizational model results from the definition of organizational roles and description of the coordination and agent performance while performing a given role. Therefore, that model focuses in organizational processes, while our model is based on a specific organizational theory.

In [K. 99] multi-agent learning is investigated. That work is based on an Organizational Learning approach based on [Arg77] in which four types of learning are considered. The first is called individual single loop learning and enhances performance in the scope of an individual norm. The second, individual double loop learning, enhances performance via changes in an individual norm. The third, known as organizational single loop learning improves performance in the scope of an organizational norm. Finally, the fourth is named organizational

double loop learning and improves performance via changes in an organizational norm. In that computational model, individual norms are implemented as individual knowledge and, correspondingly, organizational norms are implemented as organizational knowledge. A set of rules describes the individual knowledge and a set of individual knowledge corresponds to the organizational knowledge. Agents are implemented as Learning Classifier Systems (LCS) [Gol89] and learning in the MAS is supported by an extension to LCS: Organizational-learning oriented Classifier System - OCS, that is based on an architecture for Machine Learning - Genetics-Based Machine Learning (GBML) - and is made of several LCS systems. Similarly to the work presented in this paper, [K. 99] also presents a computational model that is inspired by an organizational theory associated to the Organizational Learning approach. In that case, however, Machine Learning and genetic algorithms techniques are used instead, to investigate the computational performance in processes involving multi-agent learning via implementation of concepts defined in an organizational theory. Nonetheless, we note that some of the concepts of [Arg77] may be mapped to concepts in Senge's theory. For example, individual and organizational double loop learning may be compared to personal mastery and shared vision, respectively, in the Fifth Discipline.

In [HSB02] a model for the specification of multi-agent organizations is presented. This model focuses on functional, structural and deontic aspects. The structure is related to the concepts of role, relationships among roles and groups. The functional aspect includes the concepts of global plans and missions that are structured in a type of goal decomposition tree. Functional and structural aspects are independent so that changes in the functional dimension do not require changes in the organizational structure. The only dimension that has to be adapted is the deontic, in order to reflect the modifications on the other two dimensions. Considering the work reported in this paper we note that the concepts of global missions and goals are similar to the concept of shared guiding ideas in our formal model. However, [HSB02] specifies a structure that associates global missions and plans to lower level goals. Instead, in our model, the development of shared guiding ideas is dynamic and results from interactions among agents.

In summary, some of the papers mentioned above deal with computational tools in general, and MAS technology in particular to model or simulate organizations, or use formal methods to investigate characteristics of a particular organizational theory. Other papers use organizational theories concepts to design models that can be used to implement computational systems or build MAS frameworks. Therefore, there are similarities between our work and the above mentioned research. However, the novelty of our work is the use of a Software Engineering method to build a formal model based on an organizational theory. Moreover, our model is founded on a MAS formal framework, so that our model represents a perspective on the implementation of the Fifth Discipline in the context of a MAS framework. This model presents a formalization for the main concepts of Senge's theory and, as a consequence, it is not restricted to particular organizational aspects like: coordination, team work, cooperation.

We also note that in our formal model there are no constraints concerning the representation of different types of agents, so that organizations composed of human agents, computational agents, or both, can be modeled.

6 Discussion

In this work, we presented an overview of the Fifth Discipline theory and introduced a formalization of this theory in the context of SMART, which is a MAS formal framework.

In this section we discuss some issues regarding the formalization process of Senge's theory and issues concerning the properties of the formal model presented here.

Initially, we point out that during the formalization process we observed some situations where the main reference for the LO theory [Sen90] presented some concepts in an ambiguous way. For example, Senge states in [Sen90, p. 147] that goals and objectives are distinct from visions. However, in another part of the book [Sen90, p. 149] he affirms that a vision corresponds to a specific destination, a concrete image of the desired future. In our work clear definitions are required for every concept, thus we consider that visions, either personal or shared, correspond to goals. Additionally, some concepts (or types) used in this formalization are not explicitly specified in Senge's theory, for example there are no details in [Sen90] or [SKR+94] concerning an agent's plans or its capabilities. Therefore, the formalization of these concepts follows an interpretation of the LO theory and its (plausible) mapping to current research in MAS.

Other issues arose involving the concepts of autonomy, creative tension, and motivation.

It is important to note the different issues involving the concept of autonomy as used in SMART and in our formalization. In the former case, autonomy depends on the fact that an agent is capable of (or has) motivations or not, so that the agent is able to generate its own goals. In our case, the *LearningOrgAgent* is autonomous but is also embedded in an organizational environment. Therefore, it has to adopt the goals associated with a given role in order to perform this role. Otherwise, as it is autonomous, it can refuse to adopt that goals, but it will also refuse to perform that given role in the organization. In addition, roles have an associated autonomy level (in an organizational sense), so that the agent is able to generate its own goals in the context of the role.

Additionally, the concept of creative tension in LO seems to be closely related to the concept of motivation in SMART. In fact, as creative tension may be viewed as a measure of the distance between the current state of affairs in the environment the reduction of creative tension can be considered as a motivation itself: in the *PersonalMastery* schema, the *creativetension* function produces goals.

Concerning the properties derived from the formal model and presented in section 4, it is interesting to note the importance that some individual characteristics play in an organization that plans to successfully implement the LO

theory. Hence, we observe that the agents must be honest, cooperative, tenacious and the interactions among agents must also be honest. Thus, there are several constraints on individual features of the formal models of the agents that are members of such type of formal organization.

It is important to note that the formal model presented in this paper corresponds to our interpretation of Senge's work and depicts some characteristics of his theory, thus is not intended to be a full detailed translation of the LO theory into a formal model. For example, most of the skills/activities related to systems thinking are encapsulated in functions. However, the level of abstraction presented in this model is appropriate to allow us to study important properties of the LO theory and discuss individual and organizational features that should be taken into account in an implementation of the LO theory, either in artificial or in natural organizations.

Additionally, we note that our use of formal methods for modeling systems in general presents new perspectives and reveals new, not yet explored, potentialities concerning the use of these methods. For example, consistency checking of an organization with regard to a specific organizational theory can be investigated.

All these issues should certainly pose a number of interesting problems and questions regarding the construction of formal models for the Fifth Discipline in particular, and for OT theories in general, and also for the study of different implementation processes of such theories in organizations. A further advancement in this direction is the development of a test case which is presented in [Sil04]. It uses parts of the model introduced in this article and was developed in the form of an animation of the specification using ZeTa [GB03].

Our future interest lies in the refinement of our model in order to computationally implement it using MAS development tools like *act*SMART [AL01, dL04] or the SACI [HS00] environment. We are also interested in investigating some aspects not covered in detail by Senge's theory, for example: planning, skills, the process of emergence of the shared vision. In this case, results from other works that present more detailed views of these aspects will be useful. For example, the process of emergence of organizational mental models based on individual mental models, as presented in [Kim93].

References

[AL01] R. Ashri and M. Luck. Towards a Layered Aproach for Agent Infrastructure: the Right Tools for the Right Job. In *Proceedings of the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, pages 9–16. http://citeseer.nj.nec.com/445235.html, 2001. 81

[Arg77] C. Argyris. Double Loop Learning in Organizations. *Harvard Business Review*, pages 115–125, Sept-Oct 1977. 3, 78, 79

[BCV98] R. H. Bordini, J. A. Campbell, and R. Vieira. Extending Ascribed Intensional Ontologies with Taxonomical Relations in Anthropological Descriptions of Multi-Agent Systems. *Journal of Artificial Societies and Social Simulation*, 1(4), 1998. 12

[CC96] R. Conte and C. Castelfranchi. Simulating Multi-Agent Interdepencies: A Two-Way Approach to the Micro-Macro Link. In G. N. Gilbert K. G. Troitzsch, U. Mueller and J. E. Doran, editors, *Social Science Microsimulation*. Springer-Verlag, Berlin, 1996. 34

[CD65] V. Cangelosi and W. Dill. Organizational Learning: Observations Toward a Theory. *Administrative Sciences Quarterly*, 10:175–203, 1965. 3

[Che80] B. Chellas. *Modal Logic: An Introduction*. Cambridge University Press: Cambridge, England, 1980. 12

[Chi00] I. Chiavenato. *Introdução à Teoria Geral da Administração. Edição Compacta*. Editora Campus Ltda, second edition, 2000. (in Portuguese). 2

[CLW99] M. Crossan, H. Lane, and R. White. An Organizational Learning Framework: from Intuition to Institution. *Academy of Management Review*, 24(3):522–537, 1999. 3

[CP94] K. M. Carley and M. J. Prietula, editors. *Computational Organization Theory*. Lawrence Erlbaum Associates, Publishers: Hillsdale, NJ, 1994. 76

[CP98] K. M. Carley and M. J. Prietula. Webbots, Trust, and Organizational Science. In M. J. Prietula, K. M. Carley, and L. Gasser, editors, *Simulating Organizations*, pages 3–22. AAAI Press/The MIT Press, 1998. 78

[dL01] M. d'Inverno and M. Luck. *Understanding Agent Systems*. Springer-Verlag, 2001. 1, 12, 14, 15, 18, 21, 40, 41, 42, 44

[dL04] M. d'Inverno and M. Luck. *Understanding Agent Systems*. Springer-Verlag, second edition, 2004. 81

[DS89] R. Duke and G. Smith. Temporal Logic and Z Specifications. *Australian Computer Journal*, 21(2):62–66, 1989. 57

[DTC97] D. Dunphy, D. Turner, and M. Crawford. Organizational Learning as the Creation of Corporate Competencies. *Journal of Management Development*, 16(4):232–244, 1997. 3

[Fay49] H. Fayol. *General and Industrial Management*. Pitman, London, 1949. 2

[FFMM94] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In N. Adam, B. Bhargava, and Y. Yesha, editors, *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, MD, USA, 1994. ACM Press. 45

[fIPA00] Foundation for Intelligent Physical Agents. FIPA ACL Message Structure Specification. url: http://www.fipa.org/, 2000. 45

[FL85] C. M. Fiol and M. A. Lyles. Organizational Learning. *Academy of Management Review*, 10(4):803–813, 1985. 3

[Gar93] D. A. Garvin. Building a Learning Organization. *Harvard Business Review*, pages 78–91, Jul-Aug 1993. 3

[GB03] W. Grieskamp and R. Büssow. The ZETA System. 2003. url: http://uebb.cs.tu-berlin.de/zeta. 81

[Gol89] E. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989. 79

[HG98] B. A. Huberman and N. S. Glance. Fluctuating Efforts and Sustainable Cooperation. In M. J. Prietula, K. M. Carley, and L. Gasser, editors, *Simulating Organizations*, pages 89–103. AAAI Press/The MIT Press, 1998. 77

[Hoa78] C. A. R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, 21(8):666–677, 1978. 12

[HS00] J. F. Hübner and J. S. Sichman. SACI: Uma Ferramenta para Implementação e Monitoração da Comunicação entre Agentes. In M. C. Monard and J. S. Sichman, editors, *IBERAMIA/SBIA 2000 open discussion track: proceedings*, pages 47–56. http://www.lti.pcs.usp.br/saci/doc/iberamia2000-saci.pdf, 2000. 81

[HSB02] J. F. Hübner, J. S. Sichman, and O. Boissier. A Model for the Structural, Functional, and Deontic Specification of Organizations

in Multiagent Systems. In G. Bittencourt and G. L. Ramalho, editors, *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence (SBIA'02)*, LNAI 2507, pages 118–128, Porto de Galinhas, PE, Brazil, 2002. Springer. 79

[Jac97] J. Jacky. *The Way of Z: Practical Programming with Formal Methods*. Cambridge University Press, 1997. 1, 12, 87

[Jon90] C. B. Jones. *Systematic Software Development using VDM*. Prentice Hall, second edition, 1990. 12

[K. 99] K. Takadama and T. Terano and K. Shimohara. Can multiagents learn in organization? – analyzing organizational learning-oriented classifier system. In *IJCAI'99 Workshop on Agents Learning about, from and other Agents*. http://citeseer.nj.nec.com/123628.html, 1999. 78, 79

[Kim93] D. Kim. The Link Between Individual and Organizational Learning. *Sloan Management Review*, 35(1):37–50, 1993. 3, 81

[KP99] J. Kamps and L. Pólos. Reducing Uncertainty: A Formal Theory of Organizations in Action. *American Journal of Sociology*, 104:1776–1812, 1999. 76

[KWW98] M. C. Kang, L. B. Waisel, and W. A. Wallace. Team Soar: A Model for Team Decision Making. In M. J. Prietula, K. M. Carley, and L. Gasser, editors, *Simulating Organizations*, pages 23–45. AAAI Press/The MIT Press, 1998. 76

[Lin98] Z. Lin. The Choice Between Accuracy and Errors: A Contingency Analysis of External Conditions and Organizational Decision Making Performance. In M. J. Prietula, K. M. Carley, and L. Gasser, editors, *Simulating Organizations*, pages 67–87. AAAI Press/The MIT Press, 1998. 77

[LL67] P. R. Lawrence and J. W. Lorsch. *Organization and Environment*. Harvard University Press, Cambridge MA, 1967. 77, 78

[LyLLd01] F. Lopez, y Lopez, M. Luck, and M. d'Inverno. A Framework for Norm-Based Inter-Agent Dependence. In *Proceedings of The Third Mexican International Conference on Computer Science. SMCC-INEGI*, pages 31–40. http://citeseer.nj.nec.com/lopez01framework.html, 2001. 12

[MH96] M. Masuch and Z. Huang. A Case Study in Logical Deconstruction: Formalizing J. D. Thompson's Organizations in Action in a Multi-Agent Action Logic. *Computational and Mathematical Organization Theory*, 2(2):71–114, 1996. 76

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989. 12

[MLd03] S. Munroe, M. Luck, and M. d'Inverno. Towards Motivation-Based Decisions for Worth Goals. To appear as a paper in The Third International/Central and Eastern European Conference on Multi-Agent Systems - CEEMAS 2003. Prague, Czech Republic, 6 2003. 12

[MSB99] J. Morabito, I. Sack, and A. Bhate. *Organization Modeling*. Prentice-Hall, Inc., 1999. 2

[Non91] I. Nonaka. The Knowledge-Creating Company. *Harvard Business Review*, pages 96–104, November-December 1991. 4

[PCG98] M. J. Prietula, K. M. Carley, and L. Gasser, editors. *Simulating Organizations*. AAAI Press/The MIT Press, 1998. 76, 78

[SD98] Y. So and E. H. Durfee. Designing Organizations for Computational Agents. In M. J. Prietula, K. M. Carley, and L. Gasser, editors, *Simulating Organizations*, pages 47–64. AAAI Press/The MIT Press, 1998. 78

[Sen90] P. Senge. *The Fifth Discipline - The Art and Practice of the Learning Organization*. Currency Doubleday, 1990. 1, 4, 6, 7, 8, 9, 22, 24, 25, 34, 77, 80

[Sil04] L. P. Silva. *Um Modelo Formal para a Quinta Disciplina*. PhD thesis, Instituto de Matematica e Estatistica, Universidade de Sao Paulo, 2004. (in Portuguese). 39, 81

[SKR+94] P. Senge, A. Kleiner, C. Roberts, R. Ross, and B. Smith. *The Fifth Discipline Fieldbook - Strategies and Tools for Building a Learning Organization*. Currency Doubleday, 1994. 1, 11, 18, 22, 25, 62, 80

[Spi89] J. M. Spivey. *Understanding Z - A Specification Language and its Formal Semantics*. Cambridge University Press, 1989. 12, 87

[Spi92] J. M. Spivey. *The Z Notation: A Reference Manual*. url: http://spivey.oriel.ox.ac.uk/ mike/zrm/, 2nd edition, 1992. 1, 12, 87

[Tay11] F. W. Taylor. *The Principles of Scientific Management*. Harper, New York, 1911. 2

[Tho67] J. D. Thompson. *Organizations in Action: Social Science Bases of Administrative Theory*. McGraw-Hill, New York, 1967. 76

[Tri81] E. L. Trist. The evolution of sociotechnical systems as a conceptual framework and as an action research program. In A. H. Van de Ven and W. F. Joyce, editors, *Perspectives on Organization Design and Behavior*, pages 19–75, New York, 1981. John Wiley, Wiley-Interscience. 78

[Wei01] G. Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 2001. 75

[YS99] L. Yu and B. Schmid. A Conceptual Framework for Agent Oriented and Role Based Workflow Modeling. In *Presented at CaiSE Workshop on Agent Oriented Information Systems (AOIS´99) - Heidelberg, Germany*. http://www.knowledgemedia.org/netacademy/publications.nsf/all_pk/1318, 1999. 78

[ZTC03] ZTC. ZTC - A Type Checker for Z Notation. 2003. url: http://se.cs.depaul.edu/fm/ztc.html. 15

A Z Notation Overview

We present below some introductory concepts that should help the reader to understand the Z notation. We admit, however, that this basic introduction may not provide sufficient information for the reader with no previous experience with Z. In this case, we suggest [Spi92, Spi89, Jac97] for an introduction to this notation. The information present in this appendix come from these references.

Z is a state-based formal language that uses predicate logic, set theory and ordinary discrete mathematics in its notation. Mathematical data types are used to model data in a system. The behavior of the specified system can be understood via mathematical laws which constrain these types. In Z, a collection of state variables can be used to model the state of a system. In addition, operations, which represent changes in states, can be specified. Z specifications usually start with atomic objects, for which the internal structure is not relevant. These objects are the members of the given sets, or basic types, of the specification. Based on these types, more complex types can be built: set, Cartesian product, and schema types. Furthermore, abbreviation definitions may be used to introduce global constants.

Z specifications are composed of formal and textual paragraphs. The first include, for example, declarations of formal types and axioms. The second is used to provide support for and to enhance the readability of the formal paragraphs.

A.1 Sets

The concept of set is fundamental in Z. A set comprises a collection of similar objects. Each set is treated as a unique object in a given specification and its members are named elements.

In Z only sets composed of elements of the same type can be defined. The object's type corresponds to the maximal set that has such object as an element.

We present below the representation in Z for set relations and operations. We assume that the reader has previous knowledge of these relations and operations and omit further details regarding this subject in this work.

$x \in A$	x is element of A
$\varnothing$	empty set
$A \subseteq B$	A is a subset of B
$A \subset B$	A is a proper subset of B
$A \times B$	Cartesian product
$A \cup B$	union of sets A and B
$A \cap B$	intersection of sets A and B
$A \setminus B$	difference set beteween A and B
$\bigcup A$	generalized union of elements taken from A
$\#A$	length of a finite set
$\mathbb{P}\, A$	power set of A

A.2 Definitions

In Z there are several ways to define an object: declaration, abbreviation and axiomatic definitions.

A.2.1 Declarations

The basic way to define an object is via declaration, where the name of the type is written enclosed in square brackets. Types that are declared in such way are named given types. Thus, the declaration below introduces a new given type, named T_1.

$$[T_1]$$

In one declaration it is possible to introduce more than one given type, as follows:

$$[T_1, T_2, ..., T_n]$$

A.2.2 Abbreviations

Using abbreviations it is possible to associate new names to mathematical objects previously defined in the specification. In the example presented below, *id* is a new name, declared as a global constant in the specification, that has the same type and value associated to the expression *Expression*.

$$id == Expression$$

A.2.3 Axiomatic definitions

Objects can also be defined with associated constraints that must be respected whenever the defining symbol is used in the specification. Below, *predicate* specify the constraints that apply to objects introduced in *declaration*.

$$\begin{array}{|l} declaration \\ \hline predicate \\ \hline \end{array}$$

A.2.4 Generic definitions

It is also possible to introduce axiomatic definitions in a generic way. Thus, a family of global constants is defined, parameterized by some type X.

$$\begin{array}{|l} [X] \\ \hline a : X \\ \hline pred \\ \hline \end{array}$$

In this definition, the constant a (of type X) must satisfy the predicates declared in *pred*. Furthermore, X is a set that corresponds to a formal parameter. Its scope is the body of the definition.

A.2.5 Sets and Predicates

All objects defined in Z correspond to sets. Hence, a predicate in terms of the collection of objects that satisfy its constraints.

Thus, given a predicate *pred* with a free variable a of type T, the set C of values of a that satisfy *pred* is defined as follows.

$$C = \{a : T \mid pred\}$$

Suppose that we define the symbol *Honest*.

$$Honest : \mathbb{P}\, T$$

.....

Then, in order to specify that "a is honest" we can use the representation:

$$a \in Honest$$

A.3 Schemes

Schemes are used in Z to structure a formal specification in parts that can be more easily understood and that can later be combined and reused, facilitating the specification process and reading.

A schema corresponds to a piece of formal text that has an associated name, and that describes some variables whose values are constrained in some way.

There are two parts in a schema: declarative where variables are introduced, and predicative that presents the relations and restrictions involving some (or all) of the variables declared in the first part. The predicative part may be empty.

Schemes can be written in two forms: horizontally or vertically.

$$S_1 \mathrel{\widehat{=}} [D_1;\ ...;\ D_n \mid P_1;\ ...;\ P_m]$$

S_2

$D_1;$
$D_2;$
...;
$D_n;$

$P_1;$
...;
$P_m;$

S_3

$D_1;$
...;
$D_n;$

In the schemes presented above S_1, S_2 and S_3 correspond to the names of the schemes; D_1; ...; D_n correspond to declarations of the variables; and P_1; ...; P_m correspond to the predicative part. Schema S_3 does not have a predicative part. Schemes S_2 e S_3 are presented vertically, while schema S_1 is presented horizontally.

Schemes can also be seen as types in Z. A schema enables the introduction of composite types, that include a variety of components. For example, the *Sinteger* schema corresponds to a type composed of two components: a set of integers *sint* and an integer *int*. The introduction of a variable of this type is done via a declaration like: $var_1 : Sinteger$.

Sinteger

$$int : \mathbb{Z}$$
$$sint : \mathbb{P}\,\mathbb{Z}$$

The description of an object of type schema includes a list of the components' names followed by their bindings. For example, a specific binding for the schema presented above can be described as follows.

$Sinteger\langle\!| int == 1, sint == \{1, 3, 5, 7\} |\!\rangle$

The schema type *Sinteger* is the set of all bindings where *sint* and *int* are associated to a set of integers and to an integer, respectively.

The order in which components are declared is not significant. It is also possible to reference a specific component in a schema using the selection operator ".". For example, considering the declaration $var_1 : Sinteger$, to reference the integer component of *Sinteger*, one would write: $var_1.int$.

One of the most frequent operations involving schemes is inclusion. The declarative part of a schema may contain declarations of variables of several types. Moreover, it may also contain references to other previously defined schemes' names. These references are named schema inclusion as they imply in the inclusion of all declarations and predicates of the "referenced" schema in the "referencing" schema.

For instance, in the S_a schema below, we have the inclusion of *Sinteger*.

S_a

$$Sinteiro$$
$$int2 : \mathbb{Z}$$

$$int2 < int$$
$$int2 \in sint$$

Schemes are used in Z specifications to describe states of a system and also operations on these states. There are some conventions to describe that the state defined by a particular schema is influenced by some action and that this state can be changed or stay unaltered.

Initially, there are conventions regarding the decoration in variable declarations using special characters. Hence, it is possible to identify input, output

and state variables via concatenation of the variable names to characters "?", "!" e " ′ ", respectively.

Now we can define a state description schema as a schema in which there are no decorated variables. In addition, appending the special character " ′ " to the name of a schema S indicates a new schema that includes the same variables and predicates contained in S, with character " ′ " appended to all variables in the declarative part and to every occurrence of free variables in the predicative part.

On the other hand, in operation schemes input and output variables are used. Furthermore, new values can be assigned to the variables declared in the schema.

In order to simplify the definition of an operation schema over a state schema the Δ variant is used. Assuming that S is a state schema, ΔS defines a new schema defined as follows.

ΔS

S

S'

Each variable name decorated with " ′ " indicates a state variable that contains the new value that will be assigned to the corresponding undecorated variable when the operation is completed.

For instance, an operation on the *Sinteger* schema can be defined as follows.

ChangeSinteger

$\Delta Sinteger$

$int? : \mathbb{Z}$

$int' = int?$

$sint' = sint$

There is also a simplified way to indicate that in a given operation the state variables stay unchanged, using the Ξ variant. Assuming that S is a state schema, ΞS defines a new schema defined as follows.

ΞS

ΔS

$v_1' = v_1$

$v_2' = v_2$

.

.

.

$v_n' = v_n$

In this example, $v_1', ..., v_n'$ correspond to the state variables present in S'.

In another example, it is possible to define an operation on the S_a schema defined above, that only changes the state of the variable declared in it and keeps unchanged the state described by *Sinteger*.

ChangeS$_a$

ΔS_a
$\Xi Sinteger$
$intNew? : \mathbb{Z}$

$int2' = int2 + intNew?$

A.4 Lambda and theta notation

There is a particular way to define functions in Z, named lambda expression. This expression presents a structure that includes *declaration*, *predicate* and *expression*.

$$(\lambda\, declaration \mid predicate \bullet expression)$$

Actually, considering functions as sets of pairs, the first element of each pari is defined by *declaration* and *predicate* and the second is described by *expression*. In the example below, the function *double* relates to each integer its double.

$$double == (\lambda\, integer : \mathbb{Z} \bullet integer * 2)$$

Lambda expressions can be used in any part of a formal specification. The expression above defined was used to introduce a function that can be used several times in the context of a specification, as presented below.

$$double\ 8$$

We have already seen that a schema reference is, in fact, a reference to a set of bindings. However, there may be situations where we want to have access to the bindings associated to a particular schema. In Z the θ operator has this function. Hence, assuming that *NE* is a schema, the expression θNE refers to the specific binding of *NE* that has a valid scope in the part of the specification where this expression is declared. Moreover, this operator enables the definition of a relation associated to an operation schema. Assuming that *OperationOnSchema*1 is an operation defined over *Schema*1 schema, the declaration presented below defines a relation between before and after *Schema*1 states.

OperationOnSchema1

$\Delta Schema1$

...

$$relationBetweenStates == \{OperationOnSchema1 \bullet (\theta Schema1, \theta Schema1')\}$$

Another interesting use of θ and λ notations involves schema inclusion operations. Considering the schemas presented below:

Sinteger

$int : \mathbb{Z}$
$sint : \mathbb{P}\,\mathbb{Z}$

S_a

Sinteger
$int2 : \mathbb{Z}$

$int2 < int$
$int2 \in sint$

In some situations it is necessary to refer to the *Sinteger* schema. However, we only have a reference to S_a in scope. This can be done using the following function:

$$(\lambda\, S_a \bullet \theta Sinteger)$$

A.5 Notation summary

$p \wedge q$	log. conjunction	$A \rightarrow B$	total function
$p \Rightarrow q$	log. implication	$\mathrm{dom}\, R$	domain of relation
$\forall X.q$	univ. quantification	$\mathrm{ran}\, R$	range of relation
$x \in A$	set membership	$\mathrm{seq}\, A$	finite sequences
$\varnothing$	empty set	$\preccurlyeq$	sub sequence of a sequence
$A \subseteq B$	set inclusion	$head\ s$	first element of a sequence
$A \times B$	Cartesian product	$last\ s$	last element of a sequence
$A \cup B$	set union	$\langle i, j, .. \rangle$	sequence
$A \cap B$	set intersection	$\mathbb{P}\, A$	power set
$\bigcup A$	generalized union	$\mathbb{P}_1\, A$	non empty power set
$\#A$	size of finite set		
$A \nrightarrow B$	partial function		

$\mathrm{T} ::= \mathrm{c}_1 \mid ... \mid c_m \mid d_1 \langle\!\langle E_1[T] \rangle\!\rangle \mid ... \mid d_n \langle\!\langle E_n[T] \rangle\!\rangle$ free type

Conventions and definitions:

a,b	identifiers	S'	schema after operation
p, q	predicates	ΔS	change of state
A, B	sets	ΞS	no change of state
S	schema before operation		

B Schema Structure Diagram

The diagram below represents the basic schema structure of our formal model for the Fifth Discipline.

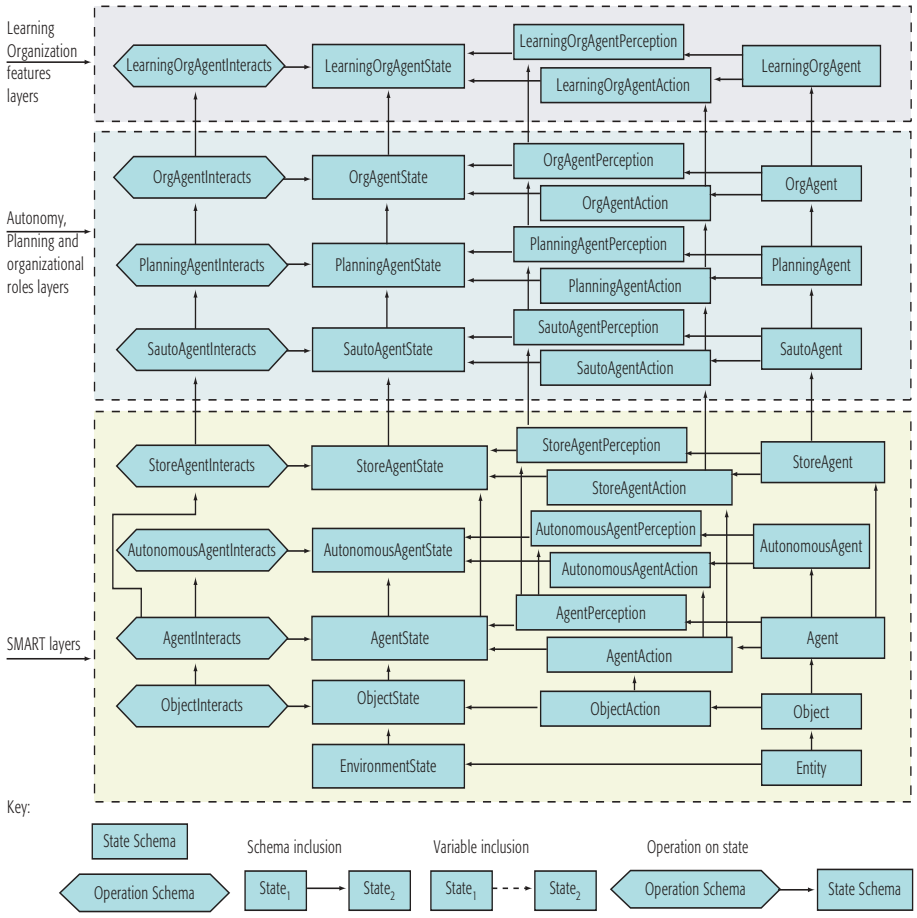

Figure 7: Formal model's basic schema structure.

Index