LEARNING SITUATION MODELS FOR UNDERSTANDING ACTIVITY

James L. Crowley, Oliver Brdiczka, Patrick Reignier
Project PRIMA, INRIA Rhône Alpes
655 Ave de l'Europe
Montbonnot, France

{James.Crowley, Oliver.Brdiczka, Patrick.Reignier}@inrialpes.fr

Abstract - **Human social activities follow loosely defined scripts in which individuals assume roles. Social conventions establish stereotypical skeletons for such scripts, with details and variations determined by the circumstances and personalities of the individuals who play the roles. Encoding social scripts in a formal representation would seem to hold the promise of building systems that provide services based on observation and understanding of human activity. Unfortunately, the cost and complexity of handcrafting models for the variety of ways in which scripts can be enacted makes such an approach infeasible. Clearly learning and development based on individual behaviour are necessary.**

In this paper, we present a method for building systems that develop social scripts through incremental supervised learning. We begin with a conceptual framework in which scripts for human activity are described as scenarios composed of actors and objects within a network of situations. We then present methods to develop situation networks by incrementally building on an existing stereotypical script. We present supervised learning techniques for learning to discriminate new situations and for learning to recognize new roles. We illustrate our approach with results from experiments with situation and role learning in an office environment.

Index Terms:* *Context Aware Systems and Services, Activity Observation, Situation, Development and Adaptation

I. INTRODUCTION

Continued exponential decline in the cost of both communications and information technology would seem to enable a large and diverse array of services for enhancing human-to-human interaction. Examples of such services would include automated camera control for video-conferencing, communication tools for collaborative work, as well as tools for organizing and conducting meetings. Unfortunately, despite the presence of enabling communications technology, the use of information technology to enhance human-to-human interaction is currently impractical because of a lack of a technology for observing and understanding social activity. Because the details of such activity are highly dependent on individual personalities and cultural background, they depend on models that must be learned and developed through interaction with users.

We propose a conceptual framework and a software model for systems that learn to observe and model human social activity. The core component of our framework is a situation model. The situation model acts as a non-linear script for interpreting the current actions of humans, and predicting the corresponding appropriate and inappropriate actions for services. This framework organizes the observation of interaction using a hierarchy of concepts: scenario, situation, role, action and entity.

The following section outlines the conceptual framework for this theory. This is followed by a proposal for a layered architecture for human interaction services. Within this layer we present a component based architectural model that uses concepts from autonomic computing to provide observation of human activity that robustly adapts to changes in the environment. We then propose techniques for learning such models through incremental development.

II. SCRIPTS FOR HUMAN ACTIVITY

Most human societies have developed and refined an artistic framework for formally encoding social interaction: the Theater. A theatrical production provides a model for social interaction in the form of a script. Theater can be used as a rich source of concepts for socially aware observation of human activity.

A theatrical production organizes the actions of a set of actors in terms of roles structured as a sequence of scenes composed, in turn, of a series of situations. A role is more than a set of lines. A role defines a space of allowed actions, including dialog, movement and emotional expressions. The audience understands the production by recognizing the roles using social stereotypes, and relating these to individual social experiences.

In a similar manner, everyday human actions and interactions can be observed and described in terms situations in which individuals play roles. Depending on the activity, actions and interactions may be more or less constrained and limited by implicit compliance with a shared script. Deviating from the script is considered impolite and can often provoke conflict or even terminate the interaction. Some activities, such as classroom teaching, formal meetings, shopping, or dining at a restaurant, follow highly structured scripts that constrain individual actions to highly predictable sequences. Other human activities occur in the absence of well-defined scripts, and are thus less predictable. We propose that when a stereotypical social script does exist, it can be used as a starting point to learn models for understanding activity to guide the behavior of services.

One important difference exists between theater and life. A theatrical script is composed of a fixed sequence of situations. Real life is much less constrained. For many activities, situations form a network rather than a sequence, and may often exhibit loops and non-deterministic branching. The complexity and difficulty of observing human activity is related to the degree of interconnectivity of situations.

III CONCEPTUAL FRAMEWORK

Translating theatrical concepts into software requires formal expression. To be meaningful, this formal expression must ultimately be grounded in procedures and actions for real systems. In earlier papers, we have proposed a conceptual framework for observing activity [1], [2] as well as a component based software models [3], [4] for building such systems. In this section we review the definitions for concepts for observing human activity. Since what we are describing is sometimes called a context model, we start with the definition of a "Context model".

Context: The situation within which something exists or happens, and that can help explain it [5]; Any information that can be used to characterize situation. [6].

We compose models for observing activity in terms scenarios composed of "situations". In common use, situation derives its meaning from the way in which something is placed in relation to its surroundings. For examples, many authors define situation in terms of position and action. The Cambridge on line dictionary defines situation as

Situation: the set of things that are happening and the conditions that exist at a particular time and place. [5].

In our case, the definition of situation requires defining two facets: Observation and Reaction. Observation refers to the concepts with which the system can observe situation. Reaction refers to the set of actions that the system should take based on the current situation.

A situation is defined in terms of the configuration of a set of entities playing roles. Configuration is expressed as a set of predicate functions whose arguments are the entities playing the roles.

Situation: A predicate expression of a set of relations over entities assigned to roles.

A situation is a form of state, expressed as a logical expression (a conjunction of predicates). This logical expression is composed of predicates whose arguments are roles. This concept generalizes and extends the common practice of defining situations based on the relative position of actors and objects.

Relations are predicate (truth) functions with one or more arguments. Relations are either true or false, depending on the properties of their arguments. Unary relations apply a test to some property or set of properties of an individual entity. Binary and higher order relations test relative values of properties of more than one entity. Examples include spatial and temporal relations (in front of, beside, higher than, etc), or other perceived properties (lighter, greener, bigger, etc.).

Relation: A predicate test on properties of one or more the entities playing roles.

Relations test the properties of entities that have been assigned to roles. Operationally, a role is an abstract generalization for a class of entities. Role classes are typically defined based on the set of actions that entities in the class can take (actors), or the set of actions that the entities can enable (props). Formally, role is a function that selects an entity from the set of observed entities.

Role: A function that selects an entity from the set of observed entities.

The definition of role can be completed by definitions for actors and props.

Actor A role for entities that can spontaneously act to change the current situation.

Prop A role for entities that cannot spontaneously act to change the current situation.

A "role" is NOT an intrinsic property of an entity, but rather, is an interpretation assigned to an entity by the system. The role function acts to select an entity from the available set of entities, and not the other way around.

To summarize so far, we have situations defined with logical expressions composed of relations over entities playing roles, and producing a set of actions to be taken by the system. As mentioned above, situations also predict possible future situations. This is captured by the connectivity of a situation network. Changes in the logical expression of relations or in the selection of entities playing roles are represented as changes in situation. Such changes can trigger system actions.

So how does the role assignment process select among the available entities? We propose to view this process as a "filter" [1]. In this view, a filter acts as a kind of sorting function for the suitability of entities based on their properties. The most suitable entity wins the role assignment.

The lowest level concepts in this framework are entity and property. A property refers to any value that can be observed, or inferred from observations. An entity is a correlated collection of properties. This solipsistic viewpoint admits that the system can only see what it knows how to see. At the same time, it sidesteps existential dilemmas related to how to define notions of "object" and "class". In this view, a chair is anything that can be used as a chair, regardless of its apparent form. More formal definitions for these two concepts are rooted in the software architectural model described below. Operational definitions for property and entity are grounded in the software components for observation of activity.

IV LAYERED SOFTWARE ARCHITECTURE

We propose a layered architectural model for services based on human activity. Four layers of this model are shown in figure 1. At the lowest layer, the service's view of the world is provided by a collection of physical sensors and actuators. This corresponds to the *sensor-actuator layer*. This layer depends on the technology and encapsulates the diversity of sensors and actuators by which the system interacts with the world. Information at this layer is expressed in terms of sensor signals and device commands.

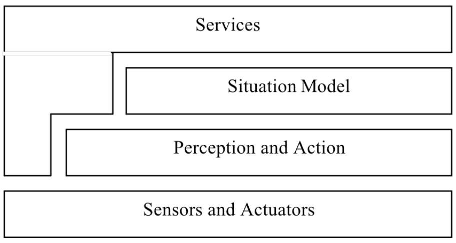

Fig. 1. A layered model for non-disruptive services

Hard-wiring the interconnection between sensor signals and actuators is possible, and can provide simplistic services that are hardware dependent and have limited utility. Separating services from their underlying hardware requires that the sensor-actuator layer provide logical interfaces, or standard API's, that are function centered and device independent. Hardware independence and generality require abstractions for perception and action.

Perception and action operate at a higher level of abstraction than sensors and actuators. While sensors and actuators operate on device specific signals, perception and action operate in terms of environmental state. Perception interprets sensor signals by recognizing and observing entities. Actions are tasks expressed in terms of a desired result rather than commands to be blindly executed.

For most human activities, there are a potentially infinite number of entities that could be observed and an infinite number of possible relations for any set of entities. The appropriate entities and relations must be determined with respect to the service to be provided. This is the role of the situation model, as described in the previous section. The situation model allows the system to focus perceptual attention and computing resources, in order to associate the current state of the activity, with the appropriate system action.

Services specify a scenario composed of a situation model, as described above. The scenario determines the appropriate entities, roles and relations to observe, acting in a top-down manner to launch (or recruit) and to configure a set of components in the perception-action layer. Once configured, the situation model acts as a bottom-up filter for events and data from perceptual components to the service.

V THE PERCEPTION-ACTION LAYER

At the perception-action layer, we propose a data-flow process architecture for software components for perception and action [7], [8], [4]. Component based architectures, as described in Shaw and Garlan [9], are composed of auto-descriptive functional components joined by connectors. Such an architecture is well adapted to interoperability of components, and thus provides a framework by which a system can adapt to an environment by exchanging or reconfiguring components.

Within the perception-action layer, we propose three distinct sub-layers, as shown in figure 2. These three sub-layers are the Module layer, the Components layer and Federation layer. The elements within each sub-layer are defined in terms of the elements in the sub-layer below. Each sub-layer provides the appropriate set of communications protocol and configuration primitives for the sub-layer above.

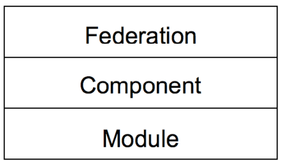

Fig. 2. Three sublayers within the perception action layer.

VI. ADAPTATION AND DEVELOPMENT

We distinguish the concepts of adaptation from development [2]. *Adaptation* allows a system to maintain consistent behaviour across variations in operating environments. The environment denotes the physical world (e.g., in the street, lighting conditions), the user (identification, location, goals and activities), social settings, and computational, communicational and interactive resources. *Development* refers to the acquisition of abilities, in this case encoded as situation models composed of the entities, roles and relations with which situation is described and service actions are performed.

Systems for providing services based on observing activity must both adapt and develop. Adaptation is necessary to maintain consistent behaviour while accommodating changes in the operating environment, task, user population, preferences or some other factors. At the same time, human activity is too complex to be fully captured in a pre-programmed situation model. An activity model must develop through observation and interaction with users. A fundamental challenge is to provide both automatic adaptation and automatic development without disruption.

Current learning technologies, such as EM, AdaBoost and neural networks, require large sets of training data – something that is difficult to obtain for an extensible environment. Non-disruptive development of context models requires new ways of looking at learning, and may ultimately require a new class of minimally supervised learning algorithms. This requires that learning be studied as part of a semi-autonomous system. It requires that systems have properties of self-description, self-evaluation and auto-regulation, and may well lead to new classes of learning algorithms specifically suitable to developing and evolving context models in a non-disruptive manner.

We are currently experimenting with techniques for adapting activity models based on pre-defined stereotypical situations. We are exploring different approaches to learning for development of activity models starting from a predefined stereotypical model using feedback about the system actions. Because the different components of the model (entities, roles, relations, and situations) depend on each other, these cannot be developed simultaneously. Thus we have focused on the development of the situation networks and the associated system actions.

Bayesian models (in particular Hidden Markov Models [10] as well as algorithms based on first-order logic [11] can be used to represent and adapt the situation network. However, these approaches do not have desirable properties concerning the extension of the number of situations. Bayesian models require a large amount of example data to extend the number of states. First-order logic algorithms cannot create new predicates (problem of higher order logic), which is necessary for the extension of situations.

Thus we propose an approach for changes in the structure of the situation network, as shown in figure 3.

The input to the algorithm is a predefined situation network along with feedback from prior use mediated by a supervisor. The supervisor corrects, deletes or preserves the actions executed by the system while observing a user in the environment. Each correction, deletion, or preservation generates a training example for the learning algorithm containing current situation, roles and configuration of relations, and the (correct) (re)action. The differences between the actions given in the training examples and the actions provided in the predefined situation network will drive the different steps of the algorithm.

Initially, our approach is to try to directly modify system actions using the existing situation network. If action A is associated with situation S, and all training examples indicate that action B must be executed instead of A, then B is associated to S and the association between A and S is deleted.

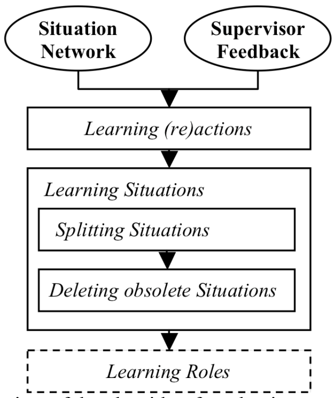

Fig 3: Overview of the algorithm for adapting system actions

Learning to Discriminate Situations

Whenever training or feedback indicates different actions for the same situation, the situation may be deemed overly general and may be split, with a new situation created for each action. The problem is to determine the configuration of roles that can be used to discriminate these new situations.

Because a context is defined by a finite number of available roles and relations, the situations within this context can be represented as a fixed-sized vector containing a binary value for each available role and for each available relation. The value 1 means that the corresponding role or relation is valid; the value 0 means that the role or relation is not valid. As a relation is applied to entities playing roles, it is represented by a binary value for each different role combination it can be applied to. The configuration of roles for a situation may contain blanks ("-") for those roles or relations that are not characteristic. A training example contains a vector with specific values reflecting the current role, relation configuration when recording the training example and the corresponding (re)action (given by the supervisor). As the context is defined by the available roles and relations, the description of the situations within this context

The determination of the characteristic configuration of roles can be seen as a classification problem. The (re)action labels of the training examples can be interpreted as class labels. For each class, we need to determine the concepts or hypotheses based on the given role, relation vectors of the class. These concepts or hypotheses are then used to construct the characteristic role, relation configurations of the corresponding sub-situation.

Our first approach to splitting situations uses the Find-S algorithm [12]. This algorithm seeks to construct the most specific hypothesis for each action based on the role and relation configuration for a given training example. The resulting hypotheses for the sub-situations often contain specific values for the existence or non-existence of roles or relations that are not necessary or characteristic. As a consequence, small variations in the configuration of roles may not be covered by the created sub-situations because their hypotheses are too specific.

To produce more general hypotheses for the sub-situations, we explored a second approach based on the algorithm for conceptual learning algorithm named Candidate Elimination [12]. This algorithm constructs the most specific and the most general hypotheses for each action based on the role, relation configurations in the given training examples. By combining the most general hypotheses for each action, we construct the role, relation configuration for the corresponding sub-situations.

Unfortunately, both the algorithms Find-S and Candidate Elimination have the restriction that they can only find one conjunctive concept for each (re)action, i.e. if the training examples indicate that a (re)action is to be executed in two different complementary role-relation configurations, Find-S and Candidate Elimination will fail to construct several hypotheses (and thus sub-situations) for this one (re)action. This is due to the fact that neither algorithm can construct disjunctive hypotheses.

We have thus investigated a third approach based on learning a Decision Tree using the algorithm ID3 [13]. The idea is to construct a decision tree that classifies the different actions found in the training examples of one situation. The attributes of this decision tree are the binary roles and relation values of the vector. Each leaf of the tree is labeled with an action. The path from the root of the tree to the leaf gives the characteristic role, relation configuration for the sub-situation to be created for this action. We can have several leaves with the same action, which corresponds to the creation of several sub-situations for this action (disjunctive hypotheses).

Learning to Recognize new Roles

If the information supplied by training examples is not sufficient to discriminate characteristic configurations for the sub-situations during the situation split, the creation and learning of new roles need to be considered. This is the case when the supervisor gives different feedback while the system perceives the same situation, role and relations configurations.

Roles, Relations	**Feedback**	**Observed Entity Properties**	**Associated Role Configuration**
(1,0,0,1)	A1	(Entity1, 101, 18) (Entity1, 105, 20) (Entity1, 108, 22)	NewRole1 = 0
(1,0,0,1)	A2	(Entity1, 25, 0) (Entity1, 21, 2) (Entity1, 18, 5)	NewRole1 = 1

Table 1: An example for learning a role. The role acceptance test is based on a calculation of the probability of the role value, given the entity position.

When creating a new role, we need to learn the corresponding acceptance test to be applied to the properties of the available entities. Learning a role acceptance test can be seen as a classification problem. The different supervisor feedback events must be distinguished based on the properties of observed entities. Table 1 gives an example. The entities and their properties are created by a tracking system running with video images from a wide-angle camera. The properties of an entity are its name and its current position in the image. Learning a role acceptance test corresponds to learning a new characteristic entity position. Given a sufficient amount of sensor-based position data, a Bayesian learning approach for learning the role acceptance may be used in this case

A problem is to decide which entity or entities to chose for learning the role acceptance test. In the example we refer to only one available entity. If there are several entities available, the entity that best allows situation discrimination must be identified. With a Bayesian approach, a maximum likelihood estimate can be used for determining this entity.

While the development of the situation network, i.e. adapting actions and situations, can be seen as generic approach that is independent of specific perceptual components, learning new roles relies on the properties generated by these components. Thus the choice of the algorithms for learning the acceptance test as well as for determining the relevant entities depends on the available perceptions representing the entity properties. Algorithms for learning role acceptance tests as discriminative recognition are currently being compared.

Experimental Demonstration

The following demonstrates development of an activity model within our SmartOffice environment [14] using our robust tracking system [15] with a wide-angle camera. The position of entities determines several roles such as comes_in or works_on_PC. Additional roles are determined by the login of an entity (person) to a computer in the environment or specific appointments marked in the agenda of the logged entity (person). The not_same_entity_as relation is used to distinguish entities in the environment. The actions of the system are based on the control of the Linux music player and the projection of different messages or presentations on different surfaces in the environment. The learning algorithms are run on data-base tables containing a representation of the current situation network and the training examples. A control process programmed a JAVA implementation (JESS) of the forward chaining rule programming environment CLIPS is used to execute the situation network. The situation network is represented by rules that have been automatically generated from the database tables produced by the learning algorithm. The supervisor feedback cannot be the system is running. Thus the control process and the learning algorithms need, at present, to run off-line.

To evaluate our method, two experiments have been executed on a predefined activity model in the SmartOffice environment. Situations in figure 4 include **S0** (empty room), **S1** (newcomer enters SmartOffice), **S2** (Person connects to and works on PC), **S5** (Connected Person sits on couch) and **S8** (Presentation in SmartOffice). The experiments are to develop the system services. The supervisor gives feedback based on goals during the experiments. As we focus on the correct execution of the system services, we do a cross-validation by adapting the predefined situation network using the supervisor feedback of the first experiment and by evaluating the second experiment on the adapted situation network (and inverse). The evaluation is done on the number of correctly classified training examples, i.e. correctly executed actions, as well as on the review of the adaptations of the predefined situation network.

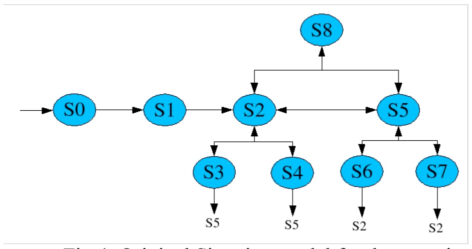

Fig 4: Original Situation model for the experiment

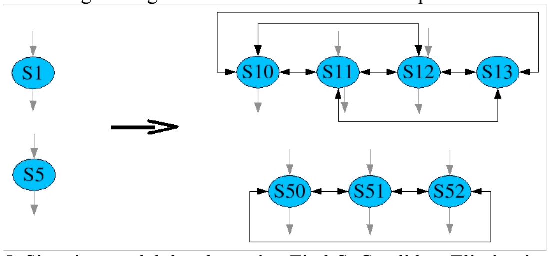

Fig 5. Situation model develop using Find-S, Candidate Elimination and Decision Tree algorithm). Situations **S1** and **S5** have been split.

The goal of both experiments was to integrate the correct behaviour for starting (turn-on) and stopping (turn-off) of a Linux music player depending on the activities of the user. The music player should be switched on when a newcomer sits on the couch, and switched off when the newcomer starts speaking or leaves the couch (situation: **S1**). The music player should similarly be switched on and off for a connected person (situation: **S5**). Figure 5 shows the adaptations of the situations after the integration of the supervisor feedback. **S1** has been split into additional sub-situations integrating sitting down on couch (**S11**), speaking on couch (**S12**) and leaving couch (**S10**). The additional sub-situations of **S5** integrate sitting down on couch (**S51**) and speaking on couch (**S52**).

Table 2, 3 and 4 show the results of the (re)action execution in the form of confusion matrices. (A8 switches on the music player, A9 switches off the music player, and A0 is the "do nothing" (re)action).

In all experiments, the structural development of the situation network corresponds to the expected changes. Concerning the correct classification of the training examples, i.e. the correct execution of the actions, the Decision Tree algorithm (ID3) gives the best results.

The improvements with Decision Tree approach are due to the fact that this algorithm supports disjunctive hypotheses. However, the Decision Tree algorithm tends to construct hypotheses that are overly general for the sub-situations, which can lead to inappropriate classifications.

This is due to the fact that the Decision Tree algorithm prefers small trees to large trees, which means that general hypotheses are preferred to specific hypotheses for the sub-situations.

Find-S	**A0**	**A8**	**A9**
A0	0.87	0.04	0.09
A8	0.50	0.50	0.00
A9	0.50	0.00	0.50

Table 2: Confusion matrix for Find-S.

C. El.	**A0**	**A8**	**A9**
A0	0.91	0.04	0.04
A8	0.66	0.33	0.00
A9	0.75	0.00	0.25

Table 3: Confusion matrix for Candidate Elimination.

D Tr.	**A0**	**A8**	**A9**
A0	0.83	0.09	0.09
A8	0.00	1.00	0.00
A9	0.00	0.00	1.00

Table 4: Confusion matrix for Decision Tree from ID3.

IX. CONCLUSIONS

Activity models for context aware services can be expressed as a network of situations concerning a set of roles and relations. Roles are abstract classes for actors or props. Entities may be interpreted as playing a role, based on their current properties. Relations between entities playing roles define situations. This conceptual framework provides the basis for adaptation and development of non-disruptive software services for aiding human-to-human interaction.

Socially aware observation of activity and interaction is a key requirement for development of non-disruptive services. For this to become reality, we need methods for robust observation of activity, as well as methods to automatically learn about activity without imposing disruptions. The framework and techniques described in this paper are intended as a foundation for such observation.

ACKNOWLEDGMENT

This work has been supported by the EC IST FAME project (IST-2000-28323) and IP CHIL as well as French national project RNTL/ProAct ContAct. This work has been performed in active collaboration with Joelle Coutaz, Gaetan Rey, Dave Snowdon, Jean-Luc Meunier and Alban Caporossi and other members of the PRIMA group.

BIBLIOGRAPHY

[1] O. Brdiczka, J. Maisonnasse, P. Reignier, Automatic Detection of Interaction Groups, 2005 International Conference on Multimodal interaction, ICMI '05, Trento It., october 2005

[2] J Coutaz, J. L. Crowley, S. Dobson, and D. Garlan, "Context is Key", Communications of the ACM, Special issue on the Disappearing Computer, Vol 48, No 3, pp 49-53 March 2005.

[3] J. L. Crowley, J. Coutaz, G. Rey and P. Reignier, "Perceptual Components for Context Aware Computing", UBICOMP 2002, International Conference on Ubiquitous Computing, Goteborg, Sweden, September 2002.

[4] J. L. Crowley, "Integration and Control of Reactive Visual Processes", Robotics and Autonomous Systems, Vol 15, No. 1, decembre 1995.

[5] Cambridge On-line dictionary of the English Language, http://dictionary.cambridge.org

[6] Dey, A. K. "Understanding and using context", Personal and Ubiquitous Computing, Vol 5, No. 1, pp 4-7, 2001.

[7] Software Process Modeling and Technology, edited by A. Finkelstein, J. Kramer and B. Nuseibeh, Research Studies Press, John Wiley and Sons Inc, 1994.

[8] J. Rasure and S. Kubica, "The Khoros application development environment ", in Experimental Environments for computer vision and image processing, H. Christensen and J. L. Crowley, Eds, World Scientific Press, pp 1-32, 1994.

[9] M. Shaw and D. Garlan, Software Architecture: Perspectives on an Emerging Disciplines, Prentice Hall, 1996.

[10] L. R. Rabiner, A Tutorial on Hidden Markov Models and selected Applications in Speech Recognition. Readings in speech recognition. p. 267-296, 1990.

[11] J. R. Quinlan, Learning Logical Definitions from Relations. Machine Learning. 5(3), p. 239-266, 1990.

[12] T.M. Mitchell, Machine Learning. McGraw Hill, New York, USA, international edition, 1997.

[13] J.R. Quinlan, Induction of Decision Trees. Machine Learning. 1(1), p. 81-106, 1986.

[14] C. Le Gal, J. Martin, A. Lux, and J.L. Crowley . SmartOffice: Design of an Intelligent Environment. IEEE Intelligent Systems. 16(4), p. 60-66, 2001.

[15] A. Caporossi, D. Hall, P. Reigneir, and J.L. Crowley, Robust Visual Tracking from Dynamic Control of Processing. PETS '04, Sixth International Workshop on Performance Evaluation of Tracking and Surveillance. Prague, May 2004