Proceedings of the IEEE ICRA 2009
Workshop on People Detection and Tracking
Kobe, Japan, May 2009

Situation Models: A Tool for Observing and Understanding Activity

James L. Crowley, *Member, IEEE*, Patrick Reignier and Remi Barranquand

Abstract— **In this paper we describe the use of situation models for observing and understanding activity. Observing activity in natural environments can be an extremely complex perceptual problem. Situation models provide a means to both focus attention in such systems and to provide default reasoning to accommodate missing and erroneous observations. We briefly review the use of situations models in Cognitive Science and then describe how such models can be used to provide services based on observation of human activity. We present a layered component-oriented software architecture in which components for perception and action maintain a situation model for use in providing human services. We describe how this model can be used to observe activity.**

I. Introduction

Human activity is extremely rich. Real world scenes can contain an overwhelming number of possible agents and objects to detect and observe. As are result, systems and services based on observation of activity must, either implicitly or explicitly, be able to choose where to look next and what to look for. Designers of system for observing activity are increasingly confronted with the problem of control of attention.

Attention is not the only problem confronting designers of systems for observing activity. Activity in the real world often occurs in less than ideal viewing conditions. Poor lighting, background clutter, object texture, and occlusions can degrade the reliability of even the most well designed systems. Thus systems and services must be able to detect and discard uncertain and unreliable observations, and if appropriate, substitute default information. In addition, many services require real time information from perception. In such systems it may be preferable to provide an immediate response with default information and to use background processes to verify that the response was correct.

Current systems for observing activities tend to be constructed in an ad-hoc manner with control structures that are hard-wired into the system design. Such systems are generally restricted to detecting a very small set of activities observed within a highly controlled environment. Adapting such systems to different operating environments or modifying such systems to observe different forms of activity can involve extensive reprogramming.

In this paper we propose an approach for constructing systems for observing activity based on a model from Cognitive Science. We propose the use of situation models to organize, control, and interpret perception of activity. We will first provide some background from Cognitive Science concerning the use of situation models as a model of human cognition. We then describe how to use such a model to build software systems that provide services. We propose a layered, component-oriented software architecture for building situation aware services, and examine how situation models can be used to structure perceptual components and to provide default information for understanding activity. We conclude with a discussion of the problems of automatically acquiring situation models through developmental learning.

II. Situation Models as Models for Cognition

Situation models have been proposed by Johnson-Laird [1], as a cognitive theory for human mental models. Over the last 25 years, theories about situation models have been adopted and developed by a large community of cognitive psychologists. Key publications include [2], [3] as well as [4].

Situations are defined as a set of relations between entities, where a relation is a predicate function and an entity is anything that can be observed. According to Radansky [2], a situation model is a mental representation of a described or experienced situation in a real or imaginary world. Situation models are commonly composed of four primary types of information:

1) A spatial-temporal framework (spatial locations, time frames)
2) Entities (people, objects, ideas, etc.)
3) Properties of entities (color, emotions, goals, shape, etc.)
4) Relational information (spatial, temporal, causal, ownership, kinship, social, etc.)

Situation models can be structured along dimensions of space, time, causality, actors and objects. Extensions of situations models have been proposed to represent intentions of actors. It is commonly assumed that both general world knowledge (knowledge about concept types, e.g., scripts, schemas, categories, etc) and referent specific knowledge (knowledge about specific entities, independent of the situation) are used in constructing situation models.

Manuscript received March 9, 2009. This work was supported in part by project ANR CASPER, as well as the European IST projects FAME (IST 2000-28323), CAVIAR (IST 2001- 37540) and CHIL (IST 506909)
James L. Crowley is Professor at Grenoble National Polytechnique Institute (INPG) and directs the PRIMA Research group at INRIA Grenoble Centre de Recherche, 655 Ave de l'Europe, 38334 St. Ismier, France.
Patrick Reignier is a Junior Professor (MdC) at the University Joseph Fourier in Grenoble, and member of the PRIMA Research group at INRIA Grenoble Centre de Recherche, 655 Ave de l'Europe, 38334 St. Ismier, France.
Remi Barraquand is a doctoral student at Grenoble National Polytechnique Institute (INPG) under the direction of James L. Crowley and member of the PRIMA Research group at INRIA Grenoble Centre de Recherche.

Situation models are used for representations of:

1) Information about events.
2) Information about sequences of events.
3) Information about collections of episodes

We have adapted the concept of situation model to construct systems and services based on monitoring and observing human activity [5], [6], [7]. Although most of our implementations have been constructed using smart environments, such services can also be designed using robotic systems. Indeed, our approach to smart environments is to see the environment as a form of "inside out" robot, observing and interacting with occupants. Thus we maintain that models for understanding activity in smart environments may also be adapted for construction of autonomous robots.

III. Situation Models for Observing Activity

Situation models can be used to addresses the twin problems of focus of attention, and operation with unreliable, erroneous or missing data. They can also be used to decouple services from the time constraints normally imposed by real-time (or near real-time) vision systems. We present our technique in the context of a service-oriented architecture constructed using a layered, component-based, software model. For the robotics and vision communities, these concepts may require some explanation.

The term "service" is used here in its most general form. Generally, it will refer to assistance that informatics systems provide to people. User services can be designed as software agents that interact and assist people. Over the last few years, we have constructed a variety of services that observe and model human activity in order to provide assistance that is dependent on human context. Such systems are generally said to be "context aware". Exampels include services for lecture recording [8], meeting services [9], monitoring of the health and well-being of elderly, and availability monitoring [7]. As sensor and actuator technology mature, we can expect to see the emergence of an increasing variety of such systems for domestic services (cleaning, logistics, cooking), commercial services (shopping, queue management, customer assistance), health monitoring and assisted living, security monitoring, and a variety of other application domains. All of these examples require observing and understanding the actions of humans. We believe that situation models will provide an important component for such systems.

We note that the term "service oriented" also has a more technical meaning for the software engineering community. In software engineering, a "service oriented" system is one in which software components interact according to a well-defined contract. For example, a location service integrates information from a variety of sources to estimate the current location of a user. Although the two uses of the term "service" are not incompatible, they can cause some confusion. Thus we will use the explicit term "software services" for services that are primarily designed to interact with software components. We will interchangeably use "user services" or simply "services" for systems that interact with and assist people.

Modern software systems are generally designed using a layered architecture. A layered architecture organizes the system into a hierarchy of interchangeable components, with well-defined interfaces. The design and operation at a particular layer may proceed independently of the underlying components. Components that make up a particular layer may be reused or shared by a variety of services. Components that are temporarily inoperative may be replaced with alternative components. A common example of this approach is provided by the current generation of location aware services on mobile devices that can interchangeably use location information from GPS, cell phone repeaters, or WIFI repeater identity. Components for providing location from WIFI, GPS or cell-phone repeaters are a form of "perceptual component" that operate in parallel using competing methods to make available a key piece of information: current location. We propose a similar approach to building components for observing activity. Perceptual components can be constructed to observe a scene with competing methods to provide information that may then be shared between different services.

A situation model falls naturally at the interface between user services and perceptual components. For user services, the situation model provides a default reasoning system that can complete or repair partial or missing information from sensing. For the perceptual components, the situation model can be used to focus attention on the objects and events that are relevant to a service, allowing irrelevant objects or events to be ignored. The situation model can be used to predict possible events, both to focus attention, and to prepare a reaction before the event occurs.

In the following, we describe a layered architecture for context aware user services based on observation of activity. We then describe the elements of the situation model, and describe how such a model can be used to configure and control perceptual components, to focus attention, predict events, and to provide default reasoning for observation of activity.

A. *Services, Sensors, and Components*

We are interested in services that provide assistance through the observation of human activity. A service determines requirements for perception and action, without specifying how these requirements are to be met. 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 makes it possible to build systems that operate in a larger range of environments, for a larger variety of functions. However such separation 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.

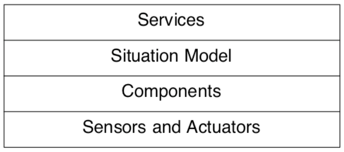

Fig. 1. A layered model for systems that observe human activity.

A layered architecture of user services is 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.

Service abilities for perception and action are provided by components for perception and action. Components make observations about the environment, interact with users, and take actions to impart changes to the environment.

In our systems, services maintain information about users and the environment in a situation model. The situation model has the form of a network of situations. Each situation has three facets: Observation, Reaction and Prediction. The observation facet specifies the entities, properties and relations needed to define the situation. This can act as a specification that serves to activate and configure a set of perception components capable of providing observations about the required entities and their relations. The reaction facet specifies how the service should behave in each situation, including both the desired state of the environment, and a specification communications that the service should make with the user. The Prediction facet indicates possible changes to the current situation, by pointing to adjacent situations and describing the events that can indicate the change.

Sensors are devices that make measurements, ranging simple devices that measure temperature or humidity, to devices that capture motion (infrared motion detectors), acoustic energy (microphones) and images (cameras) or 3D structure (range sensors, stereo vision systems). Actuators impart change on the environment. Such devices can range from information displays, control of lighting and sound systems, motorized controls for doors, windows and window blinds, as well as mobile robotic devices for logistics, cleaning or entertainment.

Components for 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 detecting, recognizing and observing people, things and events. Action components alter the environment to being it to a desired state. Tightly coupling perception and action can offer many advantages. Controlling action with perception allows a service to adapt action in accordance with the effect on the environment. Action can also be used to reconfigure the environment to improve perception, or even to probe the environment as part of perception.

B. *Components for Perception and Action*

Perception and action components are autonomous assemblies of modules executed in a cyclic manner by a component supervisor. Components communicate via synchronous data streams and asynchronous events in order to provide software services for action or perception. We propose a data-flow process architecture for software components for perception and action [10], [11], [12]. Component based architectures, as described in Shaw and Garlan [13], 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 in which components can employ competing methods to accommodate sensor modes that are unreliable or available in only limited conditions.

Components are controlled by a supervisory module. The component supervisor interprets commands and parameters, supervises the execution of the transformation, and responds to queries with a description of the current state and capabilities of the component. The auto-critical report from modules allows a component supervisor to monitor the execution time and to adapt the schedule of modules for the next cycle so as to maintain a specified quality of service, such as execution time or number of targets tracked. Such monitoring can be used, for example, to reduce the resolution of processing an image by selecting 1 pixel of N [14] or to selectively delete targets judged to be uninteresting or erroneous [15].

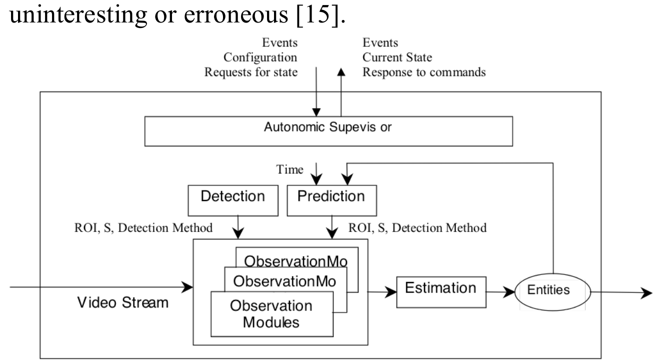

Figure 2. An example of perceptual component based on visual tracking

In addition to recognition, the supervisory component provides execution scheduling, self-monitoring, parameter regulation, and communications. The supervisor acts as a scheduler, invoking execution of modules in a synchronous manner. For self-monitoring, a component applies a model of its own behavior to estimate both quality of service and confidence for its outputs. Monitoring allows a process to detect and adapt to degradations in performance due to changing operating conditions by reconfiguring its component modules and operating parameters. Monitoring also enables a process to provide a symbolic description of its capabilities and state.

Homeostasis or "autonomic regulation of internal state" is a fundamental property for robust operation in an uncontrolled environment. A component is auto-regulated when processing is monitored and controlled so as to maintain a certain quality of service. The process supervisor maintains homeostasis by adapting module parameters to maximize estimated quality of service. For example, processing time and precision are two important state variables for a tracking process. Quality of service measures such as cycle-time, number of targets, or precision can be maintained by dropping targets based on a priority assignment or by changing resolution for processing of some targets.

During the communication phase, the supervisor may respond to requests from other components. These requests may ask for descriptions of process state, process capabilities, or may provide specification of new recognition methods. The supervisor acts as a programmable interpreter, receiving snippets of code script that determine the composition and nature of the process execution cycle and the manner in which the process reacts to events. Recognition procedures are small procedures interpreted by a lightweight language interpreter [16]. In our implementation, such procedures may be preprogrammed or they may be downloaded to the component during configuration as snippets of code using a lisp-like language.

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. The situation model allows the system to focus computing resources, to provide missing information, and to determine appropriate or inappropriate system actions for the current state of the activity.

Perceptual components communicate using Streams, Events, and Queries. Streams are synchronous communication channels for communicating continual data such as image frames or acoustic signals. An important role for perceptual components is to process streams in order to observe entities and their properties. Events are asynchronous messages generated by components in response to changes in entities or their properties. Events may be sent to other components or to the situation model. Queries are communication transactions in which a service, the situation model, or another component exchange messages with the component supervisor in order to interrogate a component about its entities and their properties.

C. *Assembling Components to Provide Services*

We have constructed a middle-ware environment [17] that allows us to dynamically launch and connect components on different machines. This environment, called O3MiSCID, provides an XML based interface that allows components to declare input command messages, output data structures, as well as current operational state. In this environment, a user service may be created by assembling a collection of perceptual components.

Available components are discovered by interrogating an component data-base. An open research challenge is to provide an ontological system for indexing components based on function in a manner that is sufficiently general to capture future functionalities as they emerge. In addition the component data-base provides information about message formats and data types for communication of streams, events and queries.

Figure 3 shows a simple example of a service provided by an assembly of perceptual components. This service integrates information from multiple cameras to provide 3-D target tracking. A set of tracked entities is provided by a Bayesian 3D tracking process that tracks targets in 3D scene coordinates. This process specifies the predicted 2-D Region of Interest (ROI) and detection method for a set of pixel-level detection components. These components use color, motion or background difference subtraction to detect and track blobs in an image stream from a camera. The O3MICID middle-ware makes it possible to dynamically add or drop cameras to the process during tracking.

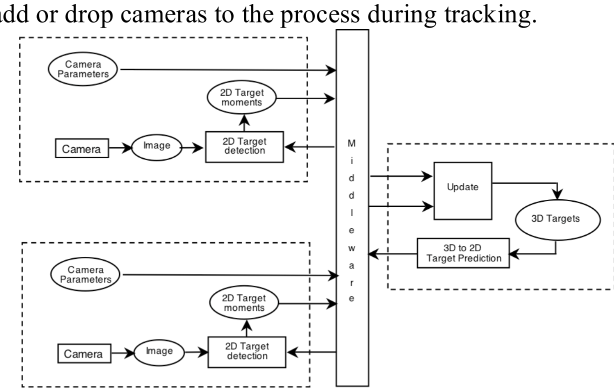

Fig. 3. An example of an assembly of perceptual components. The 3D Bayesian blob tracker provides a ROI and detection method for a number of 2D entity detection components. The result is used to update a list of 3D blobs.

D. *Entities and Relations*

Situations are defined as relations between entities. An "entity" is anything that can be observed. 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".

Formally, entities are correlated sets of observations. Entities are grounded in the software components for observation of activity, typically through some form of tracking process that correlates observations over time. Entities can be decorated with properties that make possible the determination of relations between entities.

A relation is a predicate or binary function computed on the properties of one or more entities. Relations have an arity, that specifies the number of properties that serve as arguments. An arity-1 relation is true when a property is observed to be within some range of values, or is otherwise signaled as true by a sensor. Examples can include (standing person) or (running person). Relations of Arity-2 include many of the classical spatial and temporal relations as well as more abstract functions describing social-behaviour or emotion. Spatial relations can be 2D or 3D and relative or absolute, depending on the requirements of the service. Examples can include absolute position of actors (at podium person), (seated-at table person), relative position (facing person1 person2), or even refer to the posture of persons (standing person). Observing human interaction can require perceptual components that detect more abstract social behaviour, such as (talking-to person1 person2) or (smiling-at person1 person2).

As mentioned above, the number of potential relations that might be observed is an unbounded set. The situation model for a service specifies the relations between that are required, the entities (agents and objects) that must be observed, the properties that are needed to determine relations. The task of the system designer is to provide perceptual components that can detect and track the required entities, measure the required properties, and detect when the required relations are true.

Human attention is an important relation in social situations. In our approach, we have adopted the attention model developed by Maisonnasse [18]. In this work, attention is defined as a cognitive process of selectively concentrating on one aspect of the environment while ignoring other things. We include attention of agents as one of the fundamental relations for describing social situations.

E. Generalizing with Roles

In most situations, the exact identity of the entity is not important. Thus we have generalized situation models by the introducing of the concept of "role" [5]. A role is a form of abstract model for an entity. In applying a situation model to describe a scene, a system will select from available entities to determine which entity can "fill" each role.

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.

A “role” is NOT an intrinsic property of an entity, but rather, is an interpretation applied to an entity by the system. Entities are assigned to roles by a role assignment process. Role assignment generally occurs by applying a set of tests to available entities. The role assignment process acts as a form of "filter" [19] that sorts entities based on the suitability of their properties. The most suitable entity wins the role assignment.

In our experiments for automatic learning of situation models [6], we have discovered that roles provide generalization, making it possible to greatly accelerate learning. Reactions learned for a situation composed of one set of entities can be used to understand a different set of entities.

F. Situations as Scripts for Understanding Activity

The situation model acts as a non-linear script for interpreting activity 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, entity and relations. A situation is defined as a configuration of relations over a set of entities playing roles. Thus 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 test the properties of entities that have been assigned to roles. 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.

A situation is a form of state, expressed as a logical expression (a conjunction of predicates). Situations are organized into networks, with transition probabilities, so that possible next situations may be predicted from the current situation. In our systems, the situation model drives focus of attention by specifying the entities and relations that should be attended. When a service is initiated, a list of relevant entities and relations are provided, along with the relevant configuration information. This list is used to initiate and configure the relevant perceptual and action components needed to maintain the situation model.

Each situation contains a list of expected relations, as well as expected observed entities and their expected properties. Transitions between situations can be triggered by events, and do not require verification for the entire set of relations, entities and properties. Thus it is possible for a situation to provide default values for relations, and properties that have not been verified. When interrogated by a service, a situation model may respond with the default values, whether or not these values can be currently verified. Such a response can be provided without waiting for an actual verification to occur. However, this verification can be used as an integrity check for the situation model

When a system responds with a default value, it is good practice for the system to query the relevant perceptual components to verify that the default value is correct. In some cases, this may indicate a divergence between the situation model and the environment. Such a divergence can be used to trigger a diagnostic process to recover from the current error, by adapting perception to changes in the environment or by developing the situation model by adding new situations or behaviours.

IV. 5. LEARNING SITUATION NETWORKS

We distinguish the concepts of adaptation from development [20]. *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 hidden Markov models and neural networks, require large sets of training data – something that is difficult to obtain for an uncontrolled environment. Development of context models requires new ways of looking at learning, and suggests the need for 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 [21]. 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 [22] as well as algorithms based on first-order logic [23] 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 4.

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 has been 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.

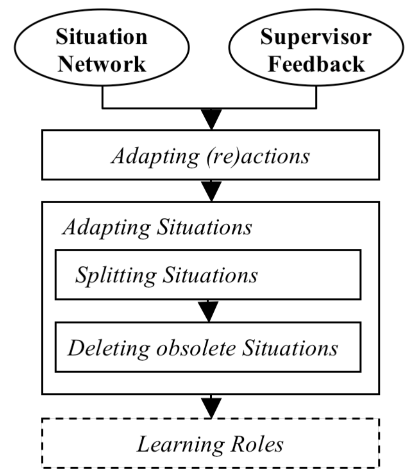

Fig 4: Overview of the algorithm for adapting system actions

V. CONCLUSIONS

Activity models for context aware services can be expressed as a network of situations concerning a set of roles, entities and relations. Roles are abstract classes for entities. Entities may be interpreted as playing a role, based on their current properties. Relations between entities playing roles define situations. This conceptual framework provides default reasoning, focus of attention, and real time response for services that require observation of human activity. This model can also provide a 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 context aware user 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.

REFERENCES

[1] P. N. Johnson-Laird, Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness, Harvard Univ. Press, Cambridge, MA, 1983.

[2] Radvansky, G. A., & Zacks, R. T. (1997). The retrieval of situation-specific information. In M. A. Conway (Ed.) Cognitive Models of Memory, pp. 173-213. Cambridge, MA: MIT Press.

[3] Zwaan, R. A. Radvansky, G. A., "Situation Models in Language Comprehension and Memory, PSYCHOLOGICAL BULLETIN, VOL 123; NUMBER 2, pages 162-185, 1998.

[4] P.N. Johnson-Laird, Mental models, MIT Press Cambridge, MA, USA, 1989.

[5] J. L. Crowley, "Context Driven Observation of Human Activity", European Symposium on Ambient Intelligence, Amsterdam, 3-5 November 2003

[6] J. L. Crowley, O. Brdiczka, and P. Reignier. Learning Situation Models for Understanding Activity In The 5th International Conference on Development and Learning 2006 (ICDL06), Bloomington, Il., USA, June 2006

[7] O. Brdiczka, J. L. Crowley, P. Reignier, Learning situation models for providing context-aware services, in "IEEE Transactions on Man, Systems and Cybernetics, Part B", Volume 38, Number 1, January 2008.

[8] F. Metze, P. Gieselmann, H. Holzapfel, T. Kluge, I. Rogina, A. Waibel, and M. Wolfel, J. Crowley, P. Reignier and D. Vaufreydaz, F. Bérard, B. Cohen, J. Coutaz, V. Arranz, M. Bertran and H. Rodriguez, "The FAME Interactive Space", 2nd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms, MLMI, Edinburgh, July 2005.

[9] M. Danninger, T. Kluge, R. Stiefelhagen, "MyConnector: analysis of context cues to predict human availability for communication", International Conference on Multimodal Interaction, ICMI 2006: pp12-19, Trento, 2006.

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

[11] 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.

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

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

[14] J. Piater and J. Crowley, "Event-based Activity Analysis in Live Video using a Generic Object Tracker", Performance Evaluation for Tracking and Surveillance, PETS-2002, Copenhagen, June 2002.

[15] D. Hall, R. Emonet, and J. L. Crowley, "An automatic approach for parameter selection in self-adaptive tracking." In International Conference on Computer Vision Theory and Applications (VISAPP), Setubal, Portugal, Feb. 2006.

[16] A. Lux, "The Imalab Method for Vision Systems", International Conference on Vision Systems, ICVS-03, Graz, april 2003.

[17] R. Emonet, D. Vaufreydaz, P. Reignier, J. Letessier, "O3MiSCID: an Object Oriented Opensource Middleware for Service Connection, Introspection an Discover", 1st IEEE International Workshop on Services Integration in Pervasive Environments - June 2006.

[18] J. Maisonnasse, N. Gourier, O. Brdiczka and P. Reignier, Attentional Model for Perceiving Social Context in Intelligent Environments, Artificial Intelligence Apllications and Innovations 2006.

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

[20] 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.

[21] R. Barraquand and J. L. Crowley, "Learning Polite Behavior with Situation Models", Third International Conference on Human Robot Interaction (HRI 2008), 12-15 March 2008, Amsterdam, The Netherlands

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

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