AI Communications 26 (2013) 303–316
DOI 10.3233/AIC-130567
IOS Press

Engineering of holonic multi agent intelligent forest fire monitoring system

Ljiljana Šerić *, Maja Štula and Darko Stipaničev
Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, Rudera Boskovica 32, 21 000 Split, Croatia
E-mail: {ljiljana.seric, maja.stula, darko.stipanicev}@fesb.hr

Abstract. In this paper we describe holonic multi agent architecture of iForestFire (Intelligent Forest Fire Monitoring System) through phases and activities of the holonic software engineering process. iForestFire is a web-based information system used in all aspects of forest fire management. Its central part is an artificial perception system, observer network, whose aim is early detection of forest fires. Although the agents that the system consists of are inseparably connected, they can be clustered in several independent systems that can be used separately. Independent holons are built from agents that are part of the observer network, as well as other agents with auxiliary functionalities.

Keywords: Multi agent system, holons, observer network, agent oriented software engineering

1. Introduction

Multi agent system is convenient paradigm for designing distributed systems, but as complexity of the system increases, the number of individual agents also increases. Adding new functionalities is usually done by adding new kinds of agents and in that case multi agent architecture becomes less clear, and more appropriate architecture could be holonic multi agent architecture [3]. Holonic multi agent architecture contains less individual elements each of which represent a group of agents designated by its group common goal. Holonic architecture also highlights structure and reusability, thus individual holons can be used separately as self contained system or as parts of another system where particular functionality is needed.

iForestFire is an example of such a complex system. iForestFire is a web-based information system [7] designed to assist in all aspects of fire protection. The system can be used as a helping aid in various fire protecting activities involving preventive activities, decision support system during the fire and the recovering activities after the fire. Motivation for implementation of such a system was found in the fact that Mediterranean countries are most threatened by fire hazards, especially in summer period. Early detection of forest fire is critical in minimizing the damage, so the system is primarily used for monitoring and forest fire detection. iForestFire consists of its hardware components – a sensor network deployed in the environment with the auxiliary hardware components (i.e. communication equipment), its software components – multi agent system for data gathering and processing and the user interface. User interface is completely web based so standard web browser is used as a client tool. Operator situated in operation center is presented with information gathered on field by sensor network, as well as a conclusion about possible fire threat.

iForestFire has already been described in other previously published papers, but from different aspects. In this paper we will describe the software engineering process of holonic multi agent architecture of iForestFire system, with the special emphasis on system organization and components. The software engineering process will be described through three phases. The three phases are present in the majority of software engineering processes (analysis, design and implementation) but the content of those three phases is adopted from ASPECS [5] where phases are analysis, agent society and implementation and deployment. The ASPECS is chosen among other agent oriented software engineering (AOSE) processes because this particular process emphasizes hierarchy and organization which is almost always present in complex sys-

*Corresponding author. E-mail: ljiljana.seric@fesb.hr.

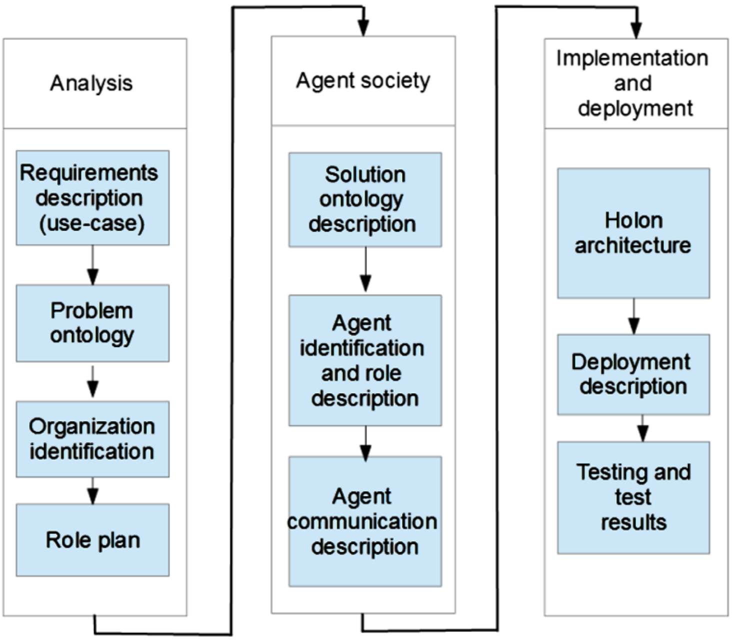

Fig. 1. Phases and activities of iForestFire system development. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-130567.)

tems [4]. In the scope of this paper we will reduce the number of activities from each phase to simplify the software engineering process. We will not discuss every activity and task of the software engineering process included in phases described in [5], but only the selection of outcomes will be presented and described. Activities of each phase of the software development process that will be discussed in this paper are shown in Fig. 1.

2. Intelligent forest fire monitoring system analysis

Traditional approach to forest fire monitoring is locating human observers on monitoring stations looking for forest fire using its own senses. The problem of human observers is well known – humans are expensive, subjective and easily tired. iForestFire system is designed to simplify the process of early fire detection by locating video camera and meteorological sensors on monitoring station. Human operator is situated in front of a computer screen where results of data analysis and understanding are presented. When fire is detected, the alarm is raised. The operator can than use manual control of the camera to inspect the situation on field in more detail and make final decision about occurrence of forest fire. A great effort is made in minimizing false alarms without losing reliability of the system, that is without increasing probability of missed fire. This can not be performed without reliable data from sensors, since good decisions can be made only based on accurate data. So great efforts are focused on data validation and alarm post processing.

Domain analysis consists of several activities that must be completed before activities from the next phase are initialized. Phases of ASPEC process are sequential. Outcomes of analysis phase that will be discussed here are: requirements description in the form of use-case diagrams, domain ontology, role plan and organization identification.

Use case diagram consists of use-cases defined in cooperation with the users, which are, in our case, firefighting troops and firefighting commanders. The central and most important use case is *forest fire detection*, but the *distant video presence* use case is equally important. This mode enables the manual control of the camera. Besides data from video camera, system inputs are also meteorological data from mini weather station located on the same location as video camera and GIS (Geographical Information System) information. *Meteorological parameters* are displayed to the user, and *GIS data* can be browsed from within the system. *Archive data* must be available for browsing and analysis.

From the administrators' point of view, *sensor administration* use case and *detection parameters adjust-*

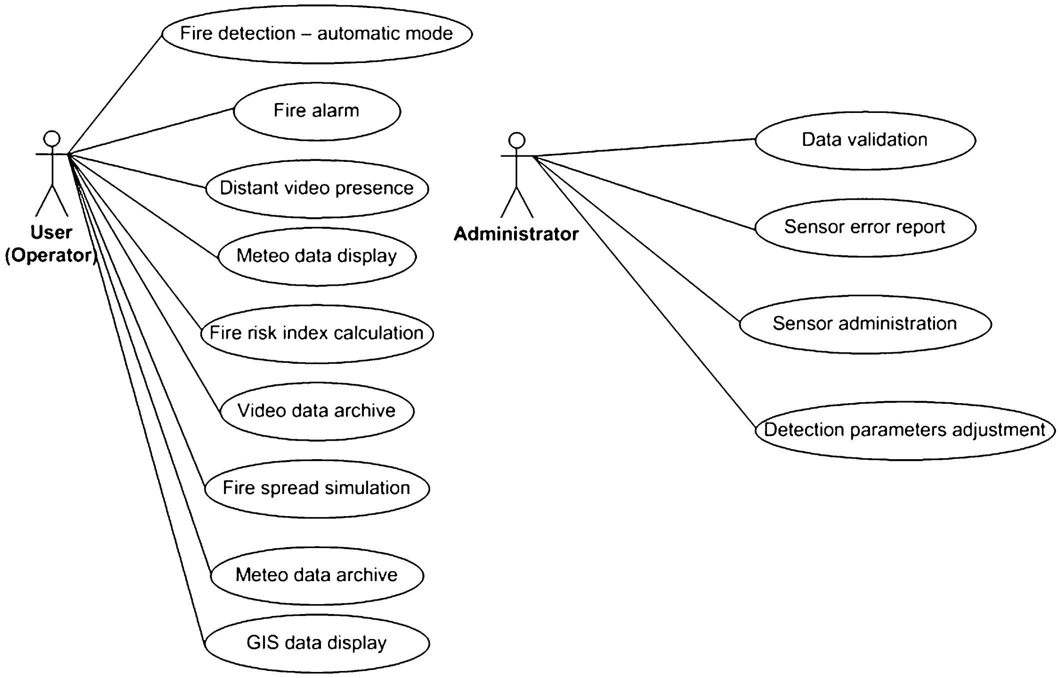

Fig. 2. Use case diagram of iForestFire system.

ment are highlighted. *Automatic data validation* and *sensor error automatic report* are also interesting. The use case diagram is shown in Fig. 2.

The *problem ontology* consists of identified problem concepts and their relations. The hardware part of system is composed of sensors and cameras mounted on particular locations in environment forming sensor network. Optimal locations of cameras and sensors are determined using Analytic Hierarchy Process (AHP) as multi criteria decision making procedure and intensive use of GIS data as described in [13]. Sensors provide meteorological data and cameras provide image of the environment. By appropriate image and data analysis the conclusion about the scenario taking place in the environment is described. The similar process performed by humans is called perception. The other important feature of perception is validation of data before it is used in making conclusions about environment scenario, so the concepts regarding validity of the data are identified here. According to the problem ontology, the system will perform artificial perception. In this paper we will focus on artificial perception core of the iForestFire system. The illustration of problem ontology is shown in Fig. 3.

The "leading role" of the iForestFire system is the role of *artificial perception* and it will be performed by

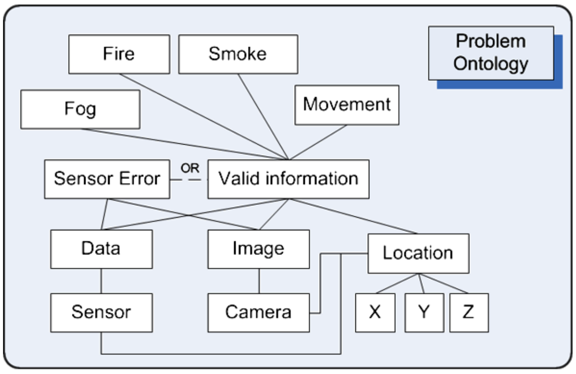

Fig. 3. Problem ontology. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-130567.)

the observer network. *Observer network* [12] is an artificial perception system where sensor network plays the role of sensors and sensor network data are processed on several levels. The aim of perception is creating the mental image and description of the scenario taking place in the environment based on stimulus of senses. Since biological principles of perception are rather complex and not investigated enough, in this work we are relying on the formal theory of perception and establish goal of simulating the final outcomes of biological perception. Observer network is formally described using the formal theory of perception [1].

The sensing part of the observer network is a sensor network that can be formally described as a set of sensors, $SN = \{S_1, S_2, S_3, \ldots\}$. An observer network can be described as a set of observer nodes $ON = \{N_1, N_2, \ldots\}$, where observer node is a set of sensors situated on the same location, thus providing data about the same environment. Observer node can be formally described with $N_i = \{S_1, S_2, S_3, \ldots\}$, where $S_1, \ldots, S_i$ are sensors that can be found on that specific location.

Another feature of observer node is its geographical location described with its geographical coordinates and additional data assigned to that coordinates (e.g. the vicinity of roads, electrical cabling, water resources etc.). The GIS data can be formally described with $L(N_i) = \{x, y, z, \ldots\}$.

In a specific time moment t, a sensor provides data labeled with $D(S_i, t)$. The sensor data can be either:

- simple data – data in the form of a single value, like temperature or humidity,
- advanced data – data in the form of file, image, video or audio data,
- virtual sensor data – there is no physical senor, but data is calculated using existing sensor data, like pressure on the sea level or wind chill. Virtual sensor data is mathematically modeled as $S_v = f(S_1, S_2, \ldots)$.

In human perception, two major conclusions are made. First one is a conclusion about the correct operation of sensors and validity of sensor data by performing self checking or proprioception. The other conclusion is made about the environment in exterioception. Observer network perception follows these conclusions and performs perception in two steps. Proprioception or self checking is performed by the network observer, and is done in two steps – first step is validating syntactic content of every sensor data, and the other is validating semantically the content of node data. Syntactic validation referres to checking if the current data satisfies the expected form. This includes checking if the data has expected number of digits, if its value is inside expected range and similar. Set of syntactically valid data for a sensor S_i is called *Syn*(S_i).

Semantic validation implies checking the meaning of the data in the context of other sensor values. In semantic validation the context is created by the data from several sensors. The set of data from sensors forming one observer node is called node information. Node information is described with $I(N_i, t) = \{D(S_j, t) : S_j N_i\}$. Formally, a set of semantically valid data defined over observer node is *Sem*(N_i).

Phenomenon checking is the true goal of the observer network. Using the sensor data that is organized in the observer node information, observer network performs analysis and alarm post-analysis and creates description of a scenario taking place in the environment. Phenomenon is defined with the set of observer node information that implies occurrence of phenomenon, *Ph*(N_i). If the observer node information is an element of this set, conclusion of the observer network is that the phenomenon has occurred in the environment. If the information is not an element of *Ph*(N_i), conclusion is that the phenomenon did not occur. Additional description of the scenario can be formally included in the results of an observer network, by assigning additional descriptions to the each element of *Ph*(N_i). In the case of iForestFire, the observer network is used for perception of a forest fire.

iForestFire offers others functionalities besides the observer network used for early detection of the forest fires. Video and meteorological data is displayed on the web interface. The system enables virtual video presence by allowing manual camera control. Archive data is stored for post-analysis. The system makes use of the static GIS data, particularly topology layer, vegetation layer and the vegetation burning characteristics layer. In addition, dynamic layer with fire risk index is calculated in real time. GIS data is also used for the fire spread modeling and prediction.

Organization of the iForestFire system can be considered from two aspects. The first aspect is the horizontal organization that identifies various functional components, and the second aspect is the vertical organization that is present in horizontal components. Functional components of the system from the user's point of view are shown in Fig. 4. Described components are presented to the user via the web graphical user interface. iMeteo component provides a network of meteorological stations and meteorological data acquisition and storing functionality. iVideo provides network of video cameras covering the region of interest. Images from video cameras are acquired cyclically in automatic wildfire detection mode, but video cam-

Fig. 4. iForestFire components. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-130567.)

eras are also used in manual mode to enable distant virtual video presence. GIS data are used in most components [13], but they could also be used separately providing the user with all the geographical information of the monitored area. Web GIS interface implements standard functionalities for GIS data browsing [13]. Archive data enables browsing through stored meteorological and video data, including the search mechanisms like date and time interval, location and preset position. Observer network component [12] is the vital component of the iForestFire system because it provides early forest fire detection. The component is designed as an artificial perception system [12]. The network of meteorological stations and video cameras plays the role of perception senses. Intelligent mechanisms composed of various agents perform calculations based on the algorithms trying to have the same results as human perception. Automatic detection results can be viewed as a part of an observer network, or independently. The reason we highlight it separately is because various alarming systems and user interface has reusability features. Micro location fire risk index (MIRIP) component is used for micro location forest fire risk index calculation [7]. From the aspect of an observer network formal description, fire risk index can be considered as a virtual sensor data that is used in identification of an area that is endangered. The component is used in preventive activities suggesting the increase of the attention for locations with higher risk index. Model of fire propagation (MOPP) component is used for the forest fire spread simulation and modelling [7]. This component is used for fire spread and behaviour prediction in decision-making during the fire but also in post-fire activities and for training purposes. MOPP makes use of the GIS data as well as meteorological data provided by the sensor network.

Vertical organization of the system includes software architecture of the observer network [12] which is divided in three layers. Particular conclusions are the result of the interaction between the neighboring layers. Layers are named following the concept of information spectrum, data, information and knowledge layer, where higher layers are focused on more sophisticated particles of information than the lower layers. Apart from the observer network architecture, the interface layer is shown as being parallel with the information and knowledge layers. The reason why this layer is separated from the observer network lies in the various interfaces for information layer and knowledge layer results representation. In addition, the interface layer is not shown above the knowledge layer because the communication of the interface elements is performed both with the information and the knowledge layer elements. The system architecture is illustrated in Fig. 5.

Sensor network generates row data that is considered on the lowest layer, the data layer. Sensor network consists of meteorological station and video cameras. On the information level, data from the sensor network is collected and validated on several levels forming information from each observer node. On the knowledge layer, information is analyzed and knowledge about the scenario taking place in the environment is created.

Conclusions about the sensor network valid functionality are made during the interaction between the data and the information layer, while the conclusions about the existence of the phenomenon according to available information is driven on interface between the information and the knowledge layer.

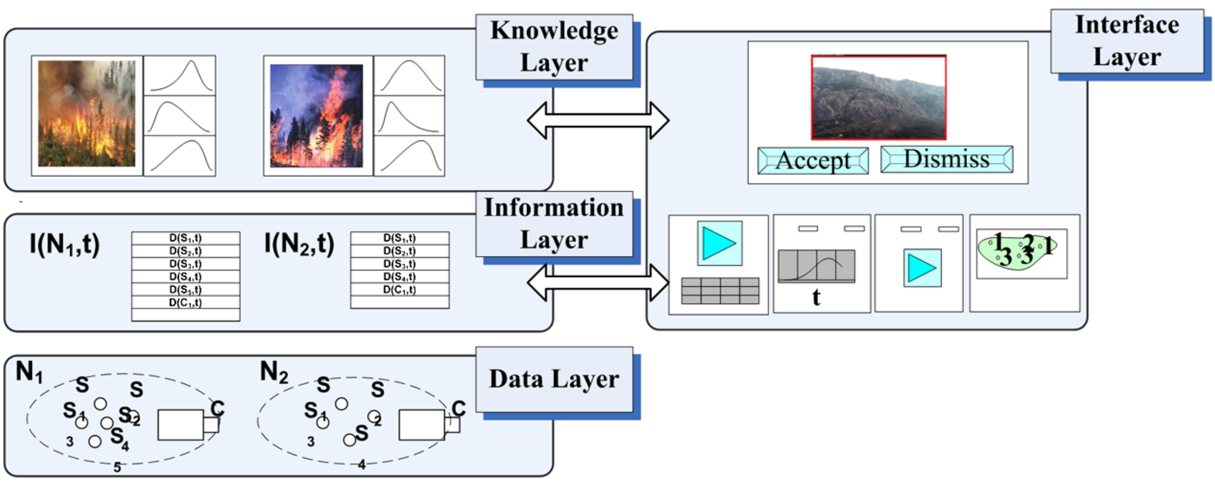

Fig. 5. Observer network architecture. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-130567.)

3. Agency domain

The first step of the agency phase is the identification of the solution ontology. The solution ontology of the iForestFire system is based on the role plan and is organized in three layers.

The solution ontology is intensively used as common ground in agent communication. The concepts are used in the knowledge base that is defined for data and information validation as well as the knowledge base for phenomenon perception. The observer network ontology can be considered at three levels, as shown in Fig. 6.

The lowest level of ontology is physical sensor network level. Concepts at this level are related to the sensor network organization and placement in the environment. Physical sensors are gathered around one or more master sensors which are advanced devices capable for communication. Sensors of the same physical location are forming one observer node. Additional important information about the location is stored in the form of the GIS data and can be obtained by querying GIS data.

Information layer ontology is consumed with the consistency and validity of the data. The concept *Sensor* corresponds to one kind of data that is available about the environment (e.g. temperature or humidity).

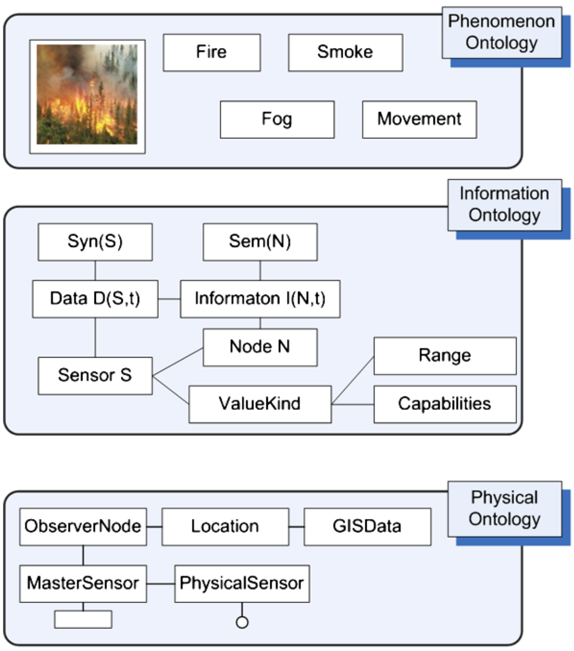

Fig. 6. Illustration of solution ontology. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-130567.)

The number of physical sensors is not the same as the number of instances of the Sensor concept because this concept includes virtual sensors as well.

On phenomenon level there are concepts related to the phenomenon of interest for the concrete observer network system. While the first two levels' concepts are common for all observer network application, these concepts depend on concrete application of the observer network and must be redefined for each use of the observer network. In the case of the forest fire observer, these concepts are related to the forest fire detection and include concepts such as smoke, motion, fog, etc.

3.1. Agent identification and agent role description

When choosing technology for the implementation of the observer network, authors gave advantage to the multi agent technology because of several reasons. Amongst the advantages of the multi agent technology are the necessity for the independent execution of individual tasks and the need for communication of software entities. However, multi agent technologies are not the only possible choice for implementation of the observer network. Another approach that can be used is a simple object oriented approach. However, this approach is not a very good option because the communication of individual objects through function callouts is not an elegant solution. More elegant solution is a multi agent system where individual agents are assigned individual tasks, and according to their tasks the agents communicate within the multi agent environment. The task of each agent can be linked to the particular layer of an observer network architecture. The implementation of all the necessary agents builds the observer network multi agent shell, and during run time, agents are instantiated as needed. Apart from the observer network multi agent shell there are user interface agents. User interface agents are built for displaying results to user or collecting user inputs that are shown in a separate layer called interface layer. They are functionally distributed on information and knowledge layers. Multi agent system is illustrated in Fig. 7.

MasterAgent is situated on the data layer of the observer network and is closely connected to sensors. The name is chosen because this agent masters several sensors. MasterAgent usually runs on a piece of hardware that is situated directly in the environment, i.e. on the embedded data server or on an advanced sensor, and has the task of presenting the sensor data when asked for it.

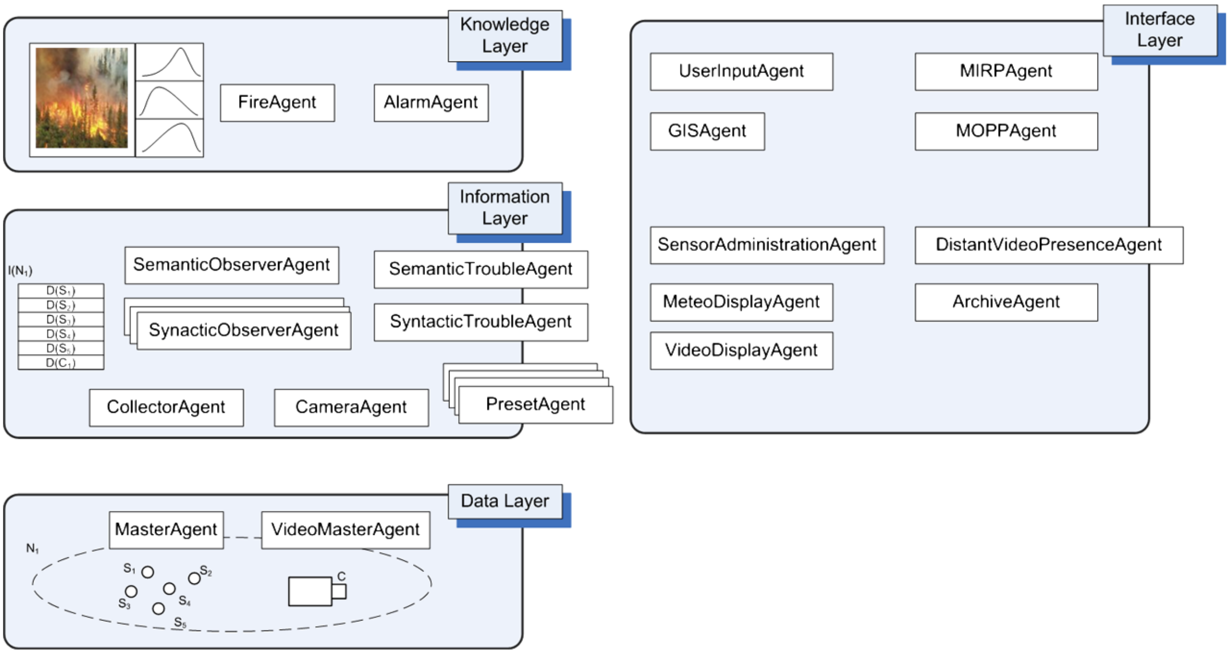

Fig. 7. Multi agent observer network shell. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-130567.)

CollectorAgent provides interface between the data and the information layer. CollectorAgent collects data from sensors of the same observer node through communication with the MasterAgent [2]. Its task can be described as follows:

$$\textit{CollectorAgent}:$$

$$\forall S_j, S_j \in N_i getData(S_j, t). \quad (1)$$

CameraAgent and ***PresetAgent*** are special kinds of a CollectorAgent, intended to collect video images. If a camera is pan/tilt/zoom movable, there are several predefined or "preset" positions on which images are acquired and the collection of images comes down to positioning the camera in a specific position and then capturing the image. CameraAgent supervises the camera and takes care of the next preset position in cycles. PresetAgent positions the camera and captures the image. Relationship between the CameraAgent and the PresetAgent and the illustration of how the surrounding 360 degrees environment is covered with 8 preset positions is illustrated in Fig. 8.

SyntacticObserverAgent performs syntactic validation of data collected by CollectorAgents. Formally this means checking if the value acquired from the sensor is within the set of expected values defined by $Syn(S_j)$. We can write:

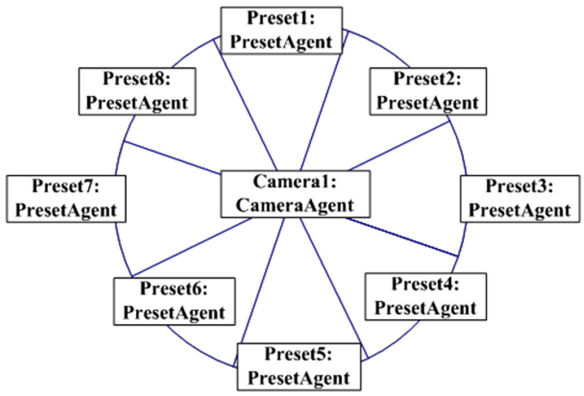

Fig. 8. Illustration of CameraAgent and PresetAgent relationship. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-130567.)

$$\textit{SyntacticObserverAgent}:$$

$$\text{if } \big(D(S_j, t) \notin Syn(S_j)\big)$$

$$\Rightarrow \textit{SyntacticTroubleAgent}. \quad (2)$$

Virtual sensor data is actually calculated as a function of other sensor data. This means that the virtual sensor data validity depends on the validity of other sensor data. It is not necessary to check the validity of the virtual sensor data, but this data is checked nonetheless to keep the consistency of the agents. Sometimes it is easier to spot the irregularity of the sensor data by having the invalid virtual sensor data, although the original simple data was syntactically valid. This kind of event is similar to the semantic error since the data itself does not appear invalid, but in the context of other data it appears illogical.

SemanticObserverAgent validates the information of the node by checking if the information composed of all sensor values is from the set of semantically valid information for that observer node. Formal description of this process is:

$$\begin{aligned}&\mathit{SemanticObserverAgent}:\\&\quad \text{if } \big(I(N_i,t) \notin \mathit{Sem}(N_i)\big)\\&\quad \Rightarrow \mathit{SemanticTroubleAgent}.\end{aligned} \tag{3}$$

Usually the sets $\mathit{Syn}(S)$ and $\mathit{Sem}(N)$ are not explicitly defined, but there is a knowledge base of rules built to be used in syntactic and semantic validation. In the case of simple meteorological value, syntactic validation would include checking if the value is within the limits of minimum and maximum values for the specific kind of sensor value. Kind of value for the sensor can be the temperature, humidity, pressure, etc.

$$\begin{aligned}&\mathit{Def}(\mathit{Syn}(S_i)):\\&\quad D(S_i,t) \neq \emptyset;\\&\quad D(S_i,t) < \mathit{getProperty}\big(\max, VK(S_i)\big);\\&\quad D(S_i,t) > \mathit{getProperty}\big(\min, VK(S_i)\big);\\&\quad \mathit{Abs}\big(D(S_i,t) - D(S_i,t-1)\big)\\&\qquad < \mathit{getProperty}\big(\max\, delta, VK(S_i)\big).\end{aligned}$$

Syntactic validation of the image from the particular preset position of the camera would include checking if the camera was positioned precisely during the image capture. This is done by checking the position of the characteristic regions on the image that must be defined by an expert for every preset position. The characteristic regions of one preset position are shown in Fig. 9.

An example of the semantic validation tests is checking the time of the day: by checking the level of light on the image, it can be concluded whether the image was taken during the day or night. By comparing these results with the current time and date, and calculating the sunrise and sunset time for the current date, we can simply check whether or not the image is semantically valid.

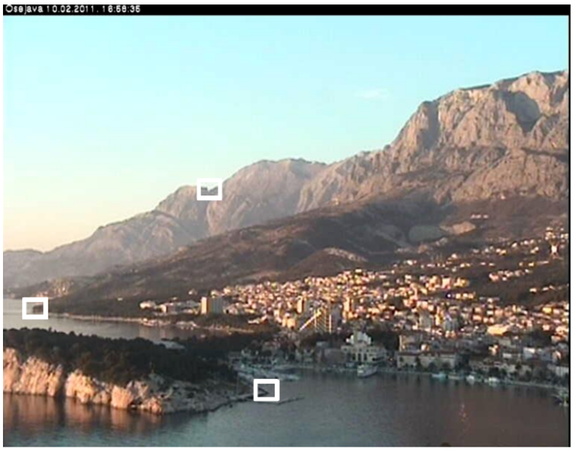

Fig. 9. Characteristic regions for preset checking. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-130567.)

$$\begin{aligned}&t \leqslant \mathit{sunrise} \quad \Rightarrow \quad \mathit{night},\\&\mathit{sunrise} < t < \mathit{sunset} \quad \Rightarrow \quad \mathit{day},\\&t \geqslant \mathit{sunset} \quad \Rightarrow \quad \mathit{night}.\end{aligned}$$

SyntacticTroubleAgent and ***SemanticTroubleAgent*** are agents performing predefined procedures in case that error in data is spotted on either syntactic or semantic level. Part of these agents' role is checking which particular sensor is responsible for the error. They also perform repairing procedures and write reports for the user. Repairing procedures include hardware reset of the master device of the sensor or of the sensor itself if it requires separate powering.

VideoDisplayAgent and ***MeteoDisplayAgent*** are presenting real time information collected from sensors, while ***ArchiveAgent*** enables browsing through stored historical data. SensorAdministrationAgent is used when upgrading the sensor network. ***Disstant VideoPresencecAgent*** enables the control of a movable camera.

Phenomenon checking agent in iForestFire system is named ***FireAgent***, and it is responsible for forest fire detection. This agent uses several algorithms for image processing fire detection and alarm post-processing using meteorological information, together with GIS data and other meta-information. Since image processing algorithms are not in the focus of this paper, the algorithms will not be described in more detail. The set $\mathit{Ph}(N_i)$ is defined by all algorithms included in the fire detection process. This set holds all the values of information of an observer node where forest fire is detected.

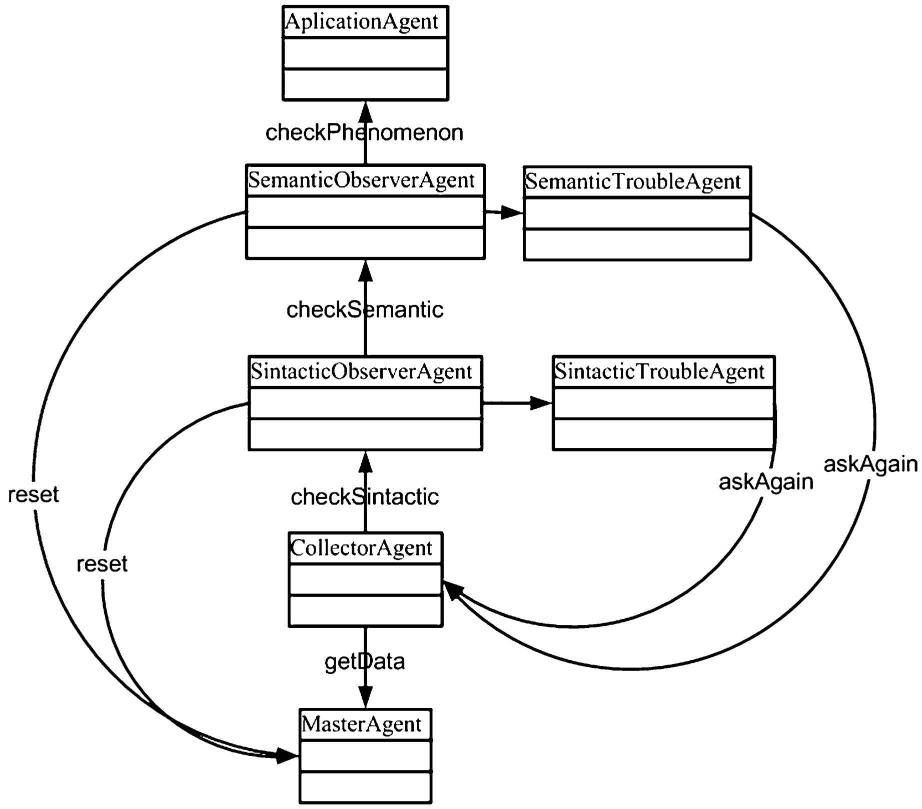

Fig. 10. A part of the communication identification diagram.

FireAgent:

$$\text{if } \big(i(N_i, t) \in Ph(N_i)\big) \Rightarrow AlarmAgent. \quad (4)$$

AlarmAgent is responsible for raising alarms to a user and ***UserInputAgent*** collects user feedback about the alarm. ***GISAgent***, as a part of interface layer, is responsible for presenting GIS data to the user. ***MIRIPAgent*** presents results of micro location forest fire risk index and ***MOPPAgent*** is an interface for fire simulation and modeling.

Communication amongst agents in multi agent shell runs both ways through architecture. A part of the communication identification diagram is shown in Fig. 10.

4. Implementation and deployment

In implementation and deployment domain we will identify and describe individual holons, consider the technologies used in implementation and describe the hardware components used in system deployment.

4.1. Holon architecture of the system

The multi agent system is clustered into several holons [3]. In the scope of this paper, holons are defined as individual parts of the multi agent system, consisting of several agents whose cooperation fulfills certain task that can be isolated as a separate independent system. Meteo holon, Video holon and the Detection holon will be described, although other holons (e.g. Simulation holon or Risk Index holon) can be identified as a part of the iForestFire system. Other holons do not rely entirely on the concept of the observer network multi agent system and will not be described in detail in the scope of this paper. Agents forming Meteo holon are shown in Fig. 11.

Meteo holon builds software support for *iMeteo* system, advanced ultrasonic meteorological station that can also be used separately from the iForestFire system. This holon consists of agents from the observer network shell, that are situated both on the data and on the information layer, and of several agents from the interface layer. Collective goal of agents is to provide current validated meteorological data. Current meteo-

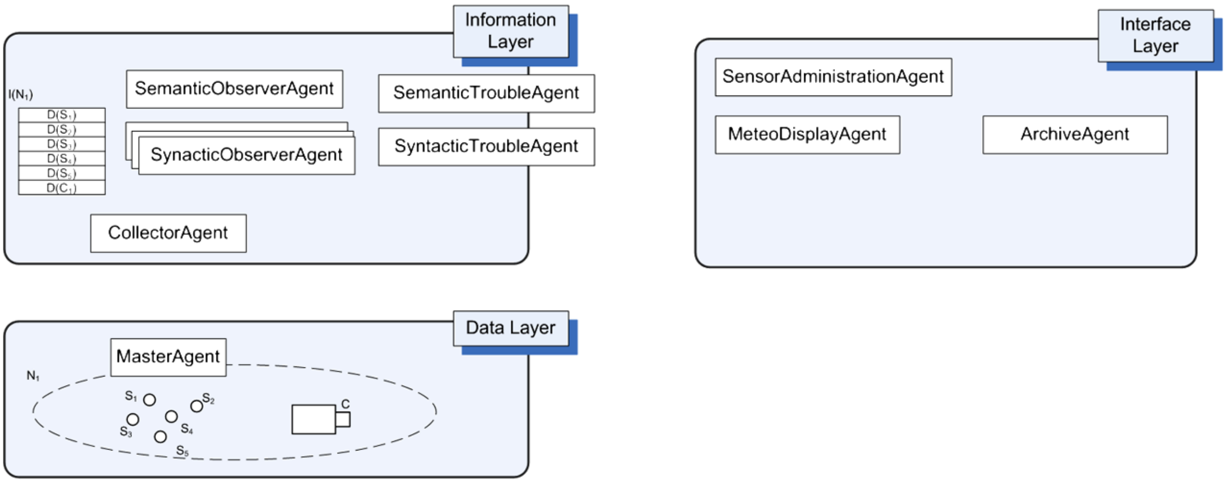

Fig. 11. Meteo holon. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-130567.)

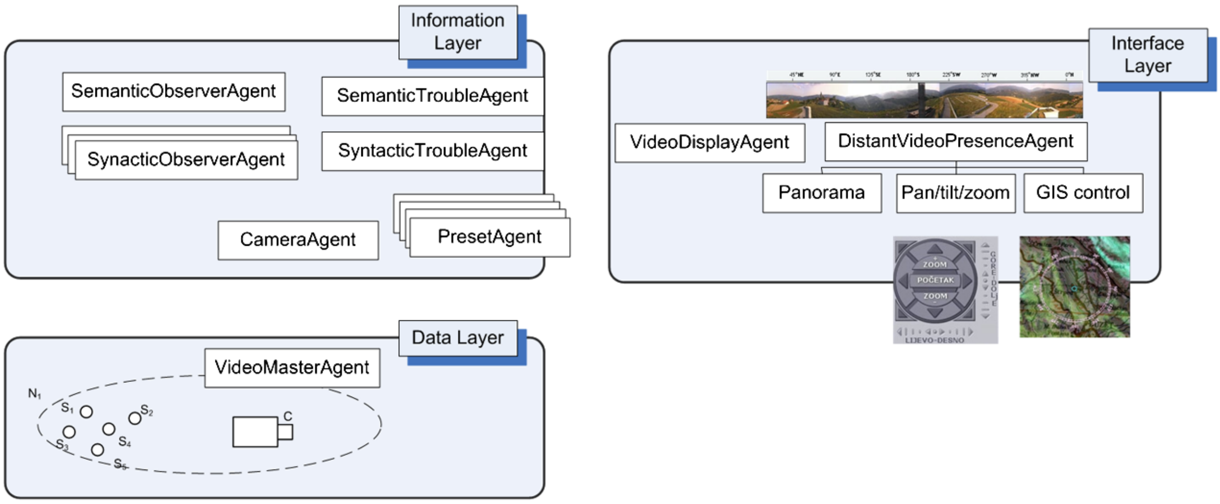

Fig. 12. Video holon. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-130567.)

rological data is displayed and can be used in other applications besides forest fire detection.

Video holon is a software part of *iVideo* system, i.e. intelligent video camera controlled with multi agent system having two modes of work: automatic when images from the camera are collected and stored cyclically, and manual when control of the camera is given to the user. Video holon is shown in Fig. 12.

Video holon consists of those agents of the observer network that are concerned with collecting images from camera. In the manual mode, automatic camera control has to be stopped, and the user interface with camera controls is activated. Besides simple pan/tilt/zoom controls, advanced user interfaces for camera control are also implemented. Some of these advanced user interfaces are panorama control (where user selects region on a panorama image of the area surrounding the camera), GIS control (where a desired area is selected on the map and pan, tilt and zoom factors are calculated with the use of elevation, distance and line of sight parameters).

Detection is the last step of perception. Based on validated information from the sensors, the scenario is labeled either as having the phenomenon or not, by combining several image processing and analysis algorithms, meteorological and the additional and learned information. Image processing algorithms include motion detection (the smoke and fire are usually growing), smoke detection during the day and fire flames detection during the night. Additional information used

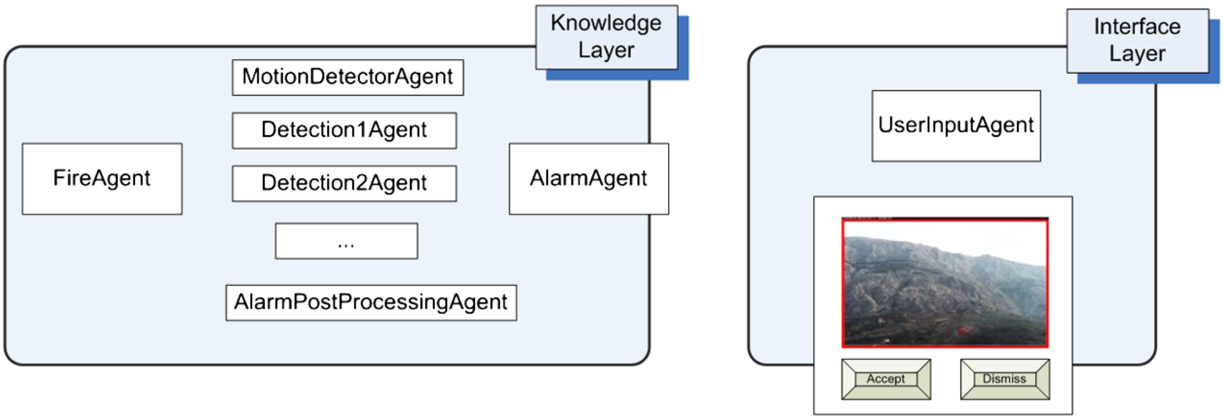

Fig. 13. Fire detection holon. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-130567.)

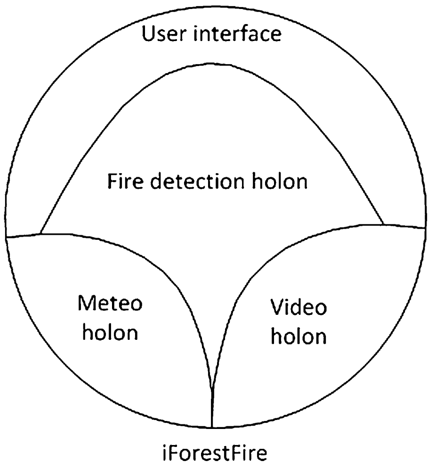

Fig. 14. Holonic solution of iForestFire.

for alarm post-analysis includes knowledge of the area (where clouds are excluded from analysis), and ignoring the alarm if the alarm was dismissed by the user in previous cyclus.

Detection is performed on the knowledge layer. Besides background detection agents, AlarmAgent is also a part of this holon that provides user insight in the results. AlarmAgent has the role of informing the user about the detected phenomenon and collecting the user feedback information about the correctness of the alarm via correspondence with UserInputAgent. User feedback is also used for enriching detection agents' knowledge base. The detection holon (see Fig. 13) can be used separately from the iForestFire system for forest fire detection on images and meteorological data obtained by any other means.

Finally holonic structure of the system can be presented as in the Fig. 14 where iForestFire system is composed of meteo holon, video holon and fire detection holon and additionally user interface holon that is built on top of holons designed for information gathering and processing and is used for presentation of their results.

4.2. Deployment description

The system deployment will be described from the hardware point of view (through description of sensor network elements and computation server) and from the software point of view (through definition of technologies used in the implementation).

Agents forming a multi agent shell are implemented in the JADE environment [9]. Agent's external knowledge is stored in knowledge base and queried using the JESS [6] engine. MasterAgent is rather specific because it executes on separate distant piece of hardware, so the implementation of the MasterAgent depends on the specific embedded device that is used. In the described system, several versions of MasterAgent are implemented, including Java Servlet, CGI and php, having different bodies, but implementing the same interface. The only demand that MasterAgent must meet in order to serve in the iForestFire system is a web interface and data display in one of the two supported formats – XML or txt properties file.

Communication between the CollectorAgent and the MasterAgent is done via standard HTTP (HyperText Transfer Protocol) request and reply. Communication among other agents is implemented in the standard FIPA ACL [14]. Agents communicate with several advanced virtual sensors via UNIX signals. Communication with the user is also performed via web interface, making the use of HTTP request parameters.

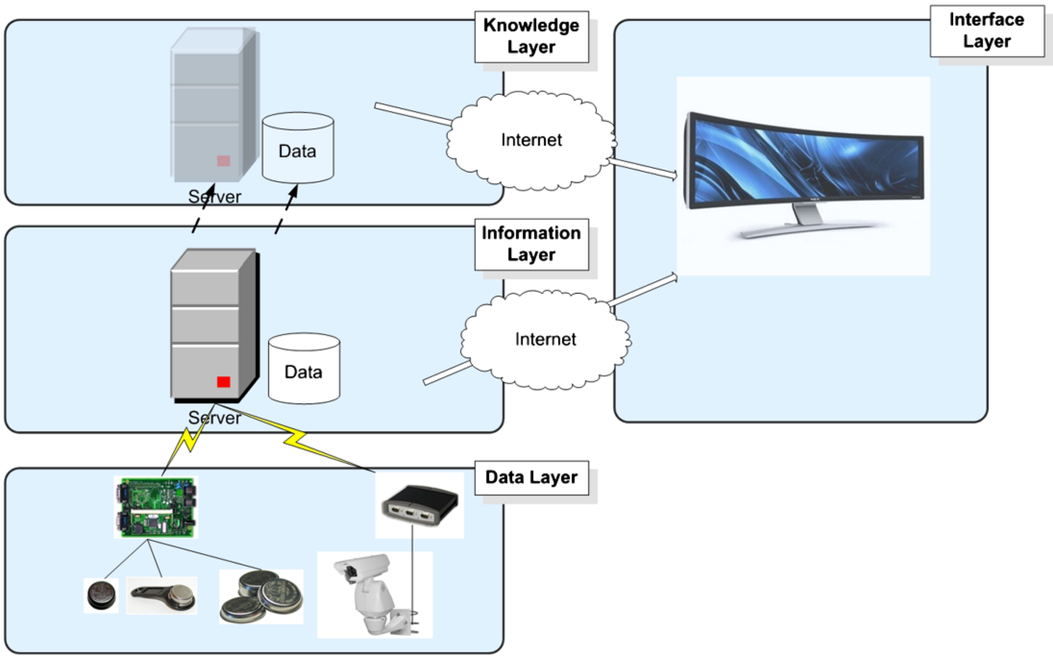

Fig. 15. Hardware elements of iForestFire. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-130567.)

User interface is a web interface, thus requesting only standard web browser and a wide band internet connection from the user. Interface is implemented using the technologies for dynamic web pages: php server scripting, client scripting with JavaScript and AJAX technologies, postgreSQL for the database.

Hardware elements of iForestFire are described as elements of the observer network three-layer architecture as shown in Fig. 15. Sensor network is assigned to the data layer. Central computational server is necessary on the information layer. On application level collected data and information and artificial perception results are coupled on computational server and presented to the user via web interface.

Sensor network consists of video camera connected and controlled via embedded video server, and mini meteorological station connected to an embedded data server. Apart from the meteorological station, additional sensors can be connected to data server, e.g. environmental sensors like lightning sensor or processing parameters sensor and actuators, like battery voltage sensor or embedded server reset switch for each embedded server at location. Reset switch is used by the TroubleAgent when the sensor data error implies that there is a problem in the operation of an embedded sensor. Embedded video server is used for publishing video data on the web, in camera control and image acquiring.

The hardware part of the information layer is a processing server with enough storage space for meteorological and processing data, video images and GIS data. The central processing server hosts agents, data bases and serves as a web sever for the web interface of the system. The same server hosts agents from the knowledge layer and web interface pages of the knowledge layer.

4.3. Testing and results

Testing and evaluation of forest fire perception system is performed through the comparison of results of an artificial perception system that is being tested and the relatively better perception system. It is impossible to identify the ideal perception system so we will rely on relatively better system – a human observer as the referent observer [12]. Evaluation of the artificial perception system can be performed throughout the online and offline evaluation scenario.

Online evaluation is performed by placing a human observer on the same location as the observer network observer node. The human observer uses his or her own senses to observe the environment, and the observer network uses sensors from the observer node. This evaluation scenario is rather unpractical because observer nodes are usually placed on difficult to reach locations, so this type of evaluation is not preferred in the iForestFire system. Less challenging version of this type of evaluation is performed when the alarm is raised and when the system asks the operator to accept or dismiss the alarm. In this case, the operator makes the conclusion using the observer node sensor readings.

Offline evaluation scenario consists of placing a human observer in front of the computer screen and of the presentation of a number of situations that were previously presented to the system of artificial perceptions. A human observer inspects the data available for each situation and notes conclusions that are later compared with the conclusions of the system.

After comparing decisions made by the human and by the artificial observer, each of the examined situations is assigned to the one of four possible sets. The four sets are:

- True positive (TP) – both observers have recognized the existence of the phenomenon.
- True negative (TN) – both observer have recognized that there is no occurrence of the phenomenon in a shown situation.
- False positive (FP) – although artificial observer has stated that the phenomenon occurred, the human observer did not recognize it. This situation is often referred to as false alarm or type 1 error.
- False negative (FN) – occurrence of the phenomenon is recognized by the human observer but not by the artificial observer. This situation is referred to as missed alarm or type 2 error.

The number of individual situations that can be examined by the human observer depends of size of the archive image and data base. Since frequency of data acquisition is rather large, the number of situations is large and it is impossible for a human observer to examine each individual situation. It is advisable to use statistical evaluation instead. However, the problem with the statistical evaluation is in determining the size of a sample set that will be statistically significant and represent the entire system archive.

In our system a method for web based statistical evaluation [11] of observer network is implemented. This method is partly based on Military standard 105E [10] which is frequently used in statistical quality control of manufacturing processes. This standard is based on the statistical analysis and provides a set of tables used to determine the sampling plan for statistical control of a bulk of products. The input data for these tables are the number of products in the bulk and the expected quality level. The sampling plan provides us with the number of products in the sample and the acceptance number of defect product which defines when will the expected quality level be met.

In the case of the iForestFire system, the number of products in the bulk is the number of individual situations in the archive image and data base. The defect product is defined as either false detection or missed detection of fire. The expected quality level is decided to be 4%, since the standard expected quality in the similar systems on the market is having no more then 10% of bad detections, and we chose first better quality.

With this input data, using a table that is part of the Military standard, we create a sampling plan which asks the human observer to examine 315 individual situations and note if the fire is present in each of the situations. The conclusion is then compared with the conclusion of the artificial perception system, and if the conclusions differ, the situation is pronounced a “bad perception”. If the number of bad perceptions exceeds 21, the expected quality is not met, and the conclusion of evaluation would be that the system has more than 4% of bad perceptions in total. If the number of bed perceptions is less then 21, the expected quality is confirmed.

iForestFire system evaluation is performed when adding new functionalities as a standard test and occasionally as a part of maintenance of the system. For the purpose of occasional evaluation of the system, the web statistical evaluation interface is created. The sampling plan is created automatically, according to the number of stored situations in the archive base. The situations that will be presented are selected using one of the implemented random algorithms and each situation is presented to the human observer together with radio buttons for noting his or her choice. After the submission, the results are presented to the user and stored in a database for future analysis, together with the identification of human observer that has performed the testing. The results of evaluation performed in March 2010 on the iForestFire observer node located on Biokovo are shown in Fig. 16. Besides the conclusion that the

Number of true detections:	TP	16
Number of false alarms:	FP	16
Number of missed alarms:	FN	0
Number of true non-detections:	TN	283

Fig. 16. Evaluation results.

system has passed the test – that does have less than 4% of bed perceptions, the important thing to notice is that in the observed sample there were no missed detections, which are more undesired situation in the case of forest fire monitoring than false alarms.

5. Conclusion

iForestFire is a distributed multi agent system that implements functionalities used as the assistance in forest fire management. During the design of the iForestFire system, great effort was made to ensure the reusability of parts of the system, so there are several independent systems that can be drawn from iForestFire system. iForestFire multi agent system is divided into specifically functional holons (Meteo holon, Video holon, Detection holon) that can be extracted from the iForestFire system and integrated into other systems that require the functionalities that the holon holds. They are implemented as a multi agent system with the reuse of certain system agents. It is important to emphasize that iForestFire is not another laboratory implementation of the multi agent system technology. It is a working system widely used in various Croatian National and Nature Parks and Istria Region, implemented on 36 monitoring units and 12 operation centers. During the period of its work numerous fires were detected.

Acknowledgements

This work was partly supported by the Ministry of Science, Education and Sport of the Republic of Croatia under Grants TP-03/0023-09 "System for early forest fire detection based on cameras in visible spectra" and 023-0232005-2003 "AgISEco – Agent-based intelligent environmental monitoring and protection systems" and Split and Dalmatia County authorities through a study "Holistic approach to forest fire protection in Split and Dalmatia County". In iForestFire system development beside authors of this paper D. Krstinić was also involved, and the main firefighter consultant was T. Vuko, the vice commander of Croatian firefighters for Adriatic Coast and Islands.

References

[1] B. Bennett, D. Hoffman and C. Prakash, *Observer Mechanics: A Formal Theory of Perception*, Academic Press, New York, 1989.

[2] L. Bodrožić, M. Štula and D. Stipaničev, Agent based data collecting in forest fire monitoring system, in: *Proc. of IEEE Conference SoftCOM'2006*, 2006.

[3] S. Brueckner, An analysis and design concept for self-organization in holonic multi-agent systems, in: *Engineering Self-Organising Systems*, LNCS, 2007.

[4] M. Cossentino, From requirements to code with the passi methodology, in: *Agent-Oriented Methodologies*, H.B. Sellers and P. Giorgini, eds, LNCS, Vol. 3690, Idea Group Pub, 2005, pp. 79–106.

[5] M. Cossentino, N. Gaud, V. Hilaire, S. Galland and A. Koukam, Aspecs: An agent-oriented software process for engineering complex systems – how to design agent societies under a holonic perspective, *Journal for Autonomous Agents and Multi-Agent Systems (JAAMAS)* **20** (2010), 260–304.

[6] E.F. Hill, *Jess in Action: Java Rule-Based Systems*, Manning Publications Co., Greenwich, CT, USA, 2003.

[7] Inteligent forest fire monitoring system, 2011, available at: http://iForestFire.fesb.hr.

[8] T. Isakowitz, M. Bieber and F. Vitali, Web information systems, *Commun. ACM* **41** (1998), 78–80.

[9] Java agent development framework, 2011, available at: http://jade.tilab.com/.

[10] Military standard 105e, sampling procedures and tables for inspection by attributes, 1989.

[11] L. Šerić, T. Bodrožić, D. Stipaničev and T. Jakovčević, Observer network web based statistical evaluation, in: *Proc. of IEEE Conference SoftCOM'2011*, 2011.

[12] L. Šerić, M. Štula and D. Stipaničev, Observer network and forest fire detection, *Inf. Fusion* **12** (2011), 160–175.

[13] D. Stipaničev, M. Bugarić and L. Šerić, Integration of forest fire video monitoring system and geographic information system, in: *Proc. of 51st Int. Symp. ELMAR*, 2009.

[14] M. Wooldridge and N.R. Jennings, Intelligent agents: Theory and practice, *Knowledge Engineering Review* **10** (1995), 115–152.