
# Paisii Hilendarski University of Plovdiv

## Faculty of Mathematics and Informatics

### Department of Computer Systems


# Research on creating a virtual operator in smart agriculture infrastructure

# ABSTRACT

of a doctoral dissertation for awarding the educational and scientific degree of Doctor (PhD)
Field of Higher Education: 4. Natural Sciences, Mathematics and Informatics
Professional Direction: 4.6. Informatics and Computer Science
Doctoral Programme: Informatics


PhD student: Ivan Stoyanov
Scientific supervisor: Prof. Asya Stoyanova-Doycheva


Plovdiv, 2023

# INTRODUCTION

The dissertation is dedicated to intelligent agriculture - a topic that is becoming an increasingly interesting object of theoretical research and practical development. In Bulgaria, the prospects for intelligent agriculture are mainly formulated in the Strategy for the Development of Artificial Intelligence in Bulgaria until 2030, as well as in the two National Scientific Programs (NNP) "Intelligent Plant Breeding" and "Intelligent Animal Breeding".

The main goal of the dissertation is to develop a personal assistant (PA) supporting farmers and agricultural specialists working in the conditions of intelligent agriculture. The PA should operate as the core of the ZEMELA platform, which is under construction. To achieve the goal, the following tasks have been formulated:

- To update the event model and propose a new version of it.
- To update the architecture of the LAND platform.
- Create a concept, model, reference architecture and lifecycle of a personal assistant for farmers.
- Prototype implementation of the personal assistant.

The research carried out during the doctoral studies is part of the work program of the NNP "Intelligent Plant Breeding" [1]. The purpose of this program is to stimulate targeted scientific and applied research for the application of artificial intelligence in agriculture. Artificial intelligence is expected to significantly contribute to the development of modern, efficient, knowledge-based agriculture to improve food quality and conserve natural resources.

In accordance with the purpose and tasks, a suitable methodology for conducting the research is proposed. The dissertation consists of an introduction, five chapters and a conclusion.

# CHAPTER ONE: STATE OF THE PROBLEM

In accordance with the thesis topic, the following are of interest:

- Cyber-physical and cyber-physical-social systems;
- The ViPS reference architecture;
- Platforms for precision and intelligent agriculture;
- Personal assistants.

In recent decades, with major breakthroughs in science, technology and engineering, humans have increasingly interacted with the physical world around them. Modern technologies of cyber-physical systems [2] and the Internet of Things [3] contribute significantly to the evolution of computer interconnectivity. They integrate the dynamics of physical processes with those of software and communication, providing abstractions and techniques for modeling, designing, and analyzing complex integrated systems.

The research conducted during the dissertation is closely related to the reference architecture called Virtual Physical Space (ViPS). The development, which started more than twenty years ago, goes through various stages. Three stages can be noted during which the Distributed eLearning Center (DeLC), Virtual Education Space (VES) and Virtual-Physical Space (ViPS) architectures were developed. DeLC [?] is a delivery environment for educational services and e-content for various forms of e-learning. The DeLC architecture is distributed and modeled as a network consisting of individual nodes called eLearning Nodes (eLNs). The most complete description of DeLC can be found in [5]. VES is an infrastructure supporting e-learning in which users, time, location, autonomy and context dependence are fundamental and which provides unified processing and interpretation of information coming from both the virtual environment and the physical world [6]. ViPS is being built as a reference architecture integrating the virtual and physical worlds [7] that can be adapted for various CPSS-like applications. In this sense, essential aspects of ViPS are the following: users are the focus of attention, virtualization of physical "things" and integration of virtual and physical worlds.

In all regions of the world, increasing efforts are being made to digitize the agricultural sector and transform it into so-called smart agriculture. A huge number of diverse projects in the field of smart agriculture are currently being developed. The FaST (Farm Sustainability Tool) project [8] aims to develop an agricultural platform providing digital services to help European farmers and agencies improve their respective capabilities in multiple agricultural, environmental and sustainability-focused activities. The Farm21

platform [9] provides field solutions that enable consistent and up-to-date access from sensors, soil, crops, weather and satellites. The FAO Regional Office for Africa is developing a regional technical platform [10] to support a community of practice of stakeholders and partners to share knowledge, best practices and lessons learned on common agricultural policies and practices and to promote collaborative efforts.

In recent years, research and practical developments related to intelligent personal assistant (IPA) have grown significantly. IPAs can be classified into two major groups: general purpose and specialized purpose. The current and highest technological level of IPA development has been achieved with Generative Pre-trained Transformer-4 (GPT-4) [11].

From the review of the state of the problem, the following conclusions were drawn, which guided the conduct of the research, the results of which are presented in the dissertation work.

• The platforms are of global purpose. They aim to solve a wide variety of tasks in the field of precision agriculture on a more global scale. These platforms are supported by serious institutions and enjoy strong funding and links to other significant projects (eg Copernicus, Galileo) from which they get the information they need.

• The ZEMELA platform presented in this dissertation is for *regional purpose*. Our platform is primarily aimed at small and medium-sized farmers. It uses data primarily from stationary sensor networks located in small outdoor blocks or greenhouses. The idea is to use the specificity of the local region to obtain optimal results. The functioning of the platform is carried out by a personal assistant using an event model and a specific presentation of specialized knowledge.

• Research on the creation of personal assistants has become increasingly relevant and significant in recent years. Certain general-purpose personal assistants are used to solve specialized tasks using their powerful language processors and pre-training capabilities. Service-oriented architectures predominate in the smart agriculture platforms reviewed. The agent-oriented approach is weakly advocated. Despite the emergence of more and better performing general-purpose intelligent personal assistants, specialized personal assistants remain an underexplored scientific field. The personal agent, which is the result of the research carried out during the doctoral studies, is an intelligent assistant with a special purpose - to support farmers working in the conditions of intelligent agriculture. We propose a different approach – it does not rely on a powerful language processor, but rather the proposed personal assistant is oriented towards using repositories of specialized knowledge represented as semantic networks (ontologies), rules

and frames. Additionally, the functionality of the personal assistant is supported by an event model.

• The orientation towards cyber-physical systems in various application areas, including smart agriculture, is becoming more and more evident. All of the platforms discussed use different types of sensors, drones, robots, and satellites to gather information about the physical world.

• The ZEMELA platform is being developed as a virtual-physical-social space - a space because personal assistants, implemented as intelligent agents, operate as a central component, managing and controlling the overall operation of the platform. ZEMELA is an intelligent agriculture adaptation of the ViPS reference architecture.

Another important conclusion: for the successful implementation of the planned personal assistant, a thorough review and analysis of the existing state of the ZEMELA platform and the event model is required.

# CHAPTER TWO: EVENT MODEL

The ZEMELA platform is being developed as a virtual-physical space. A significant challenge is the virtualization of physical objects, which is a multifaceted process. In addition to the characteristics arising from the nature of the objects, it is also necessary to take into account their temporal, spatial and event (i.e. their involvement in various happenings) aspects. For this purpose, a theoretical basis is needed to act as a specific framework within which to implement the platform. We are looking for a simplified theoretical model that can realistically be the required framework. As a result of our preliminary studies and analyses, the thesis is formed that such a model can be proposed on the concept of "event". The present dissertation also continues research to prove this thesis. For this reason, the so-called *event model* was developed.

Events are the main mechanism for activating scenarios and actions, as well as for synchronizing the work of active components in the ZEMELA platform. An **event** is something that happens or we perceive to happen at a certain place and at a certain moment (or interval) of time and that affects the operation of the Platform. At ZEMELA, since events can occur in both the physical and virtual worlds, we aim to offer a common abstract (symbolic) representation of events. These representations are stored and managed in the virtual world (ie, events in the physical world are virtualized), but can cause effects in the physical world. Finding a suitable presentation is a challenge due to the wide variety of events with their specifics for the field of interest.

Most of the event models presented above offer complex structures for representing and modeling events. To some extent this is explainable – as was emphasized events have a multi-faceted nature. Our event model can be briefly characterized as follows:

- Simplified presentation of events;
- Hierarchy of events;
- Representation of complex events through recursive definitions;
- Proactivity.

The event model provides a common unified scheme, acting as a theoretical basis for the creation of the ZEMELA platform. The development of the platform is carried out at the following levels:

- The most abstract level is the event model.
- The second level is a representational level – representation of events as knowledge with a common meaning (rules, frames, ontologies).

- Programmatic and technical level – the program implementation of events and their processing.

Classifications play a very important role in our event model. Two main classifications are defined in the event model. The first uses a hierarchical approach where higher-level events are treated as predetermined by lower-level events. Thus, our event model distinguishes the following three levels of events:

- *Base events* – usually these are atomic events ie. events that we regard as indivisible. We represent these events as a pair {event_id : event_value}. The following base events are of particular interest in the model: time (date, hour, minute), location, (extreme) sensor values. This type of event occupies the lowest place in the hierarchy. Base events can initiate various higher-level events in the platform and as such can serve other events. Depending on the specifics of the supported domain, additional base events may be specified.
- *System events* – this type of events represent various events in the system infrastructure of the platform, such as the generation/removal of dynamic components and the sending/receiving of messages. System events are typically used to manage the interaction between the PA and other platform components. These events are located in the middle level of the hierarchy.
- *Domain-events* – they represent events typical of the application area of interest. These events usually have a complex structure. For each application domain, there is a wide variety of domain-events. In our model, these events play an extremely important role.

The second classification is related to how events unfold over time. In this sense, we distinguish two large groups of events, which we have named:

- *Scheduled* – these events are expected and can occur at a certain point (or interval) in time. In normal mode, PA operation is guided by this type of events.
- *Incidental* – these events occur randomly at a specific time or interval of time. Incident events are typically identified by the Event Engine.

There is a close relationship between the two classifications. To reveal it, we will take a closer look at the basic events of the model:

- Time – in general, time can be real or relative, represented as hours, days, months or years. This event is fundamental to working with the temporal aspects of the modeled physical objects, processes or

scenarios. These types of events are usually a component of scheduled domain events.

- Location – in general, this base event is used to represent (model) the spatial aspects of the physical objects, processes or scenarios of interest. It may possibly be a component of both types of domain-events.
- Extreme value – this type of base events serve to identify unexpectedly occurring domain events. As such, they are usually a component of incidental domain events.

Formally, an event is represented as e = <eid, etype, epar> with the following elements: eid - event identifier, etype - event type, epar - list of event attributes, which can be empty.

In the event model, different groups of operators on the domain-events are introduced. The use of appropriate operators reinforces the formalism with the ability to specify event expressions. On the one hand, event expressions can formally model complex and complex event situations. On the other hand, we can implement interpreters of these expressions that will facilitate platform management. The main operators are the following:

- ei ↑ ej – the occurrence of ei causes the occurrence of ej.
- ei ↓ ej – the occurrence of ei terminates the event ej.
- accu(e) – accumulation of events.

In order to be able to use the event model as the basis for the operational management of the platform, we require that both operators meet the following requirements:

- *Non-commutativity* – ei ↑ ej ≠ ej ↑ ei, i.e. if event ei causes event ej to occur, it should not follow that ej can cause ei to occur. Similarly for the operator ↓.
- *Left-associativeness for* ↑ – from the analysis so far it appears that the operator ↑ is left-associative, i.e. ei ↑ ek ↑ ej can be interpreted as (ei ↑ ek)↑ ej or event ei causes event ek to occur, which in turn causes event ej to occur.
- *Right-associativeness for* ↓ – similarly, it appears that the operator ↓ is right-associative, i.e. the expression ei ↓ ek ↓ ej must be interpreted as ei ↓ (ek ↓ ej) or event ek must first terminate event ej before it is terminated by event ei.
- *Transitivity* – domain-events are transitive.

In the last year and a half, despite efforts and various attempts at implementation, we do not have a proper means of identifying and locating events. It turns out that the identification of domain-events (especially

accidental ones) is a serious challenge. The reasons for this are of a different nature, such as a wide variety of domain-events, different sources for obtaining the values of the parameters characterizing one domain-event are obtained from different sources, the time factor.

Summarizing the current situation, the dissertation proposes the following solution: **to make identification part of the event model**. In the event model to propose a *reference abstract event engine* (Abstract Event Engine). The platform should support different implementations of the abstract event engine, depending on the nature of the particular event domain and the required data sources. As part of the event model, we look for a formal representation of the abstract Event Engine. In this point we will consider the possibility of using a *cellular automaton* to model the machine. A cellular automaton can be considered as an abstraction (idealization) of a physical phenomenon in which space and time are discretized, i.e. the set of states of the automaton is discrete and finite [12]. The individual components of such an automaton, called cells, are identical to some computing apparatus. Often, a one-dimensional, two-dimensional, or multidimensional grid is used to represent the automaton, with individual cells connected in a uniform way to each other. Cells that influence a particular cell are called the cell's neighborhood.

We will demonstrate the idea of modeling the behavior of the Abstract Event Engine (AEE) with an example. The significant incidents are crop diseases. One of the most widespread and dangerous tomato diseases is potato blight. The first signs of the disease are noticed on the lowest leaves that are in contact with the soil. Brown spots appear, which grow and the leaves dry up. Under favorable conditions, the disease spreads very quickly on petioles, stems and finally covers the whole plant. An irregularly shaped rusty brown spot is visible on the green fruits on the stem side. This disease can be modeled as an event domain occurring in time and represented as a cellular automaton in which each individual cell models a different part of the plant. In Fig. 1. an example cellular automaton for the disease is given, where we use the following notations:

- Model cells: C1 (lowest leaves), C2 (stem leaves), C3 (stems), C4 (green fruits), etc.
- t – time point;
- q – state of the cell at time t;
- x – input at time t;
- $\delta(q,x)$ – state of the cell at time t+1 (transition function);
- $y = \lambda(q,x)$, output at time t, $\lambda$ (output function).

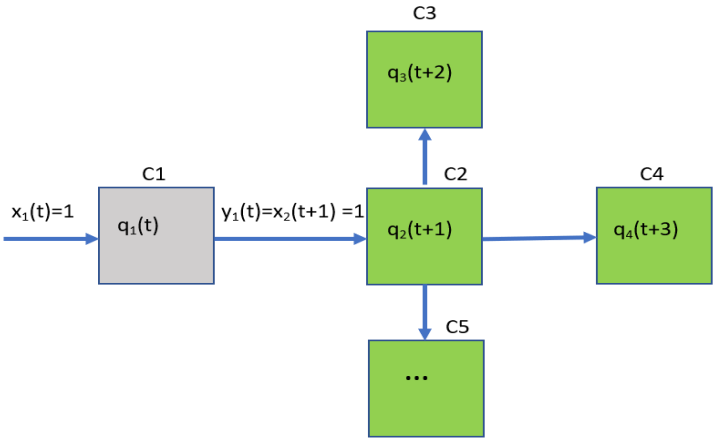


*Figure 1. Cellular automaton modeling potato blight disease in tomatoes*

Various rules can be defined for the cellular automaton to guide its operation, such as the following:

- When the appearance of brown spots is observed on the lowermost leaves that are in contact with the soil, then C1 "lives", meaning that t is the initial time, x1(t) = 1 and the internal state of the automaton becomes q1( t).
- The state of the previous cell (e.g. lowermost leaves), represented as output y1(t) = 1, is passed as input (the clock advances to the next step) to the neighboring cell (e.g. stem leaves) as y1(t) = x2(t+1) = 1 .

Thus, the proposed AEE model can be embedded in the PA belief.

# CHAPTER THREE: PLATFORM ZEMELA

The purpose of the platform is to support the development of applications for smart agriculture. ZEMELA is an adaptation of ViPS for a specific application area – in this case, smart agriculture. As such, it is constructed as a cyber-physical social space [13]. The ZEMELA platform is in the process of continuous development. Previous versions of the platform are presented in [14]. Based on the analysis carried out during the doctoral studies, here we will briefly present the current version of the platform. The goal is to offer a simplified architecture, a prototype of which will work in the conditions of the pilot project for the Plovdiv region. The proposed new version consisting of five basic components is given in Fig. 2.

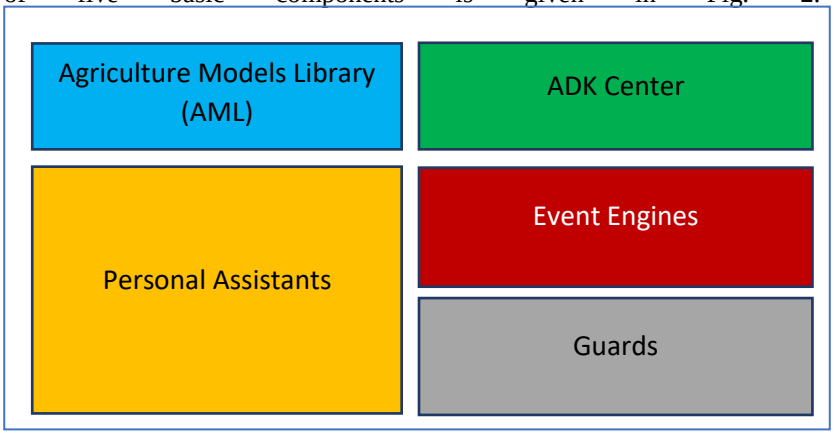


*Figure 2. New version of the LAND platform architecture*

*Personal Assistants.* They operate as the main operational mechanism ensuring the functioning of the platform. The main task of the PA is to monitor the normal course of the observed agricultural processes and scenarios. PAs are presented in detail in the next two chapters of the dissertation.

*Event Engines*. The responsibility of this type of modules, called Event Engines, is mainly to assist the PA in identifying mainly incident domain-events. The detection of such events is a basic mechanism for detecting anomalies in the vegetation of agricultural crops. The theoretical basis for implemented Event Engines is the event model presented in the previous chapter of the dissertation. In the new version of the platform, the concept of

event machines has been completely changed. Specifying a reference abstract engine as part of the model itself allows for different implementations and thus the existence of different Event Engines. This delivers much greater flexibility and variability for identifying anomalies – a core activity for the ZEMELA platform. Modules performing the role of event engines can be of different types, including those for which the recognition of incidental domain-events can be combined with other functions or obtained as a side effect - for example, the developed expert systems for early detection of diseases on agricultural crops.

*Agricultural Data and Knowledge Center (ADK Center).* This hub acts as a central distributed knowledge and data repository on the platform. It consists of two main parts. The first part is a distributed knowledge base in which theoretical knowledge about agriculture is stored and managed. Knowledge is stored about the so-called "natural artifacts" - these are agricultural crops and the factors affecting their vegetation such as soil, air and water. In addition, knowledge about "artificial artifacts" is stored - these are irrigation systems, sensor networks, ventilation systems, etc. The third type of stored knowledge is about farming activities and events occurring in farming scenarios and settings. Basically, knowledge is stored in mutually integrated ontologies, frames and rules. Some of the currently implemented major ontologies for agriculture are presented in [15]. Knowledge bases modeling agricultural knowledge in the form of rules and frameworks are described in [16]. The second part consists of relational databases and NoSQL to store various data about agricultural crops and the processes taking place in them. The data required to serve the platform can be viewed from different aspects. They can be structured, semi-structured and unstructured. A second aspect can be the frequency of changes – in this sense they can be static and dynamic. A third aspect is the nature of the represented objects – in this sense we can distinguish between natural objects and artificial objects. Considering this variety of factors, the development of an appropriate data model is essential. In the new version of the system, the concept of ADK Center has also been changed. In addition to the integration of the ViSMod module into the new repository structure, here we will present the initial idea of building a unified model of the data that will be stored in the ADK Center. Such a model was completely absent in the previous version of the ZEMELA platform. Conventionally, the model will have the distributed database in two planes - let's label them as "vertical" and "horizontal". The "vertical" plane will house the databases ("natural databases") storing data on natural objects and factors, such as agricultural crops, air, soil and water. In the horizontal plane, the so-called "artificial databases" will be built to store data on

artificial artifacts such as irrigation installations, sensor networks, ventilation installations, etc. A feature of our data model is that it should informationally provide a virtual-physical space and the related need for virtualization of physical objects [17]. For this purpose, it is additionally necessary to store and manage the spatial and temporal characteristics of physical objects.

*Agriculture Models Library*. In addition to agricultural knowledge and data, the platform supports a model library called the Agriculture Models Library (AML). Modeling different agricultural scenarios and processes would support PA work in an efficient manner. So, for example, when making a certain decision, the PA can take into account the results obtained from the processing of a certain model. Analyzing the results of the last two years (in the previous version of the platform, the use of different modeling approaches and systems was provided), in the new version of this library we propose that the leading modeling approach is based on the specification of the discrete event system DEVS. DEVS (Discrete Event Specification) is a general modeling and simulation formalism dedicated to dynamic discrete event system. The results of an experiment on the usability of this approach for modeling irrigation-related processes are presented in [18].

*Guard system (Guards).* One of the functions of the guards is to act as an interface between the virtual and physical worlds. They can support the exchange of information (and, if necessary, its transformation) between the two worlds. The physical world in the platform is perceived as a network of IoT nodes, which is able to collect up-to-date information about the "situation on the ground", i.e. sensor data from monitored open (e.g. blocks, plots) and closed agricultural areas (e.g. greenhouses).

After presenting the individual modules that make up the platform architecture, the second essential aspect is the interaction between them. For interaction between components at an abstract level, we will use the A&A (Agents & Artifacts) concept. Consistent with [19], the A&A model views agents' environments as a set of artifacts that agents can create, focus, and dispose in pursuit of their goals.

To implement the ZEMELA platform, we offer an integrated technology operating on JVM (Java Virtual Machine) (Fig. 3.).

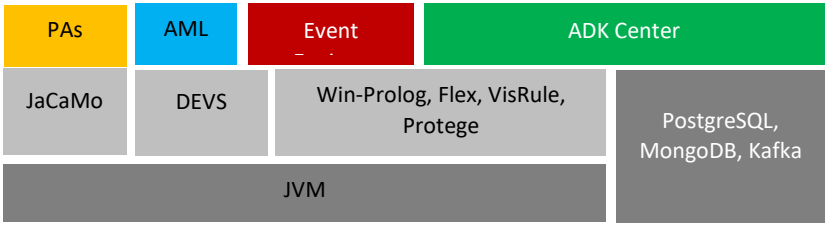


*Figure 3. Integrated technology*

To develop the models in AML, as well as for some variants of the event machines, we suggest using a DEVS-based approach.

The computing configuration where the platform can be deployed is built on three levels: local, regional and national. At the local level, sensor networks are usually used to collect information about the condition of agricultural land and plants. The regional level is a server configuration for collecting, storing and pre-processing data from dynamic sensors. A regional data center is implemented at this level. The national level will be developed to aggregate data and knowledge to produce statistics, analyzes and forecasts of national importance.

ZEMELA is a software platform implemented on top of the featured IoT three-tier vertical infrastructure that can be used to build smart agricultural applications

# CHAPTER FOUR: REFERENCE PERSONAL ASSISTANT

The personal assistant to help farmers presented in this chapter will be for a special purpose. PA uses the event model and ADK Center and is implemented as a rational practical reasoning agent with BDI architecture. The various services that PA can deliver to farmers are based on one of its fundamental abilities – to identify and locate various anomalies, mainly related to the vegetation of agricultural crops. Anomalies can take many forms and dimensions of manifestation, including as objects, numerical values, events, patterns, and observations.

In accordance with [20], the standard life cycle is presented in Fig. 4. The basic structure of the decision-making process is a loop in which the agent continuously observes the world and updates beliefs. It then deliberates to decide what intent (goal) to achieve (deliberation is done by first determining the available options and then filtering). It then uses a planning method to find a plan to achieve the accepted intention (goal). The last step is the execution of the plan.

```
// First step: Initialization

B ← consult(ADK_Center, plant); I ← beginning_vegetation(B) ;
vegetation(plant) ← true;

// Second step: Anomaly identification
while vegetation(plant) do
    getting next perception ρ via guard(…) function;
    B ← brf(B, ρ) ;
    if incidental_event_recognized(Event_Engine) then
        Die ← options(B, I) ;
        Iie ← filter(B, D, I) ;
        πie ← plan(B, I, Ac) ;
    end-if
    if incidental_event_recognized(Event_Engine) then
        Dpe ← options(B, I) ;
        Ipe ← filter(B, D, I) ;
        πpe ← plan(B, I, Ac) ;
    end-if
```

```
// Third step: Translate the selected execution plan
    execute(πie) or execute(πpe);
    vegetation(plant) ← false;
end-while
```

Figure 4. Life cycle of a standard PA for smart agriculture

The notations used in the PA life cycle pseudocode are: B (current belief), I, Iie, Ipe (current objectives), D, Die, Dpe (options available), $\pi_{ie}$, $\pi_{pe}$ (incident handling plans, respectively on planned events). In general, for the life cycle proposed here for our PA, we should note that it only models a single growing cycle for the plant selected for observation. The individual steps of the life cycle are presented as follows:

*Initialization* – this is a very essential stage in the life cycle of a PA (in the classical life cycle it is simply assumed that some initial belief $B_0$ exists). At this stage, the PA is adapted for the specific agricultural crop to be monitored. The PA must prepare its internal structures representing its faith. The necessary knowledge and data for this must be retrieved by the PA from the ADK Center by means of an appropriate interface, denoted in the pseudocode with *consult()*. Through this interface, the PA must select the information he needs for the specific agricultural crop *plant*. In this stage, the PA must also initialize its initial target – it will be a planning domain-event modeling the initial vegetation stage of the observed plant (*beginning_vegetation()*). It is also assumed that the vegetation cycle has started (*vegetation(plant)* ← *true*).

*Anomaly identification* – in this step, the work of the PA is presented to identify the events occurring during the course of one vegetation cycle of the observed plant (*vegetation(plant)* is a logical function that returns a value of true while the vegetation continues). It is important to note here that based on the event model, this life cycle step should be configured in two aspects - in terms of scheduled domain-events and respectively in terms of incidental domain-events.

*Translating the selected execution plan* - the behavior of the PA is twofold. One is behavior guided by plan events - in this case, the PA executes plans corresponding to such type of events (*execute($\pi_{pe}$)*). The other is behavior driven by incidental events - in this case the PA executes plans corresponding to such type of events (*execute($\pi_{ie}$)*).

Based on the life cycle presented in the previous point, here we will present the PA architecture (Fig.5.). The PA is a central component of the

platform that is responsible for the organization, management and control of the scenarios and processes supported by ZEMELA.

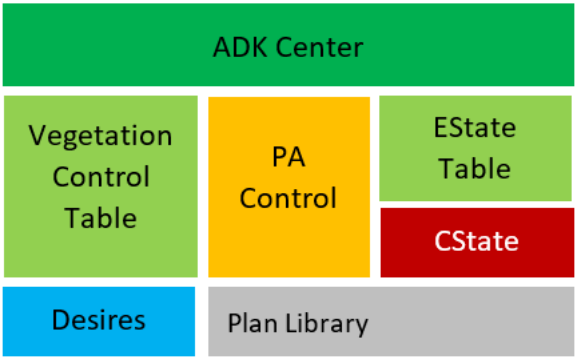


*Figure 5. PA architecture*

*Faith of the PA*. The personal assistant's belief (its perception of the environment) has a complex structure. As explained, PA uses the A&A concept, according to which all its elements are considered as artefacts. In addition to the standard components of the ZEMELA platform (ADK Center, AML, Event Engines, Guards), three important structures supporting the operational work of the PA are included as part of the belief. The underlying control structure is the *Vegetation Control Table (VCT)*, which is generated when the assistant is initialized. This structure fulfills the role of a kind of schedule of the process of the expected vegetation of the observed agricultural crop. The individual records model the individual phases of the vegetation in the form of planning domain-events. Based on this structure, the PA orients itself to the current vegetation phase of the plant. The second structure called the *EState Table* stores records of the expected state of the plant for each event. The entries in this table correspond to the entries from the VCT. The third structure known as *CState* represents the actual state of the plant for the event that occurred. This structure is different from the other two. It stores only the current real state of the observed plant and changes dynamically as the vegetation progresses over time. When adapting for a certain agricultural crop, the reference PA initializes the VCT and the EState Table. The PA extracts the necessary information from the knowledge and data stored in the ADK Center. Once initialized, these two structures remain static (they do not change their contents). EState is updated depending on data measured by sensors and/or observed by farmers. This is done by the

platform guards. Working with the three structures, the PA was able to detect anomalies in the vegetation of the observed agricultural crops. For the current phase of the vegetation, it compares the expected with the actual state and, in case of detected deviations, makes a conclusion about the presence of anomalies.

*PA options*. Options (intentions) model the PA's abilities to handle various situations arising in the course of his work. Typical options can be *trace(VCT), inform, warn, prevent, react to anomaly*.

*PA goals*. One ongoing goal of PA is to track VCT. When an anomaly is detected, the PA updates the persistent target with one that ensures it is triggered to select a plan to fix it. Depending on the specific situation, the PA chooses from the available options an actual goal to achieve, with which he commits (the goal is a selected and committed option). For example, if the PA detects a difference between an entry in the CState Table and an EState, then it can generate an *anomaly* or *warning response goal* - otherwise, it can generate *an inform goal*.

*PA Control*. This component implements the lifecycle driving the two PA behaviors. After specifying the current objective, the PA should prepare an appropriate action plan. In the normal course of the vegetation, it can simply inform the farmer about the values of the desired parameters. When an anomaly is detected, there are various possibilities. The PA maintains its own internal library of plans from which it can choose a completely ready-made or adapt a possible plan. Another option is for the PA to activate components from the platform, e.g. AML (Agent Communication Language) to implement a particular model.

# CHAPTER FIVE: SOFTWARE IMPLEMENTATION OF A PERSONAL ASSISTANT PROTOTYPE

JaCaMo [21] is a multi-agent programming framework that combines three distinct technologies, each of which can be used independently. The framework has been in development for years, and the current version is stabilized and full-featured. JaCaMo is a combination of the following three technologies: Jason, Cartago and Moise.

The current version of the prototype, adapted to detect anomalies in tomato vegetation, is presented in this point. The data used are from the region of Plovdiv. When the Personal Assistant (PA) is started, its wishes or goals are represented in JaCaMo as follows:

```
/* Initial goals */

!communicate_farmer.
!monitor_plant.
!search_anomalies.
```

The objectives are to monitor the development of the plant, to look for anomalies in its vegetation and to communicate with the farmer in a timely manner, to inform him of the results of his work, and accordingly to receive and fulfill new farming wishes. To implement communication with the farmer, the PA creates an artifact named ArtName, of type gui.UserGUI, with an empty list of parameters and ID ArtID, and then focuses on it:

```
+!communicate_farmer
    <- makeArtifact(ArtName,"gui.UserGUI",[],ArtID);
       focus(ArtID).
```

Thus, performing “actions” on this artifact, they will be reflected on the farmer's interface, and accordingly, focusing on the artifact, I will receive the received signals from the farmer. The plant vegetation monitoring objective has an achievement plan including two sub-objectives:

```
+!monitor_plant
    <- !!track_heat_units;
       !track_tomato_phases.
```

namely, monitoring the heat units and, accordingly, based on them, the assumed phase of the tomato's vegetation. The concept of heat units or degree days (DD) was first introduced by Daniel Putnam in 1868, an American agricultural scientist who was working on ways to improve crop yields. He noted that crop growth and development were closely related to temperature and proposed the idea of using accumulated heat (measured in degree days) as a way of predicting the timing of different stages of growth. Putnam's concept of degree days has since been refined and extended by other scientists and is now widely used in agriculture to track the growth and development of a wide variety of crops, including tomatoes. DDs measure the amount of heat accumulated over time. They can be calculated in many different ways. The simplest way to calculate DD accumulation is the simple average method:

```
DD = (Tmin + Tmax)/2 - Tbase
Tmin = минимална дневна температура
Tmax = максимална дневна температура
```

Tbase = долен праг на развитие

Base for tomatoes is 10 °C. The specific degree days required for different growth stages of tomato crops can vary depending on variety, planting date and weather conditions. However, in general, the following approximate values are often used as a guide for tomato crops in a climate like that of the city of Plovdiv:

Germination (Покълване): 60 DD (degree days)
Seedling stage (Фаза на разсаждане): 175 DD
Vegetative growth (Вегетативен растеж): 275 DD
Flowering (Цъфтеж): 700 DD
Fruit set (Формиране на плод): 900 DD
Maturity (Зрялост): 1500 DD

Below are the plans in Jason for calculating heat units:

```
+!track_heat_units
   <- .time(H,M,S) ; H=0 ; ?temperature(X0) ; X0=Max ; X0=Min ; .wait(3600000) ; ?temperature(X1) ;
  if (X1>X0) { X1=Max; } elif (X1<X0) { X1=Min; } else { true; } .wait(3600000) ; ?temperature(X2) ;
  if (X2>Max) { X2=Max; } elif (X2<Min) { X2=Min; } else { true; } .wait(3600000) ; ?temperature(X3) ;
  if (X3>Max) { X3=Max; } elif (X3<Min) { X3=Min; } else { true; } .wait(3600000) ; ?temperature(X4) ;
  if (X4>Max) { X4=Max; } elif (X4<Min) { X4=Min; } else { true; } .wait(3600000) ; ?temperature(X5) ;
  if (X5>Max) { X5=Max; } elif (X5<Min) { X5=Min; } else { true; } .wait(3600000) ; ?temperature(X6) ;
  if (X6>Max) { X6=Max; } elif (X6<Min) { X6=Min; } else { true; } .wait(3600000) ; ?temperature(X7) ;
  if (X7>Max) { X7=Max; } elif (X7<Min) { X7=Min; } else { true; } .wait(3600000) ; ?temperature(X8) ;
  if (X8>Max) { X8=Max; } elif (X8<Min) { X8=Min; } else { true; } .wait(3600000) ; ?temperature(X9) ;
  if (X9>Max) { X9=Max; } elif (X9<Min) { X9=Min; } else { true; } .wait(3600000) ; ?temperature(X10);
  if (X10>Max) { X10=Max; } elif (X10<Min) { X10=Min; } else { true; } .wait(3600000) ; ?temperature(X11) ;
  if (X11>Max) { X11=Max; } elif (X11<Min) { X11=Min; } else { true; } .wait(3600000) ; ?temperature(X12) ;
  if (X12>Max) { X12=Max; } elif (X12<Min) { X12=Min; } else { true; } .wait(3600000) ; ?temperature(X13) ;
  if (X13>Max) { X13=Max; } elif (X13<Min) { X13=Min; } else { true; } .wait(3600000) ; ?temperature(X14) ;
  if (X14>Max) { X14=Max; } elif (X14<Min) { X14=Min; } else { true; } .wait(3600000) ; ?temperature(X15) ;
  if (X15>Max) { X15=Max; } elif (X15<Min) { X15=Min; } else { true; } .wait(3600000) ; ?temperature(X16) ;
  if (X16>Max) { X16=Max; } elif (X16<Min) { X16=Min; } else { true; } .wait(3600000) ; ?temperature(X17) ;
  if (X17>Max) { X17=Max; } elif (X17<Min) { X17=Min; } else { true; } .wait(3600000) ; ?temperature(X18) ;
  if (X18>Max) { X18=Max; } elif (X18<Min) { X18=Min; } else { true; } .wait(3600000) ; ?temperature(X19) ;
  if (X19>Max) { X19=Max; } elif (X19<Min) { X19=Min; } else { true; } .wait(3600000) ; ?temperature(X20) ;
  if (X20>Max) { X20=Max; } elif (X20<Min) { X20=Min; } else { true; } .wait(3600000) ; ?temperature(X21) ;
  if (X21>Max) { X21=Max; } elif (X21<Min) { X21=Min; } else { true; } .wait(3600000) ; ?temperature(X22) ;
  if (X22>Max) { X22=Max; } elif (X22<Min) { X22=Min; } else { true; } .wait(3600000) ; ?temperature(X23) ;
  if (X23>Max) { X23=Max; } elif (X23<Min) { X23=Min; } else { true; }
  if (Min<10) { Min=10; } else { true; }
  if (Max>35) { Max=35; } else { true; }
  (Max+Min)/2-10=D ; -+heat_units(D) ; !track_heat_units.

-!track_heat_units
   <- .wait(3600000) ; !track_heat_units.
```

The agent checks the temperature every hour and at the end of the day determines the minimum and maximum temperature for the day, and accordingly calculates “degree days” according to the formula for that day and updates the accumulated heat units since the beginning of its work.

When the required heat units are reached and the plant is assumed to enter the next phase, the PA notifies the farmer with a message that a given phase

is assumed to be reached:

```
+!track_tomato_phases
   <- ?heat_units(D) ;
   if (D>=0 & D<60) {  .wait(3600000); !track_tomato_phases; }
   elif (D>=60 & D<175) { println("It is assumed that the 'Germination' phase has been reached!"); }
   elif (D>=175 & D<275) { println("It is assumed that the 'Seedling' stage phase has been reached!"); }
   elif (D>=275 & D<700) { println("It is assumed that the 'Vegetative growth' phase has been reached!"); }
   elif (D>=700 & D<900) { println("It is assumed that the 'Flowering' phase has been reached!"); }
   elif (D>=900 & D<1500) { println("It is assumed that the 'Fruit set' phase has been reached!"); }
   elif (D>=1500) { println("It is assumed that the 'Maturity' phase has been reached!"); }
   .drop_intention(track_tomato_phases);
   .wait(86400000); !track_tomato_phases.
```

The personal assistant uses the created GUI artifact and through an "external action" (in this case "println"), prints the message on the farm interface.
In addition to the accumulation of heat units, the Personal Assistant also monitors extreme temperature values where the vegetation process is in danger of being irreversibly interrupted:

```
+temperature(T) : heat_units(D) & D>=60 & D<175 & T<10 | T>35
   <- println("Attention, critical temperature in the 'Germination' phase!").

+temperature(T) : heat_units(D) & D>=175 & D<275 & T<10 | T>29
   <- println("Attention, critical temperature in the 'Seedling' phase!").

+temperature(T) : heat_units(D) & D>=275 & D<700 & T<10 | T>35
   <- println("Attention, critical temperature in the 'Vegetative growth' phase!").

+temperature(T) : heat_units(D) & D>=700 & D<900 & T<13 | T>32
   <- println("Attention, critical temperature in the 'Flowering' phase!").

+temperature(T) : heat_units(D) & D>=900 & D<1500 & T<13 | T>32
   <- println("Attention, critical temperature in the 'Fruit set' phase!").

+temperature(T) : heat_units(D) & D>=1500 & T<10 | T>35
   <- println("Attention, critical temperature in the 'Maturity' phase!").
```

The minimum and maximum critical temperature varies in different phases, the minimum for tomatoes is usually around their base or about 10 degrees Celsius, while the maximum reaches up to 32-35 degrees. In the case of exceeding these degrees or falling below the minimum critical, the Personal Assistant warns the farmer again through the GUI artifact. We have two sub-goals to fulfill the "anomaly search" goal:

```
!search_anomalies
   <- !!init_VCT;
      !init_EState;
```

On startup, the agent aims to initialize the VCT (Vegetation Control Table) and the EState (Estimated State Table), or the Vegetation Control Table and the Estimated State Table.

The VCT initialization plan is:

```
+!init_VCT
    <- !create_VCT;
       getInstances(tomato_stages);
       for (.member(M, instances)) {rules.add_rule}.
```

The first formula to execute in the body of the plan is to create a VCT artifact. Then we have an implementation of ``action" (external), which represents a method from the java artifact code of the same name:

```
public class VCT extends Artifact {

    private OWLOntology ontology;

    void init(String ontologyPath) {
        // Load ontology from file
        try {
            OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
            ontology = manager.loadOntologyFromOntologyDocument(getArtifactURI().resolve(ontologyPath).toURL().openStream());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @OPERATION
    List<String> getInstances(String concept) {
        List<String> instances = new ArrayList<String>();
        OWLClass cls = ontology.getOWLOntologyManager().getOWLDataFactory().getOWLClass(IRI.create(concept));
        NodeSet<OWLNamedIndividual> individuals = reasoner.getInstances(cls, false);
        for (OWLNamedIndividual ind : individuals.getFlattened()) {
            instances.add(ind.getIRI().toString());
        }
        return instances;
    }
}
```

The ``init" method loads an ontology from a file with its given path. The ``getInstances" method takes a ``concept" name as input and returns a list of instances that belong to that concept. It does this by first retrieving the corresponding OWLClass object from the ontology using its IRI, and then using an OWLReasoner object to obtain the NodeSet of the individuals that belong to that class. Finally, it adds the IRIs of the faces to a list and returns it.

The last formula in the body of the plan loops through each element of the list and applies an ``internal action" to it, which in this case records as a rule in the agent's belief base each phase of the tomato vegetation, such as the belief that we are in this phase is available when the right-hand side of the rule is true, or when the required values of heat units taken as ontology data

are reached. Below is a typical resulting vegetation control table in the agent's belief base:

```
/* Initial beliefs and rules */

temperature(X).
heat_units(D).

germination :- D>=60 & D<175.
seedling :- D>=175 & D<275.
vegetative_growth :- D>=275 & D<700.
flowering :- D>=700 & D<900.
fruit_set :- D>=900 & D<1500.
maturity :- D>=1500.
```

Accordingly, this VCT table serves as a schedule for the agent. Each time a new phase is reached, these domain events trigger the execution of plans that I have shown below.

Like the Vegetation Control Table, we handle the purpose of initializing the Estate Estimated Table in the same way:

```
+!init_EState
    <- !create_EState;
       getInstances(stages_attributes);
       for (.member(M, instances)) {+M}.
```

We create an EState artifact, using the ``getInstances'' method we extract from the ontology the attributes of each vegetation phase and traverse the list, adding each element of it as a belief in the agent's belief base. In the case of tomatoes, such a belief would look like this:

```
phase(height,stem_diameter,leaves_number,fruit_presence,fruit_color).
```

Each phase has five definitions that describe the state of the plant in it. For tomatoes, these are plant height, stem diameter, number of leaves, fruit presence and fruit color.

In the sprouting stage, the tomato plant has not yet emerged from the soil, so it does not have any height. During germination, the seed absorbs water and nutrients from the soil, and the embryo in the seed begins to grow and develop. Once the stem emerges from the soil, the transplanting stage begins and the plant begins to grow in height. Accordingly, the first two parameters (height and stem diameter) are missing in the germination phase.

The number of leaves in this phase is 1-2 cotyledons (seed leaves), and the parameters fruit presence and color are empty.

In the seedling phase, the height of the plant is about 10-15 centimeters, the diameter of the stem between 1 and 3 millimeters, the number of leaves from 2 to 4, and empty parameters for the presence and color of the fruits.

In the phase of vegetative growth, the height of the plant reaches 60-90 centimeters, the diameter of the stem between 3 and 5 millimeters, the number of leaves from 5 to 7, and empty parameters for the presence and color of the fruits.

In the flowering phase, the height of the plant reaches 90-120 centimeters, the diameter of the stem between 8 and 12 millimeters, the number of leaves from 10 to 15, and empty parameters for the presence and color of the fruits.

In the phase of fruit formation, the height of the plant is between 90-120 centimeters, the diameter of the stem between 15 and 20 millimeters (or even more), the number of leaves from 10 to 15, and fruits of green color are available.

In the mature phase, the height of the plant is between 90-120 centimeters, the diameter of the stem between 25 and 50 millimeters, the number of leaves from 5 to 10, and fruits of red, yellow, orange or even purple color are available.

Current State (CState) represents the current state of the plant. This structure differs from VCT and EState because it stores only the current real state of the plant and changes dynamically as the vegetation progresses over time. Incoming data from the sensors and processed by the guards, the CState is updated in the agent's belief base like the EState, but with dynamic values.

```
cstate(height,stem_diameter,leaves_number,fruit_presence,fruit_color).
```

In order to detect anomalies, the Personal Assistant, guided by the schedule, at any moment compares the expected state for the phase in which it is

currently located with the dynamic data for the current state of the plant.

```
+germination
   <- .findall(X,germination(X),E);
      .findall(Y,cstate(Y),C);
      compare.lists(E,C);
      println(result).

+seedling
   <- .findall(X,seedling(X),E);
      .findall(Y,cstate(Y),C);
      compare.lists(E,C);
      println(result).

+vegetative_growth
   <- .findall(X,vegetative_growth(X),E);
      .findall(Y,cstate(Y),C);
      compare.lists(E,C);
      println(result).

+flowering
   <- .findall(X,flowering(X),E);
      .findall(Y,cstate(Y),C);
      compare.lists(E,C);
      println(result).

+fruit_set
   <- .findall(X,fruit_set(X),E);
      .findall(Y,cstate(Y),C);
      compare.lists(E,C);
      println(result).

+maturity
   <- .findall(X,maturity(X),E);
      .findall(Y,cstate(Y),C);
      compare.lists(E,C);
      println(result).
```

For each of the phases, it extracts the parameters from the beliefs and puts them into a list. It does the same with the parameters from the belief about the current state of the plant, compares the two lists via an "internal action" written in java, and prints the results on the farm interface.

# CONCLUSION

The results of the research conducted within the dissertation can be summarized as follows.

*Event model*. A new version of the event model has been introduced. It retains the main productions presented in the previous version. The new stuff is about refining the definitions of the basic event operations in terms of their algebraic properties. Their non-commutativity, as well as their left and right associativity, respectively, are motivated. Distribution remains undetermined for the time being. A completely new concept of an abstract event machine has been introduced, now as part of the event model. It is proposed to formalize it as a cellular automaton. Further tasks related to the practical implementation of the abstract machine are forthcoming.

*The ZEMELA platform*. A new version of the ZEMELA platform architecture is presented. In the new version, a new structure of the specialized knowledge and data repository ADK Center is proposed, integrating the ViSCoD component of the previous version. Simplified structure of the specialized library for AML models. In the new version, it is proposed to use mainly the DEVS approach. Analysis of the results obtained so far shows that it can be effective for the purposes of the platform.

*Personal assistant*. A reference architecture of a personal assistant designed to support farmers working in the context of smart agriculture has been developed. The personal assistant is specialized in purpose, based on a specific event model and using a repository of specialized knowledge and data. In the version presented in the thesis, the personal assistant is adapted for smart agriculture. A prototype implementation of the assistant implemented using the JaCaMo development framework is also presented.

There are various opportunities for future developments and projects related to the dissertation topic, such as expanding PA functionality, machine learning in PA, interaction of PA with technological chains, "pulling down" the intelligence of the platform, reengineering the platform architecture and moving to a "pure" agent-oriented version, deploying PA on different devices.

LIST OF PUBLICATIONS CITED IN THE DISSERTATION

**OF IVAN STANIMIROV STOYANOV**,

PHD STUDENT AT THE DEPARTMENT OF COMPUTER SYSTEMS


1. S. Stoyanov, J.Todorov, I. Stoyanov, V. Tabakova-Komsalova, L. Dukovska, ZEMELA – An Intelligent Agriculture Platform, Big Data, Knowledge and Control Systems Engineering – BdKCSE'2021, 28–29 October 2021, Sofia, Bulgaria.
2. S. Stoyanov, I. Stoyanov, V. Tabakova-Komsalova, A. Dukovski, L. Doukovska, An Event-Based Platform Supporting Smart Agriculture Applications. 2022 IEEE 11th International Conference on Intelligent Systems (IS), Warsaw, Poland, 2022, pp. 1-5, doi: 10.1109/IS57118.2022.10019674.

# ACKNOWLEDGMENTS

I would like to express my gratitude to the individuals without whom this dissertation work would not have been possible.

I owe a huge debt of gratitude to my scientific supervisor, Prof. Dr. Asya Stoyanova-Doycheva, for trusting me and dedication in every situation, ideas and advice, which I appreciate extremely much.

I express special thanks to Art. cor. prof. d.n. Lyubka Dukovska and ch. assistant professor Dr. Veneta Tabakova-Komsalova for her positive energy and warm attitude, valuable guidance and unreserved help.

I thank the entire team implementing the National Scientific Program "Intelligent Agriculture" for the help and friendly creative environment in which the scientific research presented in this dissertation was carried out.

REFERENCED LITERATURE


[1] Национална програма „Интелигентно растениевъдство 2021-2023“, 2020.

[2] W. Wolf. Cyber-physical systems. Computer, 42(3):88-89, March 2009.

[3] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A survey. Comput. Netw., 54:2787-2805, Oct. 2010.

[4] S. Stoyanov, T. Glushkova, E. Doychev, A. Stoyanova-Doycheva, V. Ivanova, Cyber-Physical-Social Systems and Applications. Part I: Reference Architecture,, LAP LAMBERT Academic Publishing, 2019.

[5] Е. Дойчев, Среда за електронни образователни услуги, дисертация, Пловдивски университет „Паисий Хилендарски“, 2013.

[6] S.Stoyanov, V. Valkanov, I. Popchev, A. Stoyanova-Doycheva, E. Doychev, A Model of Context-Aware Agent Architecture, Compt. Rend. Acad. Bulg. Sci., Tome 67, № 4, 2014, pp. 487-496.

[7] S. Stoyanov, Context-Aware and Adaptable eLearning Systems, PhD Thesis, STRL, De Montfort University, Leicester, UK, 2012.

[8] FaST Platform, https://fastplatform.eu/.

[9] Farm21, https://www.farm21.com/farming-platform-to-farm-smarter/

[10] FAO, https://www.fao.org/platforms/common-practices/en.

[11] GPT-4, https://www.bbc.com/news/technology-64959346.

[12] B. P. Zeigler, A. Muzy, E. Kofman, Modeling and Simulation. Discrete Event and Iterative System. Computational Foundations, Elsevier, Academic Press, 2019.

[13] S. Stoyanov, J. Todorov, I. Stoyanov, V. Tabakova-Komsalova, L. Dukovska An Event-Based Platform Supporting Smart Agriculture Applications, ZEMELA – An Intelligent Agriculture Platform (BdKCSE2021).

[14] S. Stoyanov, I. Stoyanov, V. Tabakova-Komsalova, A. Dukovski, L. Doukovska, An Event-Based Platform Supporting Smart Agriculture Applications. 2022 IEEE 11th International Conference on Intelligent Systems (IS), Warsaw, Poland, 2022, pp. 1-5, doi: 10.1109/IS57118.2022.10019674.

[15] A. Stoyanova-Doycheva, E. Doychev, V. Ivanova, V. Valkanov, V. Tabakova-Komsalova, Event Ontology about Wheat Cultivation, AgriControl 2022The 7th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture, 14-16 September 2022, Munich, Germany.

[16] S. Stoyanov, V. Tabakova-Komsalova, L. Doukovska, I. Stoyanov & A. Dukovski, An Event-Based Platform Supporting Smart Agriculture

Applications, IEEE Intelligent Systems IS'22, Warsaw, Poland, October 12-14, 2022.
[17] К. Граматова, Изграждане на виртуално образователно пространство като екосистема в Интернет на нещата, дисертация, Пловдивски университет „Паисий Хилендарски", 2017.
[18] P. Yochkova, V. Tabakova-Komsalova, S. Cherecharov, L. Doukovska, S. Stoyanov, DEVS Modeling of an Irrigation System, IEEE Intelligent Systems IS'22, Warsaw, Poland, October 12-14, 2022.
[19] Ricci A, Piunti M, Viroli M, Omicini A. 2009. Environment programming in CArtAgO. In: El Fallah Seghrouchni A, Dix J, Dastani M, Bordini R, eds. Multi-agent programming. Boston: Springer, 259–288 DOI 10.1007/978-0-387-89299-3_8.
[20] Иван Стоянов, Станимир Стоянов, Ирина Кръстева, Персонален туристически екскурзовод ‚Ловеч и региона', Международна научна конференция „Tech-Co" 2020, 17-18 юли, Ловеч (https://www.tugab.bg/images/tk-lovech/konferencia/Techko-Lovech-2020-w.pdf).
[21] Boissier, O., Bordini, R., Hübner, J., Ricci, A. 2020. Multi-Agent Oriented Programming Programming. Multi-Agent Systems Using JaCaMo, The MIT Press Cambridge, Massachusetts London, England.