Plovdiv University "Paisii Hilendarski"
Faculty of Mathematics and Informatics
Department of Computer Systems

MODELLING OF AN INTELLIGENT SUPPLY CHAIN IN SMART AGRICULTURE SYSTEMS

ABSTRACT
of a dissertation for the award of an educational and scientific degree "Doctor" in the field of higher education 4. Natural Sciences, Mathematics and Informatics; professional direction 4.6. Informatics and Computer Sciences;
Doctoral programme in Informatics

Part-time doctoral student: Laska Delkova Kostadinova-Tsankova
Scientific supervisor: Assoc. Prof. Emil Doichev

Plovdiv, 2026

The dissertation was discussed and referred for defence at an extended meeting of the Department Council of the Department of Computer Systems at the Faculty of Mathematics and Informatics at Plovdiv University "Paisii Hilendarski" held on 13 March 2026.

The dissertation "MODELLING OF AN INTELLIGENT SUPPLY CHAIN IN INTELLIGENT AGRICULTURE SYSTEMS" contains 212 pages, 64 figures, 19 tables and 234 references, which include scientific literature in the field of informatics, computer science and economics. The list of the author's publications consists of 10 titles, 2 of which are cited in the dissertation.

The defence of the dissertation will take place on 22 May 2026 at ……. in the Conference Hall of the new building of Plovdiv University "Paisii Hilendarski", Plovdiv, 236 Bulgaria Blvd. The materials related to the defence are available to interested parties at the Dean's Office of the Faculty of Mathematics and Informatics, new building of Plovdiv University, room 330, every working day from 8:30 a.m.

The materials related to the defence are available to interested parties at the Dean's Office of the Faculty of Mathematics and Informatics, new building of Plovdiv University, room 330, every working day from 8:30 a.m. to 5:00 p.m.

CONTENTS

INTRODUCTION

This dissertation examines the research, modelling and creation of an effective supply chain management system for a real company performing public procurement in Plovdiv. Its main objective is to develop an innovative model and architecture for optimising logistics processes and risk management in the supply of strategic public institutions – hospitals, schools, kindergartens and social services. Today's dynamic and interconnected economic environment increases the complexity and uncertainty in supply chains. Risk management is becoming a key factor for sustainability, reliability and adaptability. This is particularly important in public procurement, where the requirements for transparency, timeliness and accountability are high. In response to these challenges, the dissertation proposes a conceptual model, architectural framework and prototype of an integrated supply chain management system, developed as an upgrade to the ZEMELA platform. The aim is to expand its functionalities and adapt it to the specific needs of public institutions. The proposed solution is aimed at optimising supply and proactively managing internal and external risks through the use of artificial intelligence, machine learning, scenario planning and simulation modelling.

The relevance of the research stems from the growing competition and conflict between participants in the supply chain, which requires the application of an integrated and scientifically sound approach. The thesis offers practical and adaptive solutions tailored to the specifics of the public sector in Bulgaria and based on internationally established good practices.

The subject of this dissertation research is risk management in supply chains, with a particular focus on the contradictions and challenges arising from the regulatory and procedural framework of public procurement.

The subject of study in this dissertation is the supply chain in its entirety, viewed as a set of processes for planning, organizing, managing, and controlling the flow of goods and services from the supplier to the end recipient.

The main objective of this dissertation is to develop a model of an intelligent supply chain. A prototype implementation of the model will be integrated into the existing ZEMELA platform, integrating the platform with the logistics system of a food supplier that delivers to childcare facilities in the municipality of Plovdiv. Within the scope of this objective, it is envisaged to build an event model for tracking, analysing and optimising key points in the supply chain – from the emergence of needs, through public procurement procedures, to the final execution and acceptance of deliveries by childcare facilities. The model will enable the logistics system to adapt to events with potential risk and will facilitate better coordination between all stakeholders.

To achieve the set goal, **the** following specific **research tasks** are formulated:

- Analysis of modern supply chains and their integration into smart farming platforms.
- Development of a model and architecture for the system.
- Prototype implementation.

To achieve the set goal and in accordance with the formulated tasks, this dissertation proposes **a methodology for conducting the research**, which is based on a systematic and step-by-step approach to the creation of an intelligent supply chain management system.

CHAPTER ONE: SUPPLYS CHAIN

In the context of global economic and technological connectivity, supply chains are a key element of modern manufacturing and trade. They encompass the flows of materials, information and finance from the source to the end customer and have become a strategic factor for competitiveness. Today, organisations face challenges such as pandemics, geopolitical instability, inflation and environmental requirements. This chapter of the dissertation examines various definitions of "supply chain", basic models, risks and disruptions, as well as examples from real companies. Basic concepts for smart supply chains and their logistics are also presented. The main concepts are:

The essence of the supply chain. The supply chain is a network of organisations and processes that ensure the movement and transformation of products from raw

materials to the end user, including reverse logistics. Efficiency depends on coordination, synchronisation and adaptability.

Key elements of supply chain management (SCM). SCM combines planning, procurement, production, delivery and reverse logistics. ERP and other digital systems provide transparency and real-time control through IoT, blockchain and analytical tools.

The basic concepts of smart supply chains are:

Direct-to-Consumer (DTC). The Direct-to-Consumer (DTC) model simplifies the distribution channel by eliminating intermediaries and establishing a direct link between the manufacturer and the end customer. This ensures complete product traceability – from origin and production to logistics [1]. DTC can be implemented through direct partnerships with end consumers or institutions (schools, kindergartens, hospitals), through farmers' markets and clear labelling of product origin [2], [3]. In this model, the producer takes control of logistics activities such as warehousing, forecasting, order fulfilment and returns management [4].

On-Demand Supply. The on-demand model is triggered by a specific customer request and focuses on speed of execution. However, it often does not take into account sustainability, optimal routes or quality parameters of delivery, which can lead to higher costs, longer delivery times and an increased environmental footprint [5].

The logistics of a food warehouse connecting producers with nurseries requires coordinated management of supply, storage and distribution to ensure the delivery of safe and fresh food for children.

Supply and inbound logistics. The warehouse works with local farmers (seasonal and fresh produce, reduced "food miles"), processed food manufacturers and specialist suppliers for specific diets. Key activities include demand forecasting, supplier management and strict quality control. Transport includes cold chain maintenance, synchronised delivery schedules and acceptance checks.

Storage and inventory management. The warehouse uses temperature-controlled zones, the FIFO principle and real-time inventory tracking systems. Batch and expiry date management is particularly important due to the service provided to vulnerable groups. The goal is to minimise spoilage and prevent shortages or excess inventory.

Order processing and outbound logistics. Orders are picked by staff or automated systems, packaged to preserve quality, and clearly labelled. Deliveries are optimised by route and time windows, maintaining a cold chain to the end recipient where necessary. Processes are in place for managing returns and sustainable waste treatment.

Information systems and control. ERP systems support the coordination of orders, stocks and transport, while GPS and IoT provide real-time monitoring. Effective communication with nurseries ensures timely deliveries and transparency.

Compliance and sustainability. Processes are based on HACCP and national food safety requirements, including REGULATION No. 14 OF 9 DECEMBER 2021 ON FOOD HYGIENE, as well as traceability principles. Sustainability includes local sourcing, eco-friendly packaging and the use of electric or hybrid vehicles.

The food warehouse is a key link between producers and nurseries. The effectiveness of the system depends on integrated quality management, traceability, digital technologies, regulatory compliance and sustainable practices that ensure a safe and timely food supply.

In conclusion, supply chains are undergoing a period of significant transformation. Organisations that manage to adapt to the new conditions by implementing modern technologies, focusing on sustainability and building flexible systems will gain a significant competitive advantage [6]. For Bulgarian companies, this is both a challenge and an opportunity for modernisation and better integration into global supply chains. In the future, the development of these chains will depend on the ability of organisations to balance efficiency and sustainability, globalisation and localisation, as well as automation and the human factor [7]. Companies that achieve this balance and build flexible and sustainable systems will be the most successful in the new era of global trade.

CHAPTER TWO: CONCEPTUAL FRAMEWORK

This chapter presents an analysis of the efficiency of a food supplier in Plovdiv, serving municipal and private childcare facilities in accordance with the Public Procurement Act. The approach is interdisciplinary – logistics, public administration, quality management and economics – with the aim of assessing actual performance and formulating improvements. The approach is interdisciplinary – logistics, public administration, quality management and economics – with the aim of assessing actual performance and formulating improvements.

Research methodology. A mixed method is applied:

- *Quantitative analysis* – number of deliveries, delays, volumes, costs, survey results.
- *Qualitative analysis* – interviews, process observation, assessment of communication and organisational culture.

This allows for an assessment of both the administrative and legal framework and the operational logistics.

The methods used for data collection are:

- Document analysis – buyer profiles, technical specifications under the Public Procurement Act, contracts, internal documents, HACCP plan.

- Semi-structured interviews – 7 key participants (management, logistics, municipality, directors).
- Survey – 15 employees from childcare facilities and 5 from the supplier; assessment of deadlines, quality, communication and transparency.
- Observation – monitoring of the actual logistics process (order–warehouse–transport–delivery).

Methods of analysis: SWOT analysis; BPMN process modelling; comparative analysis of orders (Plovdiv, Rakovski, Asenovgrad, Pazardzhik, Chirpan); regulatory analysis (Public Procurement Act, HACCP, ISO 22000).

This study is divided into four main stages. The first is "preparatory", during which information is collected and a case study is selected. The second is "field research", where interviews, surveys and observations are conducted. In the third stage, the data is processed, analyses and diagrams are prepared. In the fourth stage, conclusions and predictions are made, and conclusions and recommendations are formulated. The limitations identified are limited access to part of the documentation and focus on a single supplier.

The BPMN model of the process includes: Announcement and preparation of a tender (submission via CAIS EOP); Evaluation and conclusion of a contract; Operational implementation – orders, procurement, incoming control; Delivery by refrigerated transport and logging; Accounting and traceability (invoicing, certificates, archive).

An analysis of the logistics system is provided in Table 1.

The main challenges are in the following groups:

Administrative – High documentation burden; Formal errors → risk of removal; Delays in classification and contracting

Contractual and regulatory - Unrealistic deadlines (12-24 hours); Restrictions on product replacement; Fixed prices in the face of inflation

Logistical - Seasonal fluctuations; Shortage of qualified personnel; Lack of an integrated digital system.

Financial - Delayed payments (up to 45 days); Inability to index prices.

Table 1. Analysis of the logistics system

Key participants	Main stages	Key indicators (3 months)
Supplier Sub-suppliers Children's facilities Plovdiv Municipality	Receipt of applications (weekly). Planning and supply with incoming control. Storage according to temperature regimes (HACCP). Preparation and labelling. Delivery with GPS and temperature monitoring.	On-time deliveries: 96% Completed without deviations: 93% Complaints: 2.7%

Control authorities (BFSA, RZI)	Reporting and invoicing.	Delivery time: 24–48 hours

Proposals for optimisation. Through digitisation – integrated system (orders–warehouse–transport–invoicing); Mobile application with barcode scanning and electronic signature. Supply – Framework agreements with subcontractors; Centralised logistics centre. Coordination – Standardised electronic orders; Regular meetings with contractors. Financial solutions – Indexation for inflation; Partial advance payments. Staff - Training on public procurement and food safety; Creation of a "Quality and Coordination" team. Expected results: fewer errors, better traceability, financial sustainability and increased satisfaction.

Conceptual framework. The aim of the logistics system is to ensure high-quality, safe and timely delivery of food products to nurseries, kindergartens and schools in Plovdiv in full compliance with the Public Procurement Act (PPA), food standards (HACCP, ISO 22000) and the requirements of the contracting authority.

Levels of management - Strategic - selection of suppliers, investments; Tactical - planning of volumes and routes; Operational - processing and delivery.

Key processes

Order processing → procurement → warehouse logistics (FIFO) → routing → delivery → reporting.

The study uses the ISCPM model [8], [9] to examine effectiveness, which integrates: SRM (suppliers); ISCM (internal logistics); CRM (customers).

The model includes strategic, planning and operational levels and combines financial, process, training and customer indicators. Sustainability is achieved through a balance between four stakeholders: shareholders, customers, employees and society/nature.

The supply chain is structured and efficient, but operates in an environment of high administrative and financial uncertainty. Through digitalisation, standardisation and contractual flexibility, greater sustainability, competitiveness and long-term efficiency in the execution of public procurement contracts can be achieved.

CHAPTER THREE: EVENT MODEL OF THE SUPPLY CHAIN

This chapter is one of the core chapters of this dissertation and presents the created event model of a supply chain, which will be added to the smart agriculture platform "ZEMELA". The platform is in the process of continuous development. Previous versions of the platform are presented in [10], [11]. A review of the status of the platform and the event model has been made.

After analysing the basic event model [12] regarding its use for creating a delivery management system, an adaptation scheme was proposed, including the following steps:

- Specification of new event types. The most important ones are delivery and request.
- Specifying the list of parameters. We introduce three specific parameters that are essential for supply chain management: supplier, delivery destination, activity.
- Choosing how to represent events. For the delivery management system, we choose frames and production rules.

Creating a theoretical model. Based on the developed theoretical model, the event machine of the system was implemented. For this application, a new aspect is taken into account, which is irrelevant in the event model for crop vegetation, but which plays an essential role in planning in a supply chain management system. We will call this factor **conflict**. While in the basic event model used in ZEMELA, anomalies (deviations) in plant vegetation play a significant role, in the adapted version, which must serve the supply chain, other factors are essential. For example, conflicts are of particular importance. It is essential to define and characterise conflicts, as the event engine must be able to resolve conflict situations.

is a situation in which two or more operations, processes, or resources compete for access to the same data, resources, or system components in a way that can lead to errors, unexpected behaviour, or data loss. Some of the common types of conflicts in computer science can be: resource conflict, concurrency conflict, merge conflict, database conflict, naming conflict.

When planning food deliveries, the term "conflict" can refer to situations where there are contradictions between goals, resources, requirements, or constraints that make it hard to deliver stuff efficiently and on time. A conflict in food delivery planning can be a mismatch between different factors — such as availability, transport capacity, delivery time, customer priorities, expiry dates or logistics routes — that prevents the optimal execution of the delivery plan. Types of conflicts in this context include resource conflicts, priority conflicts, time conflicts, storage requirement conflicts, and loading and unloading conflicts.

Conflicts in delivery planning can be resolved using various methods, such as: optimisation algorithms, heuristics, constraint satisfaction; production and priority rules; planning and management systems; manual correction and re-routing by logistics operators.

A characteristic feature of supply chain management is that conflicts usually arise on the basis of the values of the parameters of orders and deliveries, respectively. In such a case, the representation of a conflict can be reduced to a constraint problem. Thus, a conflict is the triplet (P, D, C), where: P is the set of parameters involved in the conflict; D is the set of definition domains of the individual parameters; C is the set of constraints that must be satisfied to resolve the conflict.

As a problem with constraints, conflicts can also be resolved with algorithms based on the concept of k-consistency [13]. In this dissertation, a method for resolving conflicts has been chosen that uses production rules and frame structures combined with two inference methods – forward chaining and backward chaining. The implementation of this approach is presented in Chapter 5 of the dissertation.

The original version of the ZEMELA architecture provides for not a single event interpreter (event machine), but a number of different interpreters. Why is this option provided in the reference architecture of ZEMELA? Practical experience with the platform and the analyses performed show that the identification of domain events (especially incidental ones) is a serious challenge. The reasons for this are of various kinds.

Adaptation of the platform to different problem areas. There is a wide variety of domain events. For example, due to their nature, domain events in agriculture can be very different - e.g. drought, excessive moisture, plant diseases, pests, environmental pollution with various harmful substances, etc. For supply chain management, events are of a completely different nature, as shown in the previous point. It is difficult to find common characteristics for this huge variety of domain events.

Different sources. Usually, the values of the parameters characterising a domain event are obtained from different sources. Usually, the values of the parameters characterising a domain event are obtained from different sources. This is to be expected for virtual-physical systems such as the ZEMELA platform.

The time factor. In most cases, the identification process takes place over time. We can assume that at discrete points in time we can obtain and need to analyse data, usually from different sources. From this point of view, incident events are particularly difficult to identify. An additional reason may be that the event model is at a highly abstract level, while attempts to create a means of identifying, localising and managing events are only made at a lower technological and software-technical level. Taking into account the factors discussed above in the reference architecture of the ZEMELA platform, it has been proposed that the identification of domain events should become part of the event model. Thus, when adapting the platform to the application area of interest, it is possible to develop different types of event machines along with changes

in the presentation of the events themselves. As an example of a possible event machine guiding the monitoring of tomato vegetation, a machine using a cellular automaton as a theoretical model has been proposed [12].

Event machines based on finite automata. We demonstrate various possibilities for creating supply chain management systems using event machines based on Mily and Moore automata [14]. To illustrate their basic application, we will use relatively trivial examples involving a small number of elements.

First example: using a Mealy machine to avoid duplicate orders. Suppose we have a centre for receiving orders that must coordinate the activities for their fulfilment. Orders can arrive through various communication channels – electronic or paper. A customer sends an electronic request, but also a paper one for security reasons. Suppose that the management system has a service that eliminates duplicate requests. Figure 1 shows the implementation of such an event machine in the logical programming language WIN-Prolog [15].

The input data (the original sequence of requests) is represented as a list (In). The result of the machine's work (a sequence without duplicate requests) is also represented as a list (Out).

The internal states of the event machine are represented as Prolog facts (Figure 2). The machine starts with a specified initial state (in this case, 0). The example uses two requests (r1, r2), and the output symbol Λ is represented as an empty list ([ ]).

example-rdr.pl

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Remove duplicate requests
% -------------------------
% (MEALY Automat)
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Event engine
% ------------

remove_duplicate_requests(In, Out) :-
    start_event_engine(I),
    execute_event_engine(I, In, Out).

execute_event_engine(S, [ ], [ ]).

execute_event_engine(S, [H|T], Y) :-
    internal_state(S, H, [ ], N),
    execute_event_engine(N, T, Y).

execute_event_engine(S, [H|T], [X|Y]) :-
    internal_state(S, H, X, N),
    execute_event_engine(N, T, Y).
```

Figure 1. Prolog implementation of an event machine for duplicate elimination

```
% Event engine internal states
% ----------------------------

start_event_engine(0).
internal_state(0, r1, r1, 1).
internal_state(0, r2, r2, 2).
internal_state(1, r1, [ ], 1).
internal_state(1, r2, r2, 2).
internal_state(2, r1, r1, 1).
internal_state(2, r2, [ ], 2).
```

Figure 2. Internal states of an event machine for eliminating duplicates

Figure 3 shows the result of an approximate execution of the event machine for eliminating duplicate requests.

Second example: using a Milly automaton to manage combined deliveries. With it, we demonstrate a Milly automaton-based event machine for managing combined deliveries. The program implementation (WIN-Prolog) of the event machine for managing combined requests, satisfying the scenario, is shown in Figure 4.

```
Console
------------------------------------------------
BDS WIN-PROLOG 8.000 X86 S/N 0022442966 14 Jul 2022
Copyright 1989-2022 Brian D Steel (www.solanum.org)
Licensed To: Stoyanov
B=64 L=64 R=64 H=256 T=2048 P=8192 S=64 I=256 O=256
------------------------------------------------
| ?-
# 0.000 seconds to consult c:\users\user\documents\lsc\Пример2\example-rdr.pl
| ?- remove_duplicate_requests([r1, r2, r2, r1, r1, r1, r1], X).
X = [r1,r2,r1] ;
```

Figure 3. Example session with an event machine for eliminating duplicates

example-cd.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Combined delivery
% -----------------
% (MEALY Automat)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Event engine
% ------------

combined_delivery(In, Out) :-
     start(I),
     combined_delivery(I, In, Out).

combined_delivery(S, [ ], [ ]).

combined_delivery(S, [H|T], Y) :-
     internals(S, H, [ ], N),
     combined_delivery(N, T, Y).

combined_delivery(S, [H|T], [X|Y]) :-
     internals(S, H, X, N),
     combined_delivery(N, T, Y).
```

Figure 4. Event machine for combined deliveries

```
% Event engine internal states
% ----------------------------
start(0).

internals(0, f1, [ ], 0).
internals(0, f2, [ ], 0).
internals(0, w5, [ ], 1).
internals(0, w10, [ ], 4).

internals(1, f1, w5, 0).
internals(1, f2, w5, 0).
internals(1, w5, [ ], 2).
internals(1, w10, [ ], 3).

internals(2, f1, w5, rn).
internals(2, f2, w5, rn).
internals(2, w5, [ ], 3).
internals(2, w10, w5, 3).
internals(rn, [ ], w5, 0).

internals(3, f1, 'dara', 0).
internals(3, f2, 'rozovo sarce', 0).
internals(3, w5, w5, 3).
internals(3, w10, w10, 3).
internals(4, w10, w5, 3).

internals(4, f2, w10, 0).
internals(4, f1, w10, 0).
internals(4, w5, [ ], 3).
internals(4, w10, w5, 3).
```

Figure 5. Internal states of an event machine for combined deliveries

```
# 0.000 seconds to consult c:\users\user\documents\lsc\Пример2\example-cd.pl
| ?- combined_delivery([w10, w10, f1], X).
X = [w5,dara] ;
```

Figure 6. Example session with an event machine for combined deliveries

Third example: using a Moore automaton to swap the order of execution of orders. A Moore machine is a finite automaton with an output that appears in every state. For example, when represented as a graph, if the output associated with state i is x, then i/x denotes the internal state of the automaton (the corresponding vertex in the graph). A typical state transition for a Moore machine can be represented as shown in Figure 7.

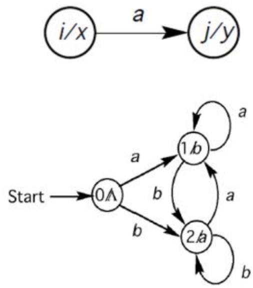

Figure 7. Example Moore automaton

Each time an internal state is entered, an output is performed. So the first output always appears immediately after the machine starts. These types of automata have an initial state but no final state, as they do not aim to accept or reject an input string. A Moore machine has one output from each state for each letter of the alphabet. For our purposes, we want to construct an event machine for exchanging the sequence for executing orders, based on a Moore machine. The program implementation (WIN-Prolog) of the machine is shown in Fig. 8. The database of internal states modelling the scenario is given in Figure 9. As in the previous example, the output symbol Λ is represented as an empty list ([ ]).

example-cser.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Exchange the sequence of execution of requests %
% ---------------------------------------------- %
% (MOOR Automat)                                 %
%                                                %
%                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Event engine
% ------------

exchange_sequence_execution_requests(In, Out) :-
     start_event_engine(I),
     exchange_sequence_execution_requests(I, In, Out).

exchange_sequence_execution_requests(S, [ ], [ ]) :-
     internals(S, [ ], Y, Z).

exchange_sequence_execution_requests(S, [ ], [X]) :-
     internals(S, X, Y, Z).

exchange_sequence_execution_requests(S, [H|T], Y) :-
     internals(S, [ ], H, N),
     exchange_sequence_execution_requests(N, T, Y).

exchange_sequence_execution_requests(S, [H|T], [A|Y]) :-
     internals(S, A, H, N),
     exchange_sequence_execution_requests(N, T, Y).
```

Figure 8. Event machine for swapping the order of execution of orders

```
% Event engine internals
% ----------------------

start_event_engine(0).
internals(0, [ ], r1, 1).
internals(0, [ ], r2, 2).
internals(1, r2, r1, 1).
internals(1, r2, r2, 2).
internals(2, r1, r1, 1).
internals(2, r1, r2, 2).
```

Figure 9. Internal states of an event machine for exchanging the order of execution of orders

The example session in Figure 10 demonstrates the use of this event machine.

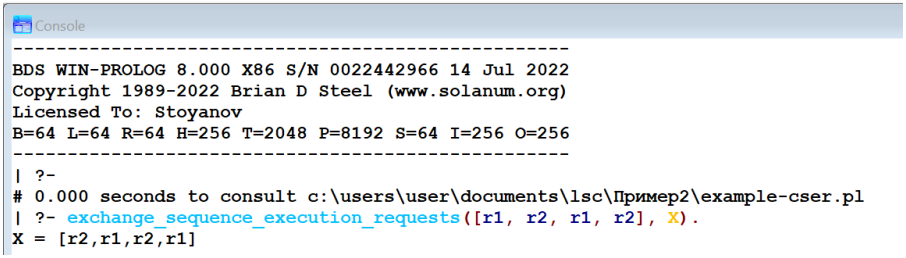

Figure 10. Example session with an event machine for swapping the order of execution of orders

Event machine based on the STRIPS concept. STRIPS performs a depth-first search in a tree of states connected by operators. The system first attempts to achieve the goal, converting prerequisites into subgoals and continuing the process until the goal is achieved or the possibilities are exhausted. Thus, STRIPS uses backward chaining to establish goals. If a condition cannot be fulfilled, the system returns to the last point of choice (OR node) through the backtracking mechanism. Since backward chaining and backtracking are built into the Prolog language, implementing a STRIPS-like system in Prolog is relatively easy.

To demonstrate the use of the STRIPS planner concept to implement a sample event machine for supply chain management, let's consider the following scenario. Let us assume that customers order the delivery of certain food products. If suppliers already have the desired products, they simply deliver them to the customer. If they do not have them, they can produce them or subcontract the delivery to a subcontractor. If we assume that they decide to produce the products themselves, it must be ensured that:

- They have sufficient stocks of raw materials;
- They have suitable employees available to do the work;
- Their machines are in good working order.

These requirements can be summarised in Table 2. The table gives specific instances of operator-related objects (e.g. purchasing raw materials), but other instances are often possible (e.g., purchasing something). The table shows that each operator has preconditions that must be satisfied before the operator can be executed. Satisfying a precondition is a subproblem of the overall task, and therefore the development plan typically has a hierarchical structure (including subplans).

We will now look at how STRIPS can be used to create an event machine that manages product delivery under the constraints imposed. Let us assume for the moment that the initial state of the problem is as follows: The customer has ordered cheese; The customer does not have any cheese; We have a team available; We do not have raw

materials; The materials are supplied by a supplier; We can afford to purchase raw materials; The machines are in working order.

Table 2. Summary of the scenario

Operator and object	Prerequisite	Effect
delivers *a product*	*product* manufactured **And** *order* placed by *customer*	*customer* has *product*
subcontracts *product*	*subcontractor* available **AND** *subcontractor* costs lower product price	*product* ready
manufactures *product*	*team* available **AND** *raw materials* available **AND** *machine* in working order	*product* ready **AND** *raw materials* reduced **AND** *machine* maintenance
orders *raw materials*	we can afford *raw materials* **AND** *raw materials* available	we have *raw materials* **AND** *money* taken from *account*
borrow *money*	Good connections with *the bank*	*money* added to *account*
Sell *assets*	*assets* exist **and** sufficient time to sell	*money* added to *account*
repair *machine*	*parts* available	*machine* ready **AND** preventive maintenance scheduled

To process an order, the event machine constructs an AND-OR search tree as shown in Figure 11.

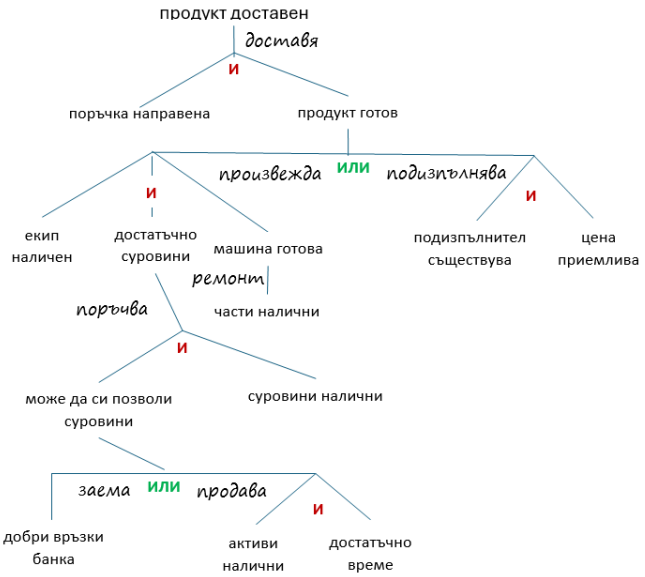

Figure 11. AND-OR tree for planning order execution

The software implementation (SWI-Prolog) of the event machine is shown in Figure 12. It should be noted that performing an operation changes the state of the world model (the internal database). For example, the operator `can_allow_raw_materials`, applied to raw materials, increases the stock of materials to a sufficient quantity to fulfil the order. In this way, it fulfils one of the prerequisites of another operator (ready). This operator can then also change the state of the world, as it results in the product being ready for delivery.

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                     %
% LSC Planner                                                         %
% ================================================================    %
% Управление на верига за доставка на високо ниво                     %
%                                                                     %
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
доставен(Customer, Product, Newplan):-% доставка на продукт
 готов(Product, Plan1),
 поръчка(Product, Customer, Plan2),
 merge([Plan1, Plan2, доставя], Newplan).

готов(Product, Newplan):-
 % subcontract the manufacturing
 наличен(подизпълнител, Plan1),
 цена_подизп_приемлива(Plan2),
 merge([Plan1, Plan2, подизпълнител], Newplan).

цена_подизп_приемлива([]):-
 % no action is required if the subcontractor's
 % price is OK, i.e. less than the disk price
 цена(подизпълнител,Price1,_),
 цена(продукт,Price2,_),
 Price1 < Price2.

готов(Product, Newplan):-
 % manufacture the product ourselves
 наличен(екип,Plan1),
 достатъчно_суровини(Plan2),
 машина_работи(Plan3),
 merge([Plan1, Plan2, Plan3, произвежда], Newplan).

достатъчно_суровини([]):-
 % no action required if we have sufficient stocks
 % already
 актуален_баланс(материали,Amount,_),
 Amount >= 1.
```

Figure 12. Program implementation of an event machine for food delivery

There are a number of different ways in which we can build a system for our example of delivering a product to a customer. For the purposes of illustration, we will assume the following scheme:

- The sequence of operators used to achieve a goal is stored as a list representing a plan.
- Each effect is represented as a clause whose last argument is the plan for achieving the effect. `delivered(our_customer, product, Plan).`
- If a particular effect does not require any actions, then the argument corresponding to its plan is an empty list. `available(subcontractor, []).`
- The prerequisites for achieving an effect are represented as Prolog rules.
- The final condition of this rule is the merge relation, which is used to merge subplans into a sequence of actions. This relation is not a standard Prolog predicate, so we will have to create it. The purpose of merge is to "flatten" a hierarchical list (the first argument) and assign the result to the second argument. We can define merge through four separate rules corresponding to different structures that the first argument may have. We can ensure that the four rules are considered mutually exclusive by using the cut operator.

Since it is not specified that we have assets or time to sell them, or that the machine is working, all these statements are considered false according to the closed world assumption.

The initial state of the event machine is shown in Figure 13. Since it is not specified, for example, that we have assets or time to sell them, or that the machine is working, all these statements are considered false according to the closed world assumption.

```
% set up the initial world state
цена(продукт,1000,[]).
цена(материали,200,[]).
цена(подизпълнител,500,[]).
актуален_баланс(сметка,0,[]).
актуален_баланс(материали,0,[]).
връзки_банка(добри,[]).
поръчка(кашкавал,вен,[]).
наличен(подизпълнител,[]).
наличен(рез_части,[]).
наличен(екип,[]).
```

Figure 13. The initial state of the event machine

The session shown in Figure 14 demonstrates the operation of the event machine.

The predicate that achieves the desired result is delivered, which depends on the conditions of the cheese being ready and the order being placed. The second condition

has already been fulfilled. The first condition becomes a subgoal that can be solved in one of two ways: through a subcontractor or through in-house cheese production.

```
% c:/Users/user/Documents/LSC/Пример1/Planner-SupplyChainBG.pl compiled 0.00 sec, 24 clauses
?- доставен(Customer, Product, Newplan).
Customer = вен,
Product = кашкавал,
Newplan = [подизпълнител, достави] ;
Customer = вен,
Product = кашкавал,
Newplan = [заем, поръчка_суровини, ремонт_машина, произвежда, достави] ;
```

Figure 14. Test session using the event machine

We can now ask our Prolog system to generate appropriate delivery plans from a specific supplier and/or for a specific product (cheese). We can also request plans to achieve any other effect that is represented in the model. For example, we can request a plan that provides us with sufficient raw materials as follows:

```
?- sufficient_raw_materials(Plan).
Plan = [loan, order_raw_materials];
No
```

CHAPTER FOUR: TECHNOLOGIES USED

This chapter describes the technologies used and their integration. In this chapter, we provide a detailed analysis of the flex expert system toolkit. We examine how flex is integrated into the LPA software package and its relationship with the Prolog language. The chapter is structured into several sections covering the main components of the system (flex expert system), its architecture, functionalities and applications in real scenarios. Flex expert system is a powerful tool for developing expert systems, created by a team of specialists in the field of artificial intelligence and logical programming. It is distinguished by its flexibility and integration with the Prolog language (Figure 15).

The integration of Flex into the LPA package is achieved through several key mechanisms:

- Shared knowledge base: Flex and Prolog share a common knowledge base, which allows for seamless information exchange between components.
- Unified syntax: Although Flex provides its own syntax for defining rules and facts, it is compatible with the Prolog syntax, which facilitates integration.
- Two-way communication: Flex can call Prolog predicates, and Prolog can use Flex functionalities, creating a flexible development environment.
- Common data structures: LPA components use compatible data structures, allowing for easy exchange of information between them.

- Integrated development environment: LPA provides a unified development environment that brings together all components, including Flex.

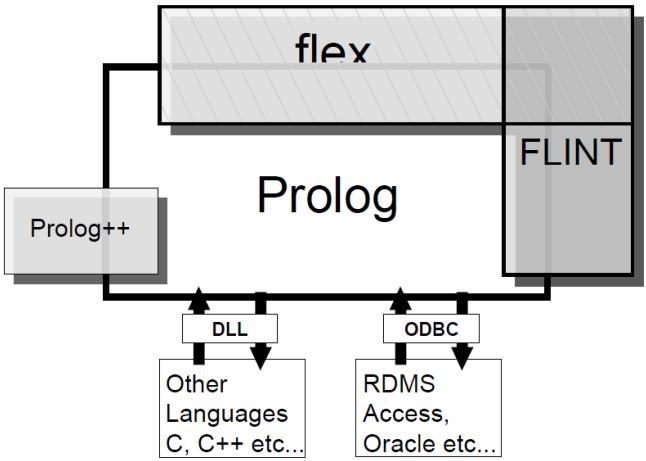

Figure 15. The LPA product suite

This integration allows developers to utilise the full potential of logic programming, combining it with the powerful tools for creating expert systems provided by flex. The result is a flexible and powerful platform for developing intelligent applications. The relationship between the flex expert system and the Prolog language is fundamental to understanding the overall architecture of the system. Flex integrates with Prolog in a way that allows their strengths to complement each other.

Prolog, as a logical programming language, provides a powerful mechanism for representing knowledge and logical inference. It works on the principle of unification and resolution, allowing declarative programming, where the developer describes what needs to be decided, rather than how exactly to decide it. Flex builds on this foundation, providing a higher level of abstraction specifically designed for creating expert systems. The architecture of Flex is designed to be extensible, allowing developers to add new functionalities and adapt the system to the specific requirements of different domains. To better understand the position of Flex Expert System in the ecosystem of expert system development tools, a comparison with other popular tools has been made. This comparison shows that Flex Expert System has its own unique place in the ecosystem, offering a combination of flexibility, power, and integration with logical programming that makes it particularly suitable for complex expert systems requiring rich knowledge representation and logical inference.

The user interaction component in Flex is a key part of the system, as it allows for the dynamic collection of knowledge that is not available in advance. Through questions, menu structures, and the possibility of explanations, the system provides an intuitive interface that combines the power of production rules with the convenience of dialogue interaction.

CHAPTER FIVE: PROTOTYPE OF A SUPPLY CHAIN MANAGEMENT SYSTEM

In this chapter, we present an event machine in which events, including conflicts, are represented as production rules and frames. The prototype of the supply chain management system using the new event machine has been implemented with Flex technology in Bulgarian and English versions.

Prototype architecture

Figure 16 shows the general architecture of the delivery management system.

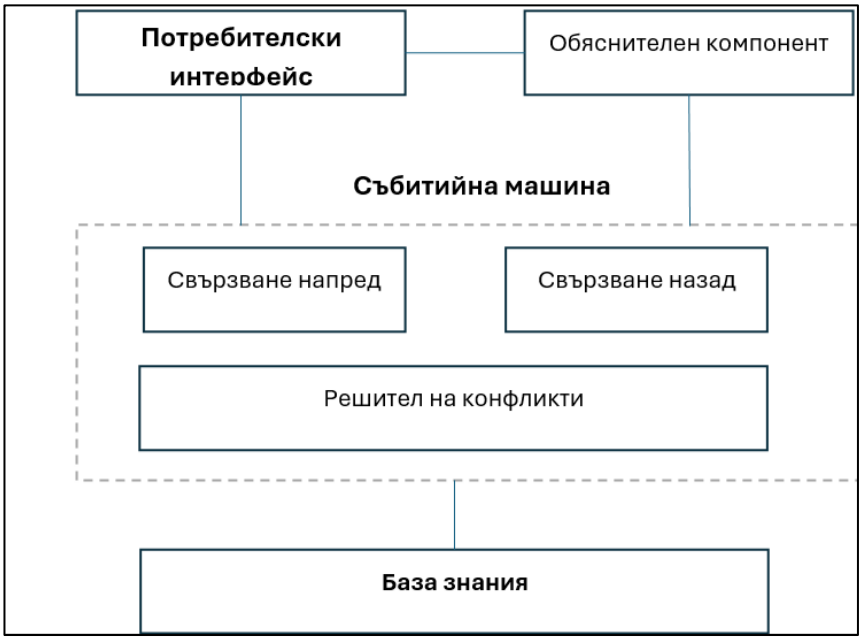

Figure 16. General architecture of the delivery management system

The system is built on the following three architectural levels:

- Knowledge base – includes the knowledge necessary for delivery planning.
- Event machine – an active component of the system. The event machine is a type of inference engine which, taking into account the current state of the problem (the

facts in the database), selects and activates the production rules stored in the knowledge base that are appropriate for solving the specific case.

- User interface – this component connects with the users of the system. An explanatory module is also integrated into the user interface.

The event machine is adapted to generate a schedule for food deliveries to kindergartens. In this example, the methods for forward and backward chaining for generating a delivery schedule are distinguished. Automatic generation of schedules and timetables is a classic problem in artificial intelligence, which involves both search components and strategic decision-making components. It has similar characteristics to resource allocation, as well as to general and specific planning.

Forward chaining and backward chaining are two contrasting methods for generating schedules given the various constraints associated with the delivery of food products to kindergartens.

In this example, information objects representing deliveries, kindergartens, suppliers, items, and periods are used. Suppliers can deliver certain products to certain kindergartens and have periods of free time when they are not available. Given this data, the main requirement is to plan daily delivery schedules so that: A supplier can only make one delivery during a given period; Only one delivery per day is made to a kindergarten.

Conflict situations are characteristic of solving this type of problem. In the example event machine, we represent conflicts as production rules. The two methods differ in how they handle these situations.

Inference method: **forward chaining.** Forward chaining is a data-based inference technique. It starts with the available data and applies inference rules to new data until the goal is achieved. This method is typically used in situations where the initial data set is extensive and the goal is to extract conclusions from it. The inference mechanism is executed in the following steps: It starts with the known facts in the database (the facts); Application of rules; Derivation of new facts; Repetition - this process is repeated until no more rules can be applied or a specific goal is achieved. Advantages of forward chaining : Simplicity; Automatic data processing; Completeness; Effectiveness in certain scenarios. The following disadvantages can be identified: Inefficiency in tasks oriented towards specific goals; Intensive use of memory; Complexity with large sets of rules. In summary, we can conclude that this is the most effective method, although there is no guarantee that a potentially existing solution will always be found. With this approach, conflicts are resolved by exchanging pre-assigned suppliers or pre-assigned deliveries. In this way, conflicts are resolved directly when they arise.

The mixed set of rules (`ruleset forward_chaining_timetable`), representing the programmatic implementation of Flex forward chaining (Figure 17), includes three basic production rules: `extend_timetable` (Figure 18), `resolve_supplier_conflict` (Figure 19) and `resolve_product_conflict`, as well as the meta-rules `select rule using first come first served`, `update ruleset by removing any unsatisfied rules and initiate by doing restart`. The production rules represent knowledge from the problem domain and, through their application, can be used to search for answers to queries (questions) to the system. The meta-rules are instructions to the event machine on how to select and apply the production rules.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Метод за извод: свързване напред
% ================================
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ruleset forward_chaining_timetable
   contains extend_timetable,
            resolve_supplier_conflict,
            resolve_product_conflict  ;
   select rule using first come first served ;
   update ruleset by removing any unsatisfied rules ;
   initiate by doing restart .
```

Figure 17. Forward linking

```
% Това правило генерира валидни детска градина, период и доставчик.
% След това ги разпределя към текущия график.
% ------------------------------------------------------------------
rule extend_timetable
   if   детска градина C необходимо разписание for период P
   and  T може сигурно достави S to детска градина C в период P
   then remember that детска градина C is доставка S by T в период P .
```

Figure 18. Rule for partial filling of a delivery schedule

In addition, the set contains meta-rules that control the operation of the event machine in "forward linking" mode. The meta-rule `using first come first served` determines how production rules are selected and applied – in this case, the first suitable rule (in the order of storage in the knowledge base) is selected and activated. The second meta-rule, `update ruleset by removing any unsatisfied rules`, tells the event machine how to update the set of production rules – here, all rules that cannot be satisfied are removed. The third meta-rule, `initiate by doing restart`, specifies how to initiate the production rules.

```
% Това правило разрешава потенциални конфликти с доставчици
% ------------------------------------------------------------
rule resolve_supplier_conflict
   if   детска градина C необходимо разписание for период P
   and  T може достави S to детска градина C в период P
   and  S is not доставя to C at any друг P1 така is наличен for период P
   and  T доставя детска градина C1 в период P и не може доставя детска градина C
   and  T1 може сигурно достави S1 to детска градина C1 в период P
   and  T1 is different from T
   then forget    that детска градина C1 is доставка something by T в период P
   and  remember that детска градина C1 is доставка S1 by T1 в период P
   and  remember that детска градина C is доставка S by T в период P .
```

Figure 19. Rule for resolving conflicts related to suppliers

The main production rule `extend_timetable` is activated multiple times to generate a possible delivery schedule. The rule works with a partially filled schedule, which is completed at each iteration. The second production rule, `resolve_supplier_conflict`, is used to resolve conflicts caused by supplier issues. The third production rule, resolve_product_conflict, is used to resolve conflicts related to the products being delivered.

Inference method: **backward chaining.** Backward chaining is a goal-based technique. It starts with the goal in the rules (the then-part) and applies them backwards to determine which facts must be true (given in the if-part of the rule) in order to achieve the goal. This method is less efficient than forward chaining, but if a solution exists, this approach will eventually find it. In this approach, conflicts are resolved by blindly revoking the last assigned supplier and delivery, replacing them with alternatives, and then attempting to fill in the rest of the schedule. If this new assignment does not result in a solution, then this process is repeated until all supplier-delivery combinations have been exhausted.

The program implementation of backward chaining is shown in Figure 20. The basic rule `backward_chain_timetable` is executed recursively, with the memorisation of kindergarten, goods, supplier and period being cancelled when no more suppliers can be proposed for the remaining kindergartens and periods.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Метод за извод: свързване назад
% ==============================
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
relation backward_chain_timetable
   if   детска градина C необходимо разписание for период P
   and  !
   and  T може сигурно достави S to детска градина C в период P
   and  remember that детска градина C is доставка S by T в период P
   and  backward_chain_timetable .

relation backward_chain_timetable .
```

Figure 20. Backward chaining

When backward linking, the event machine resolves three types of conflicts. In the second case, a potential conflict situation between suppliers is modelled due to the fact that they have already been assigned to a specific kindergarten for the desired period (Figure 21). In the third type, potential conflicts arising from the products to be delivered are identified (Figure 22).

```
% Идентифицира потенциални конфликти между доставчици поради това,
% че достачиците вече са разпределени за детска градина за даден период.
% ------------------------------------------------------------------------
relation T доставя детска градина C1 в период P и не може доставя детска градина C
   if  детска градина C1 is доставка something by T в период P
   and C1 is different from C .
```

Figure 21. Resolving the second type of conflicts

```
% Идентифицира потенциални конфликти поради стоките,
% които вече се доставят в дадена детска градина през определен период.
% ------------------------------------------------------------------------
relation S is вече доставя to C в период P1 което е конфликт с период P
   if   детска градина C is доставка S by somebody в период P1
   and  P1 is different from P .
```

Figure 22. Resolving the third type of conflicts

Starting and executing the event machine. The event machine can be activated using the basic meta-rule `action run` (Figure 23). This rule incorporates the production rule `timetable_method`, as well as the two new meta-rules `forward_chaining_timetable` and `backward_chaining_timetable`. In this implementation of the delivery management system, the user is given the option to choose the method that the system will use to generate the delivery schedule. Once the method has been selected in the meta-rule, the corresponding rule for its activation and management is activated. The backward chaining is initiated by the meta-rules `backward_chaining_timetable` (Figure 24). The body of the rule is a chain of sequential activation (`restart`) of backward chaining (`backward_chain_timetable`) and display of the generated schedule (`print_table`).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Сратиране на събитийната машина
% ================================
% Преди действителното стартиране потребителя може да избере
% между два метода за извод (свързване напред или свързване назад).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
action run ;
  do  ask timetable_method
  and write( 'Разсъждавайки ...' )
  and nl
  and if   the answer to timetable_method is 'forward chaining solution'
      then do forward_chaining_timetable
      else do backward_chaining_timetable
      end if .
```

Figure 23. Starting the event machine

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Инициране на избрания метод за извод
% ====================================

% Това действие инициира решението за обратно свързване.
% ------------------------------------------------------
action backward_chaining_timetable ;
   do   restart
   and  backward_chain_timetable
   and  print_table .
```

Figure 24. Initiating the backward chaining inference method

Forward chaining is initiated in a similar manner.

The knowledge base is built on the following three levels: Production rules; Meta-rules; Database.

In artificial intelligence, production rules, also known as condition-action rules or "if-then" rules, are a fundamental way of representing knowledge and reasoning. They describe how a system should behave based on a set of conditions. Essentially, they state that if a certain condition is met, then a specific action must be taken. In logical programming languages (such as Prolog, CLIPS, Jess, etc.), production rules are a fundamental mechanism for representing knowledge and deriving new facts. Characteristics of working with production rules: They support a declarative style of programming; They support a deduction mechanism; They work on a fact base – a set of statements that are considered true at a given moment; They can be executed iteratively – one rule can trigger another rule (a chain of conclusions).

In artificial intelligence, meta-rules are rules that govern the application of other rules (often called object-level rules or production rules). They act as a control mechanism, guiding the artificial intelligence system on how to use its knowledge base.

User interface. Figure 25 shows a segment of the system's user interface. The user interface can be significantly more attractive. On the one hand, our goal is to demonstrate the operation of the event machine using a simplified user interface. On the other hand, in logical programming, the data required for a specific case (the facts) is usually assumed to be available in the system database. In the current prototype implementation, the user can choose the method by which the delivery schedule is generated. The user interface has a built-in explanation module, which is activated using the `because` rule.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
% Потребителски интерфейс
% =======================
%%%%%%%%%%%%%%%%%%%%%%%%%%
question timetable_method
  Кой метод ? ;
  choose one of 'forward chaining solution',
                'backward chaining solution'
  because 'Събитийната машина може да генерира график с един от двата метода - но не едновременно!' .
```

Figure 25. User interface

Use of templates. Similar to a synonym, a template aids the readability of KSL expressions. In the supply chain management system, templates are used to support the operation of the event machine (Figure 26). A template is also used to improve the readability and understanding of rules and relationships.

The template in Figure 27 supports the process of assigning a supplier and food product to a specific kindergarten for a specific period according to the specifications in the input data.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Шаблони
% =======
% Пример - първият шаблон специфицира следните типове фрази:
% 1) на детска градина X се доставя Y от Z през N-ти период.
% 2) на детска градина X не се доставя Y от Z през N-ти период.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

template доставка
   детска градина ^ is доставка ^ by ^ в период ^ ;
   детска градина ^ is not доставка ^ by ^ в период ^ .

template без_разписание
   детска градина ^ необходимо разписание for период ^ .

template потенциален_доставчик
   ^ може достави ^ to детска градина ^ в период ^ .

template сигурен_доставчик
    ^ може сигурно достави ^ to детска градина ^ в период ^ .

template неподходящ_доставчик
   ^ доставя детска градина ^ в период ^ и не може доставя детска градина ^ ;
   ^ does not доставя any ^ в период ^ и така is на разположение to доставя детска градина ^ .

template неподходящ_субект
   ^ is вече доставя to ^ в период ^ което е конфликт с период ^ ;
   ^ is not доставя to ^ at any друг ^ така is наличен for период ^ .
```

Figure 26. Event machine templates

```
% Генерира всеки доставчик и стока за дадена детска градина и
% период според спецификациите на изходните данни.
% -------------------------------------------------------------
relation T може достави S to детска градина C в период P
   if T is an instance of доставчик
   whose свободно_време does not include P and
   whose продукти include S and
   whose детски_градини include C .
```

Figure 27. Template supporting the assignment of a supplier, product and period

Sample sessions with the delivery management system. To demonstrate the functionality and features of the implemented event machine prototype, we will now look at sample sessions for generating a schedule for food deliveries to kindergartens. The data for kindergartens, suppliers and products are simplified and symbolic, as the aim is to present the machine's operating mechanism more clearly. Two examples are mainly considered, applying forward chaining and backward chaining solutions,

respectively. To start an example session with the event machine, enter the run command in the command line (Figure 28).

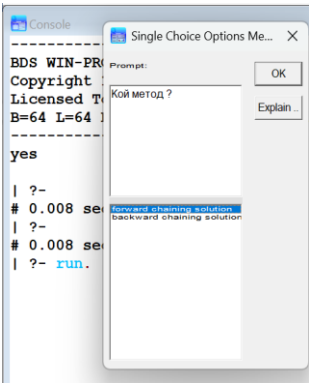

Figure 28. Starting a session

After starting the event machine, the user can select the method to be used to generate the delivery schedule (Fig. 29). The reduced version of the schedule includes three kindergartens, the food products to be delivered, the respective suppliers, and the times of day when the deliveries will be made.

The second demonstration example shows the selection of a backward connection for the method that the event machine will use to generate a delivery schedule under the same conditions – the state of the database (the facts) remains unchanged. The result of the execution is shown in Figure 30. Comparing the two results, it can be seen that the event machine generates the same result for the same facts, modelling a specific real-life scenario.

```
Разсъждавайки ...
детска_градина(астра)
период(1,мляко,иван)
период(2,сирене,иван)
период(3,плодове,ласка)
период(4,вода,стоян)
период(5,месо,георги)

детска_градина(божур)
период(1,хляб,венета)
период(2,вода,стоян)
период(3,сок,стоян)
период(4,месо,георги)
период(5,сирене,иван)

детска_градина(гергина)
период(1,плодове,ласка)
период(2,хляб,венета)
период(3,месо,георги)
период(4,мляко,иван)
период(5,вода,стоян)

yes
```

Figure 29. Generated forward-looking delivery schedule

```
Разсъждавайки ...
детска_градина(астра)
период(1,мляко,иван)
период(2,сирене,иван)
период(3,плодове,ласка)
период(4,вода,стоян)
период(5,месо,георги)

детска_градина(божур)
период(1,хляб,венета)
период(2,вода,стоян)
период(3,сок,стоян)
период(4,месо,георги)
период(5,мляко,иван)

детска_градина(гергина)
период(1,плодове,ласка)
период(2,хляб,венета)
период(3,месо,георги)
период(4,мляко,иван)
период(5,вода,стоян)

yes
```

Figure 30. Generated delivery schedule with backward linking

CONCLUSION

This dissertation examines the analysis, modelling and development of an integrated supply chain management system with a focus on public procurement and public institutions. Synergy has been achieved between supply chain management theory, risk management and the development of an intelligent software prototype based on event architecture and expert technologies.

The proposed methodology for integrating the event model into the ZEMELA platform demonstrates the possibility of effectively applying artificial intelligence, logical inference and adaptability in a real logistics environment. By modelling the interactions between contractors, suppliers and consumers, the concept of an intelligent supply chain that responds proactively to events and risks is confirmed.

The results have both theoretical and practical value, providing a framework for building sustainable, transparent and digitally managed logistics systems in the public sector. The prototype allows for optimised planning and distribution, faster response to risk, improved communication and increased traceability in the execution of public procurement contracts.

The research confirms the thesis that the supply chain should be viewed as a dynamic, adaptive and knowledge-based system in which information technology and expert solutions are key to sustainability and efficiency. The developed architecture contributes to computer science through the integration of computer science, logistics and management.

Future developments include the integration of machine learning and predictive analytics, the construction of a cloud and microservice architecture, the implementation of blockchain for transparency and security, expansion to other sectors, the development of a simulation module for training, and the creation of a national platform for intelligent public procurement management.

In conclusion, the research shows how modern information technologies can transform public procurement management and lays the foundation for building an intelligent, sustainable and adaptive supply chain.

MAIN RESULTS IN THE DISSERTATION

1. A conceptual event model and architecture for intelligent supply chain management has been developed as an extension of the ZEMELA platform.
2. A methodology for integrating the event model has been developed, demonstrating the use of artificial intelligence, logical inference and adaptive behaviour in a real logistics environment.
3. Implementation of the prototype and its testing in real conditions.

The dissertation achieves a combination of theoretical modelling and practical implementation, offering a scientifically sound architecture and a working prototype for intelligent supply chain management in the public sector. The results have both a scientific contribution to the field of informatics and direct applicability in the management of public procurement.

ACKNOWLEDGEMENTS

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

I owe enormous gratitude to my academic supervisor, Assoc. Prof. Emil Doichev, PhD. I would also like to thank Prof. Asya Stoyanova-Doicheva, PhD, and Prof. Stanimir Stoyanov, PhD, for their trust in me and their dedication in every situation, as well as for their ideas and advice, which I greatly appreciate.

I would like to express my special gratitude to Assoc. Prof. Veneta Tabakova-Komsalova for her positive energy and warm attitude, valuable guidance and unreserved assistance.

BIBLIOGRAPHY

[1] Kotler, P., Kartajaya, H., & Setiawan, I. (2021). Marketing 5.0: Technology for Humanity. Wiley.

[2] Sirieix, L., Kledal, P. R., & Sulitang, T. (2011). Organic food consumers' trade-offs between local or imported, conventional or organic products: A qualitative study in Shanghai. International Journal of Consumer Studies, 35(6), 670–678. doi:https://doi.org/10.1111/j.1470-6431.2010.00960.x.

[3] Christopher, M., & Holweg, M. (2017). Supply Chain 2.0 revisited: A framework for managing volatility-induced risk in the supply chain. International Journal of Physical Distribution & Logistics Management, 2-17. doi:https://doi.org/10.1108/IJPDLM-09-2016-0245.

[4] Riza Nugraha, E. (2024). Digital Transformation in Marketing: From Marketing 3.0 to Marketing 5.0. Global International Journal of Innovative Research, 2(11), 2528–2544. doi:https://doi.org/10.59613/global.v2i10.348.

[5] Chopra, S. &. (2021). Supply Chain Management: Strategy, Planning, and Operation (7th ed.). (Pearson, Ed.).

[6] Martin Christopher, Helen Peck. (1 July 2004). Building the Resilient Supply Chain. International Journal of Logistics Management, 15(2), 1-14. doi:https://doi.org/10.1108/09574090410700275.

[7] Wieland, A. (2021). Dancing the supply chain: Toward transformative supply chain management. Journal of Supply Chain Management, 57(1), 58-73. doi: https://doi.org/10.1111/jscm.12248.

[8] Saleheen, F., & Habib, M. (2022). Integrated Supply Chain Performance Measurement Model for the Manufacturing Industry. Journal of Service Science and Management, 15, 55-70. doi:10.4236/jssm.2022.152005

[9] Saleheen, F., & Habib, M. (2022). An integrated model for measuring supply chain performance in manufacturing industry. Journal of Service Science and Management, 15(2), 55-70. doi:doi:10.4236/jssm.2022.152005.

[10] Stanimir Stoyanov, Jordan Todorov, Ivan Stoyanov, Veneta Tabakova-Komsalova, Lubka Dukovska. (2021). ZEMELA – An Intelligent Agriculture Platform. Big Data, Knowledge and Control Systems Engineering – BdKCSE'2021, (pp. 1-6). doi:10.1109/BdKCSE53180.2021.9627248.

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

[12] Stoyanov, I. (06 2023 r.). Research on the creation of a virtual operator in smart agriculture infrastructure. Dissertation. Plovdiv, Bulgaria: Plovdiv University "Paisii Hilendarski".

[13] S.J.Russell, P. Norvig. (2016). Artificial Intelligence. A Modern Approach. Pearson. Retrieved from https://www.amazon.com/Artificial-Intelligence-Approach-Stuart-Russell/dp/9332543518.

[14] J.L.Hein. (2009). Prolog Experiments in Discrete Mathematics, Logic, and Computability. Portland State University. Retrieved from https://samples.jbpub.com/9780763772062/PrologLabBook09.pdf

[15] Rebecca Shalfield, Clive Spenser, Brian D. Steel and Alan Westwood. (2021). WIN-PROLOG 8.0, London, England: Logic Programming Associates Ltd.