Пловдивски университет „Паисий Хилендарски“
Факултет по математика и информатика
Катедра „Компютърни системи“

МОДЕЛИРАНЕ НА ИНТЕЛИГЕНТНА ВЕРИГА ЗА ДОСТАВКИ В СИСТЕМИ ЗА ИНТЕЛИГЕНТНО ЗЕМЕДЕЛИЕ

АВТОРЕФЕРАТ
на дисертационен труд за присъждане на образователна и научна степен “доктор“ в област на висше образование 4. Природни науки, математика и информатика; професионално направление 4.6. Информатика и компютърни науки;
докторска програма Информатика

Задочен докторант: Ласка Делкова Костадинова-Цанкова
Научен ръководител: доц. д-р Емил Дойчев

Пловдив, 2026

Дисертационният труд е обсъден и насочен за защита на разширено заседание на катедрен съвет на катедра: „Компютърни системи“ към Факултет по математика и информатика при ПУ „Паисий Хилендарски“, проведено на 13.03.2026 г.

Дисертационният труд „МОДЕЛИРАНЕ НА ИНТЕЛИГЕНТНА ВЕРИГА ЗА ДОСТАВКИ В СИСТЕМИ ЗА ИНТЕЛИГЕНТНО ЗЕМЕДЕЛИЕ“ съдържа 212 страници, 64 фигури, 19 таблици и 234 литературни източника, които включват научна литература от областта на информатиката и компютърните науки и икономиката. Списъкът на авторските публикации се състои от 10 заглавия, от които 2 цитирани в дисертационния труд.

Защитата на дисертационният труд ще се състои на 22.05.2026 г. от ……. часа в Заседателната зала на нова сграда на ПУ „Паисий Хилендарски“, гр. Пловдив, бул. „България“ №236.

Материалите по защитата са на разположение на интересуващите се в Деканата на ФМИ, нова сграда на ПУ, каб. 330, всеки работен ден от 8:30 до 17:00 часа.

СЪДЪРЖАНИЕ

УВОД

Настоящият дисертационен труд разглежда изследването, моделирането и създаването на ефективна система за управление на веригата за доставки за реално действаща компания, изпълняваща обществени поръчки в Пловдив. Основната му цел е разработването на иновативен модел и архитектура за оптимизиране на логистичните процеси и управлението на риска при снабдяване на стратегически публични институции – болници, училища, детски градини и социални услуги. Съвременната динамична и взаимосвързана икономическа среда повишава сложността и несигурността във веригите за доставки. Управление на риска се превръща в ключов фактор за устойчивост, надеждност и адаптивност. Това е особено значимо при обществените поръчки, където изискванията за прозрачност, навременност и отчетност са завишени. В отговор на тези предизвикателства дисертацията предлага концептуален модел, архитектурна рамка и прототип на интегрирана система за управление на веригата за доставки, разработена като надграждане на платформата ЗЕМЕЛА. Целта е разширяване на нейните функционалности и адаптирането ѝ към специфичните потребности на публичните институции. Предложеното решение е насочено към оптимизиране на снабдяването и проактивно управление на вътрешни и външни рискове чрез използване на изкуствен интелект, машинно обучение, сценарийно планиране и симулационно моделиране.

Актуалността на изследването произтича от нарастващата конкуренция и конфликтност между участниците във веригата за доставки, което налага прилагането на интегриран и научно аргументиран подход. Трудът предлага практически приложими и адаптивни решения, съобразени със спецификата на обществения сектор в България и основани на международно утвърдени добри практики.

Предмет на настоящото дисертационно изследване е управлението на риска във веригите за доставки с особен акцент върху противоречията и предизвикателствата произтичащи от нормативната и процедурна рамка на обществените поръчки.

Обект на проучване в дисертационния труд е веригата за доставки в нейната цялост, разглеждана като съвкупност от процеси по планиране, организиране, управление и контрол на потока от стоки и услуги от доставчика до крайния получател.

Основната цел на настоящия дисертационен труд е да се разработи модел на интелигентна верига за доставки. Прототипна реализация на модела да се интегрира в съществуващата платформа ЗЕМЕЛА, осъществявайки интеграция между платформата и логистичната система на фирма-доставчик на хранителни продукти, осъществяваща доставки за детски заведения на територията на Община Пловдив. В рамките на тази цел се предвижда изграждане на събитиен модел за проследяване, анализ и оптимизация на ключовите точки във веригата за доставки – от възникването на потребностите, през процедурите по обществена поръчка, до окончателното изпълнение и приемане на доставките от страна на детските заведения. Моделът ще позволи адаптивно поведение на логистичната система при настъпване на събития с потенциален риск и ще подпомага по-добра координация между всички заинтересовани страни.

За постигане на поставената цел се формулират следните конкретни **изследователски задачи**:

- Анализ на съвременни вериги за доставки и интегрирането им в платформи за интелигентно земеделие.
- Разработване на модел и архитектура на системата.
- Прототипна реализация.

За постигане на поставената цел и в съответствие с формулираните задачи, в рамките на настоящия дисертационен труд е предложена **методика за провеждане на изследването**, която се основава на систематичен и поетапен подход към създаването на интелигентна система за управление на веригата за доставки.

ПЪРВА ГЛАВА: ВЕРИГИ ЗА ДОСТАВКИ

В условията на глобална икономическа и технологична свързаност веригите за доставки са ключов елемент на съвременното производство и търговия. Те обхващат потоците от материали, информация и финанси от източника до крайния клиент и се превръщат в стратегически фактор за конкурентоспособност. Днес организациите се изправят пред предизвикателства като пандемии, геополитическа нестабилност, инфлация и екологични изисквания. В тази глава на дисертационния труд са разгледани различни дефиниции на „верига за доставки“, основни модели, рискове и прекъсвания, както и примери от реални компании. Представени са и базови концепции за интелигентни вериги за доставки и тяхната логистика. Основните понятия са:

Същност на веригата за доставки. Веригата за доставки представлява мрежа от организации и процеси, които осигуряват придвижването и трансформацията на продукти от суровини до краен потребител, включително обратна логистика. Ефективността зависи от координация, синхронизация и адаптивност.

Ключови елементи на управлението на веригата за доставки (Supply Chain Management, SCM). SCM обединява планиране, снабдяване, производство, доставка и обратна логистика. ERP и други дигитални системи осигуряват прозрачност и контрол в реално време чрез IoT, блокчейн и аналитични инструменти.

Базовите концепции на интелигентни вериги за доставки са:

Директно насочване към потребителя (Direct-to-Consumer, DTC). Моделът Direct-to-Consumer (DTC) опростява дистрибуционния канал чрез елиминиране на посредниците и изграждане на пряка връзка между производител и краен клиент. Това осигурява пълна проследимост на продукта – от произхода и производството до логистиката [1]. DTC може да се реализира чрез директни партньорства с крайни потребители или институции (училища, детски градини, болници), чрез фермерски пазари и ясно обозначаване на произхода на продуктите [2], [3]. В този модел производителят поема контрол върху логистични дейности като складиране, прогнозиране, изпълнение на поръчки и управление на връщания [4].

Доставки „при поискване“ (On-Demand Supply). Моделът „при поискване“ се активира от конкретна клиентска заявка и акцентира върху бързината на изпълнение. Често обаче не отчита устойчивостта, оптималните маршрути или качествените параметри на доставката, което може да доведе до по-високи разходи, удължено време за доставка и увеличен екологичен отпечатък [5].

Логистиката на хранителен склад, свързващ производители с детски градини, изисква координирано управление на снабдяването, съхранението и дистрибуцията, за да се гарантира доставка на безопасна и прясна храна за деца.

Снабдяване и входяща логистика. Складът работи с местни фермери (сезонни и свежи продукти, намалени „хранителни километри“), производители

на преработени храни и специализирани доставчици за специфични диети. Ключови дейности са прогнозиране на търсенето, управление на доставчици и строг контрол на качеството. Транспортът включва поддържане на студена верига, синхронизиран график на доставките и проверка при приемане.

Съхранение и управление на запасите. Складът използва температурно обособени зони, FIFO принцип и системи за проследяване на наличности в реално време. Особено важно е управлението на партиди и срокове на годност поради обслужването на уязвими групи. Целта е минимизиране на разваляването и предотвратяване на недостиг или свръхналичности.

Обработка на поръчки и изходяща логистика. Поръчките се комплектоват чрез персонал или автоматизация, опаковат се с оглед запазване на качеството и се етикетират ясно. Доставките се оптимизират по маршрут и времеви прозорци, като при необходимост се поддържа студена верига до крайния получател. Предвидени са процеси за управление на връщания и устойчиво третиране на отпадъци.

Информационни системи и контрол. ERP системи подпомагат координацията на поръчки, запаси и транспорт, а GPS и IoT осигуряват мониторинг в реално време. Ефективната комуникация с детските градини гарантира навременни доставки и прозрачност.

Съответствие и устойчивост. Процесите се базират на HACCP и националните изисквания за безопасност на храните, включително НАРЕДБА № 14 ОТ 9 ДЕКЕМВРИ 2021 Г. ЗА ХИГИЕНАТА НА ХРАНИТЕ, както и на принципите за проследимост. Устойчивостта включва локално снабдяване, екологични опаковки и използване на електрически или хибридни превозни средства.

Хранителният склад е ключово звено между производители и детски градини. Ефективността на системата зависи от интегрирано управление на качеството, проследимостта, дигиталните технологии, нормативното съответствие и устойчивите практики, гарантиращи сигурно и навременно хранително снабдяване.

Като заключение може да се отбележи, че веригите за доставки преминават през период на значителна трансформация. Организациите, които успеят да се адаптират към новите условия чрез внедряване на съвременни технологии, фокус върху устойчивостта и изграждане на гъвкави системи, ще придобият съществено конкурентно предимство [6]. За българските компании това е едновременно предизвикателство и възможност за модернизация и по-добра интеграция в глобалните вериги за доставки. В бъдеще развитието на тези вериги ще зависи от способността на организациите да балансират между ефективност и устойчивост, глобализация и локализация, както и между автоматизация и човешки фактор [7]. Компаниите, които постигнат този баланс и изградят гъвкави и устойчиви системи, ще бъдат най-успешни в новата ера на глобалната търговия.

ВТОРА ГЛАВА: КОНЦЕПТУАЛНА РАМКА

Главата представя анализ на ефективността на доставчик на хранителни продукти в гр. Пловдив, обслужващ общински и частни детски заведения по реда на Закон за обществените поръчки. Подходът е интердисциплинарен – логистика, публична администрация, управление на качеството и икономика – с цел оценка на реалното изпълнение и формулиране на подобрения.

Методология на изследването. Приложен е смесен метод:

- *Количествен анализ* – брой доставки, закъснения, обеми, разходи, анкетни резултати.
- *Качествен анализ* – интервюта, наблюдение на процеси, оценка на комуникация и организационна култура.

Това позволява оценка както на административно-правната рамка, така и на оперативната логистика.

Методи използвани за събиране на данни са:

- Документален анализ – профили на купувача, технически спецификации по ЗОП, договори, вътрешни документи, HACCP план.
- Полуструктурирани интервюта – 7 ключови участници (мениджмънт, логистика, община, директори).
- Анкетно проучване – 15 служители от детски заведения и 5 от доставчика; оценка на срокове, качество, комуникация и прозрачност.
- Наблюдение – проследяване на реалния логистичен процес (заявка–склад–транспорт–предаване).

Методи за анализ са: SWOT анализ; BPMN моделиране на процеса; Сравнителен анализ на поръчки (Пловдив, Раковски, Асеновград, Пазарджик, Чирпан); Нормативен анализ (ЗОП, HACCP, ISO 22000).

Настоящето изследването е разделено на 4 основни етапа. Първият е „подготвителен“, при който се събира информация и се прави избор на казус. Вторият е „теренно проучване“ където се провеждат интервюта, анкети и наблюдения. През третия етап данните се обработват, изготвят се анализи и диаграми. През четвърти етап се правят изводи и предловения, формулират се заключения и препоръки. Като ограничения са идентифицирани ограничен достъп до част от документацията и фокус върху един доставчик.

BPMN модел на процеса включва: Обява и подготовка на оферта (подаване чрез ЦАИС ЕОП); Оценка и сключване на договор; Оперативно изпълнение – заявки, снабдяване, входящ контрол; Доставка с хладилен транспорт и протоколиране; Отчетност и проследимост (фактуриране, сертификати, архив).

Анализ на логистичната система е направен в таблица 1.

Основните предизвикателства са в следните групи:

Административни - Висока документационна тежест; Формални грешки → риск от отстраняване; Забавяния при класиране и договориране

Договорни и регулаторни - Нереалистични срокове (12–24 ч.); Ограничения при замяна на продукти; Фиксирани цени при инфлация

Логистични - Сезонни колебания; Недостиг на квалифициран персонал; Липса на интегрирана дигитална система.

Финансови - Забавени плащания (до 45 дни); Невъзможност за индексация на цени.

Таблица 1. Анализ на логистичната система

Основни участници	**Основни етапи**	**Ключови показатели (3 месеца)**
Доставчик Поддоставчици Детски заведения Община Пловдив Контролни органи (БАБХ, РЗИ)	Прием на заявки (седмично). Планиране и снабдяване с входящ контрол. Съхранение по температурни режими (HACCP). Подготовка и етикетиране. Доставка с GPS и температурен мониторинг. Отчетност и фактуриране.	Доставки в срок: 96% Изпълнени без отклонения: 93% Рекламации: 2.7% Време за доставка: 24–48 ч.

Предложения за оптимизация. Чрез Дигитализация – Интегрирана система (заявки–склад–транспорт–фактуриране); Мобилно приложение с баркод сканиране и електронен подпис. Снабдяване - Рамкови споразумения с поддоставчици; Централизиран логистичен център. Координация - Стандартизирани електронни заявки; Регулярни срещи с възложители. Финансови решения - Индексация при инфлация; Частични авансови плащания. Персонал - Обучения по ЗОП и хранителна безопасност; Създаване на екип „Качество и координация“. Очаквани резултати: по-малко грешки, по-добра проследимост, финансова устойчивост и повишена удовлетвореност.

Концептуална рамка. Целта на логистичната система е да се осигури качествена, безопасна и навременна доставка на хранителни продукти към детски ясли, градини и училища в Пловдив при пълно съответствие със Закона за обществените поръчки (ЗОП), хранителните стандарти (HACCP, ISO 22000) и изискванията на възложителя.

Нива на управление - Стратегическо – избор на доставчици, инвестиции; Тактическо – планиране на обеми и маршрути; Оперативно – обработка и доставка.

Ключови процеси

Обработка на заявки → снабдяване → складова логистика (FIFO) → маршрутизация → доставка → отчетност.

Изследването използва модела ISCPM [8], [9] за изследване на ефективността, който интегрира: SRM (доставчици); ISCM (вътрешна логистика); CRM (клиенти).

Моделът включва стратегическо, планиращо и оперативно ниво и комбинира финансови, процесни, обучителни и клиентски показатели. Устойчивостта се постига чрез баланс между четири заинтересовани страни: акционери, клиенти, служители и общество/природа.

Веригата за доставки е структурирана и ефективна, но работи в среда на висока административна и финансова несигурност. Чрез дигитализация, стандартизация и договорна гъвкавост може да се постигне по-висока устойчивост, конкурентоспособност и дългосрочна ефективност при изпълнение на обществени поръчки.

ТРЕТА ГЛАВА: СЪБИТИЕН МОДЕЛ НА ВЕРИГА ЗА ДОСТАВКИ

Тази глава е една от същинските за настоящата дисертация и в нея е представен създадения събитиен модел на верига за доставки, който ще се добави към платформата за интелигентно земеделие „ЗЕМЕЛА“. Платформата е в процес на непрекъснато развитие. Предишни версии на платформата са представени в [10], [11]. Направен е преглед на състоянието платформата и събитийния модел.

След анализ на базовия събитиен модел [12], относно използването му за създаване на система за управление на доставки е предложена схема за адаптация, включваща следните стъпки:

- Специфициране на нови типове събития. Най-съществените са доставка и заявка.
- Конкретизиране списъка с параметрите. Въвеждаме три конкретни параметъра, които са съществени за управлението на веригата за доставки: доставчик, цел на доставка, активност.
- Избор на представяне на събитията. За системата за управление на доставките избираме фрейми и производствени правила.

Създаване на теоретичен модел. На основата на разработения теоретичен модел бе имплементирана събитийната машина на системата. За това приложение се отчита един нов аспект, който е без значение в събитийния модел за вегетация на земеделските култури, но който играе съществена роля при планирането в една система за управление на верига на доставки. Този фактор ще го наречем **конфликт**. Докато в основния събитиен модел, използван в ЗЕМЕЛА, аномалиите (отклонения) във вегетацията на растенията играят съществена роля, то в адаптираната версия, която трябва да обслужва веригата за доставка, съществени са други фактори. Така например, от особено значение са конфликтите. Съществено е да се даде определение и да се характеризират конфликтите, понеже събитийната машина трябва да е в състояние да разрешава конфликтни ситуации.

Конфликт в компютърните науки е ситуация, при която две или повече операции, процеси или ресурси се конкурират за достъп до едни и същи данни,

ресурси или системни компоненти по начин, който може да доведе до грешки, непредвидено поведение или загуба на данни. Някои от често срещани типа конфликти в компютърните науки могат да бъдат: конфликт при достъп до ресурси (Resource Conflict), конфликт при конкурентен достъп (Concurrency Conflict), конфликт при сливане на код (Merge Conflict), конфликт в бази от данни (Database Conflict), конфликт на идентификатори или имена (Naming Conflict).

При планиране на доставки на храни, понятието „конфликт“ може да бъде съотнесено до ситуации, в която възникват противоречия между цели, ресурси, изисквания или ограничения, които затрудняват ефективното и навременно изпълнение на доставките. Конфликт при планиране на доставки на храни може да бъде несъответствие между различни фактори — като наличности, транспортен капацитет, време на доставка, приоритети на клиенти, срокове на годност или логистични маршрути — което пречи на оптималното изпълнение на плана за доставки. Видове конфликти в този контекст: конфликт на ресурси, конфликт на приоритети, времеви конфликт, конфликт с изисквания за съхранение, конфликт при зареждане и разтоварване.

Конфликти при планиране на доставките могат да бъдат решени с различни методи, като например: Оптимизационни алгоритми, евристики, удовлетворяване на ограничения; Производствени и приоритетни правила; Системи за планиране и управление; Ръчно коригиране и пренасочване от логистични оператори.

Характерно за управлението на вериги за доставки е, че конфликтите възникват обикновено на основата на стойностите на параметрите на заявките, съответно на доставките. В такъв случай представянето на един конфликт може да се сведе до проблем с ограничения. Така конфликт е тройката (P, D, C), където: P е множеството на участващите в конфликта параметри; D е множеството на дефиниционните области на отделните параметри; C е множеството на ограниченията, които трябва да бъдат удовлетворени за решаване на конфликта.

Като проблем с ограничения конфликтите могат да се решават също с алгоритми, базиране на понятието k-консистентност [13]. В настоящата дисертация е избран метод за решаване на конфликти, който използва производствени правила и фреймови структури, съчетани с два метода за извод – свързване напред и свързване назад. Реализацията на този подход е представена в пета глава на дисертацията.

В оригиналната версия на архитектурата на ЗЕМЕЛА се предвижда не един единствен интерпретатор на събития (събитийна машина), а множество различни интерпретатори. Защо е дадена тази възможност в референтната архитектура на ЗЕМЕЛА? От практическия опит с платформата и на основата на направените анализи се оказва, че идентифицирането на домейн-събитията (особено инцидентните) е сериозно предизвикателство. Причините за това са от различно естество.

Адаптация на платформата за различни проблемни области. Наблюдава се голямо разнообразие на домейн-събития. Така например, поради своето естество домейн-събитията в земеделието могат да бъдат много различни - напр.,

засушаване, прекомерна влага, болести по растенията, вредители, натоварване на околната среда с различни вредни вещества и т.н. За управлението на верига за доставки събитията са от съвсем друго естество, както беше показано в предишната точка. За това огромно разнообразие от домейн-събития трудно се намират общи техни характеристики.

Различни източници. Обикновено стойностите на характеризиращите едно домейн-събитие параметри се получават от различни източници. Обикновено стойностите на параметрите, характеризиращи едно домейн-събитие, се получават от различни източници. Това е очаквано за виртуално-физически системи, каквато е платформата ЗЕМЕЛА.

Факторът време. Процесът на идентифициране в повечето случаи протича във времето. Можем да приемем, че в дискретни времеви моменти можем да получаваме и е необходимо да анализираме данни, обикновено от различни източници. От тази гледна точка, особено трудно се идентифицират инцидентните събития. Допълнителна причината може би е в това, че събитийният модел е на високо абстрактно ниво, докато опитите за създаване на средство за идентифициране, локализиране и управление на събитията се правят само на по-ниско технологично и програмно-технично ниво. Отчитайки дискутираните по-горе фактори в референтната архитектура на платформата ЗЕМЕЛА е предложено идентификацията на домейн-събитията да стане част от събитийния модел. Така, при адаптацията на платформата за интересуващата ни приложна област да е възможно заедно с промените на самото представяне на събитията да се разработват различни видове събитийни машини. Като пример за възможна събитийна машина, направляваща мониторинга на вегетацията на домати, е предложена машина, използваща като теоретичен модел клетъчен автомат [12].

Събитийни машини, базирани на крайни автомати. Демонстрираме различни възможности за създаване на системи за управление на вериги за доставки като използваме събитийни машини, базирани на автомати на Мили и съответно на Мур [14]. Представяйки принципното им приложение ще използваме сравнително тривиални примери, включващи малко на брой елементи.

Първи пример: използване на автомат на Мили за избягване дублиране на поръчки. Да предположим, че имаме един център за получаване на заявки, който трябва да координира дейностите по тяхното изпълнение. Заявките могат да пристигат по различни комуникационни канали – електронни или хартиени. Един клиент изпраща електронна заявка, но за сигурност и хартиена такава. Да предположим, че в системата за управление съществува услуга, която премахва дублирането на заявки. На Фигура 1 е дадена имплементацията на такава събитийна машина на езика за логическо програмиране WIN-Prolog [15].

Входните данни (оригиналната последователност от заявки) са представени като списък (In). Резултатът от работата на машината (последователност без дублирани заявки) е представена също като списък (Out).

Вътрешните състояния на събитийната машина са представени като Пролог-факти (Фигура 2). Машината започва с означено начално състояние (в случая 0). В примера се използват две заявки (r1, r2), а изходният символ Λ е представен като празен списък ([ ]).

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).
```

Фигура 1. Пролог имплементация на събитийна машина за елиминиране на дубликати

```
% 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).
```

Фигура 2. Вътрешни състояния на събитийна машина за елиминиране на дубликати

На Фигура 3 е показан резултатът от примерно изпълнение на събитийната машина за елиминиране на дублиращи заявки.

Втори пример: използване на автомат на Мили за управление на комбинирани доставки. С него демонстрираме базирана на автомат на Мили събитийна машина за управление на комбинирани доставки. Програмната реализация (WIN-Prolog) на събитийната машина за управление на комбинирани заявки, удовлетворяваща сценария, е дадена на Фигура 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] ;
```

Фигура 3. Примерна сесия със събитийна машина за елиминиране на дубликати

```
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).
```

Фигура 4. Събитийна машина за комбинирани доставки

```
% 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).
```

Фигура 5. Вътрешни състояния на събитийна машина за комбинирани доставки

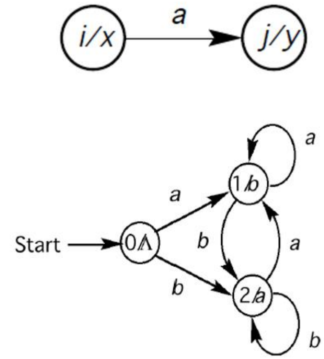
```
# 0.000 seconds to consult c:\users\user\documents\lsc\Пример2\example-cd.pl
| ?- combined_delivery([w10, w10, f1], X).
X = [w5,dara] ;
```

Фигура 6. Примерна сесия със събитийна машина за комбинирани доставки

Трети пример: използване на автомат на Мур за размяна последователността на изпълнение на поръчките. Машината на Мур е краен автомат с изход, който се появява във всяко състояние. Например, при представяне като граф, ако свързаният със състояние i изход е x, тогава с i/x се означава вътрешното състояние на автомата (съответния възсел в графа). Типичен преход на състояние за машина на Мур може да бъде представен по начин, представен на Фигура 7.

Всеки път, когато се влезе в едно вътрешно състояние, се извършва изход. Така че първият изход винаги се появява веднага след стартиране на машината. Този вид автомати имат начално състояние, но няма крайно състояние, тъй като не целят приемане или отхвърляне на входен низ. Машината на Мур има един край от всяко състояние за всяка буква от азбуката. За нашите цели искаме да конструираме събитийна машина за размяна последователността за изпълнение на поръчки, базирана на автомат на Мур. Програмната имплементация (WIN-Prolog) на машината е представена на Фиг. 8. Базата данни на вътрешните

състояния, моделиращи сценария, е дадена на Фигура 9. Както и в предишния пример изходният символ Λ е представен като празен списък ([ ]).



Фигура 7. Примерен автомат на Мур

```
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).
```

Фигура 8. Събитийна машина за размяна последователността на изпълнение на поръчки

```
% 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).
```

Фигура 9. Вътрешни състояния на събитийна машина за размяна последователността на изпълнение на поръчки

Примерната сесия на Фигура 10 демонстрира използването на тази събитийна машина.

```
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-cser.pl
| ?- exchange_sequence_execution_requests([r1, r2, r1, r2], X).
X = [r2,r1,r2,r1]
```

Фигура 10. Примерна сесия със събитийна машина за размяна последователността на изпълнение на поръчки

Събитина машина, базирана на STRIPS концепцията. STRIPS извършва търсене в дълбочина в дърво от състояния, свързани чрез оператори. Системата първо се опитва да постигне целта, като при наличие на предпоставки ги преобразува в подцели и продължава процеса, докато целта бъде изпълнена или възможностите се изчерпат. Така STRIPS използва свързване назад за установяване на целите. При невъзможност дадено условие да бъде изпълнено, системата се връща към последната точка на избор (ИЛИ-възел) чрез механизма на възврат. Тъй като свързването назад и backtracking са вградени в езика Пролог, реализирането на система, подобна на STRIPS, в Пролог е сравнително лесно.

За да демонстрираме използването на концепцията на STRIPS планировчик за реализиране на примерна събитийна машина за управление на верига на доставки нека разгледаме следния сценарий. Да приемем, че клиенти поръчват доставката на някакви хранителни продукти. Ако доставчиците вече имаме желаните продукти, те просто ги доставят на клиента. Ако нямаме такива, могат да ги произведат или да възложат доставката на подизпълнител. Ако приемем, че решат сами да произвеждат продуктите, трябва да е гарантирано, че:

- Разполагат с достатъчен запас от суровини;
- Имат подходящи служители на разположение за извършване на работата;
- Машините им са в добро работно състояние.

Тези изисквания могат да бъдат обобщени в Таблица 2. В таблицата са дадени специфични инстанции на обекти, свързани с оператори (например закупуване на суровини), но често са възможни и други инстанции (например закупуване на нещо). Таблицата показва, че всеки оператор има предварителни условия, които трябва да бъдат удовлетворени преди операторът да може да бъде изпълнен. Удовлетворяването на предварително условие е подпроблем на цялостната задача и затова планът за разработване обикновено има йерархична структура (включва и подпланове).

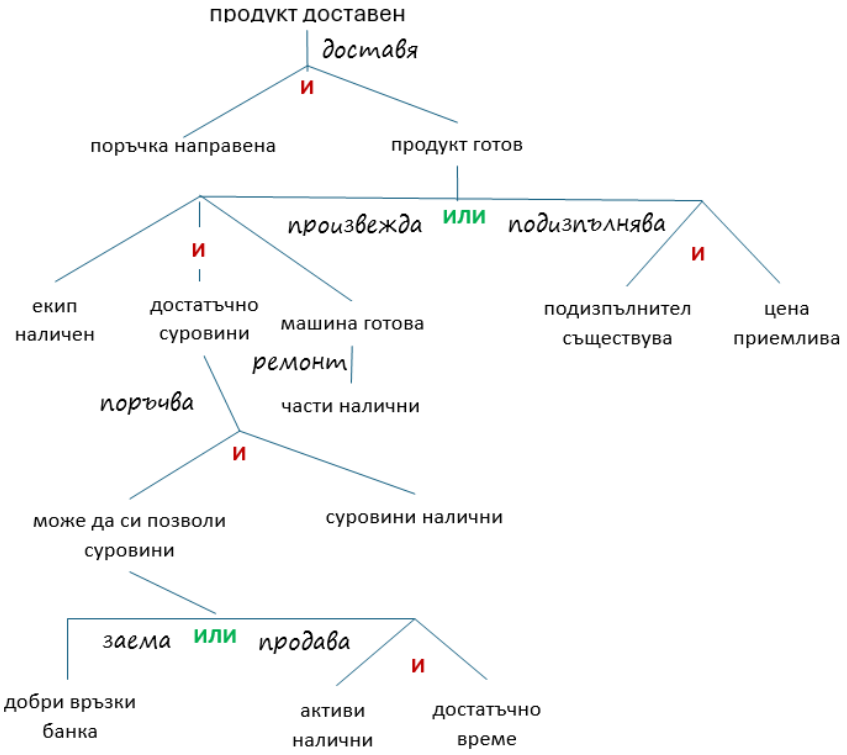
Сега ще разгледаме как STRIPS може да се използва за създаване на събитийна машина, управляваща доставка на продукти в условията на направените ограничения. Нека за момента приемем, че първоначалното състояние на проблема е както следва: Клиентът е поръчал кашкавал; Клиентът не разполага с кашкавал; Имаме на разположение екип; Нямаме суровини; Материалите се предлагат от доставчик; Можем да си позволим закупуване на суровини; Машините са в работно състояние.

Таблица 2. Обобщение на постановката на сценария

Оператор и обект	**Предусловие**	**Ефект**
доставя *продукт*	*продукт* произведен **И** *поръчка* направена от *клиент*	*клиент* има *продукт*
подизпълнява *продукт*	*подизпълнител* наличен **И** разходи *подизпълнител* по-малки цена продукт	*продукт* готов
произвежда *продукт*	*екип* наличен **И** *суровини* налични **И** *машина* изправна	*продукт* готов **И** *суровини* намалени **И** *машина* профилактика

поръчва *суровини*	можем да си позволим *суровини* **И** *суровини* налични	имаме *суровини* **И** *пари* взети от *сметка*
заема *пари*	добри връзки *банка*	*пари* добавени в *сметка*
продава *активи*	*активи* съществуват **И** достатъчно време за продажба	*пари* добавени в *сметка*
ремонтира *машина*	*части* налични	*машина* в готовност **И** профилактика назначена

За да обработи една поръчка събитийната машина построява едно И-ИЛИ дърво за търсене както показано на Фигура 11.



Фигура 11. И-ИЛИ дърво за планиране изпълнение на поръчка

Програмната имплементация (SWI-Prolog) на събитийната машина е дадена на Фигура 12. Трябва да се има предвид, че извършването на една операция променя състоянието на модел на света (вътрешната база данни). Например, операторът `може_позволи_суровини`, приложен към суровините, повишава запасите от материали до достатъчно количество, за да изпълним поръчката. По

този начин той изпълнява една от предпоставките на друг оператор (готов). След това този оператор може също да промени състоянието на света, тъй като това води до готовност на продукта за доставка.

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                          %
% 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.
```

Фигура 12. Програмна реализация на събитийна машина за доставка на хранителни продукти

Съществуват редица различни начини, по които можем да изградим система за нашия пример за доставка продукт на клиент. За целите на илюстрацията ще приемем следната схема:

- Последователността от оператори, използвани за постигане на цел, се съхранява като списък, представляващ план.

- Всеки ефект е представен като клауза, чийто последен аргумент е планът за постигане на ефекта. `доставен(our_customer, product, Plan).`
- Ако конкретен ефект не изисква никакви действия, тогава аргументът, съответстващ на неговия план, е празен списък. `наличен(подизпълнител, []).`
- Предпоставките за постигане на ефект са представени като правила на Пролог.
- Крайното условие на това правило е merge релацията, която се използва за сливане на подпланове в една последователност от действия. Тази релация не е стандартен предикат на Пролог, така че ще трябва да го създадем. Целта на merge е да "изглади" йерархичен списък (първия аргумент) и да присвои резултата на втория аргумент. Можем да дефинираме merge чрез четири отделни правила, съответстващи на различни структури, които може да има първият аргумент. Можем да гарантираме, че четирите правила се считат за взаимно изключващи се чрез използване на cut оператора.

Тъй като не е посочено, че имаме активи или време да ги продадем, или че машината работи, всички тези твърдения се считат за неверни според предположението за затворения свят.

Началното състояние на събитийната машина е представено на Фигура 13. Тъй като не е посочено например, че имаме активи или време да ги продадем, или че машината работи, всички тези твърдения се считат за неверни според предположението за затворения свят.

```
% 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,[]).
```

Фигура 13. Началното състояние на събитийната машина

Представената на Фигура 14 сесия демонстрира работата на събитийната машина.

Предикатът, който постига желания резултат е доставен, което зависи от условията на кашкавалът готов и поръчката направена. Второто условие вече е изпълнено. Първото условие се превръща в подцел, която може да бъде решена по един от двата начина: чрез подизпълнител или чрез собствено производство на кашкавал.

```
% 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] ;
```

Фигура 14. Тестова сесия с използване на събитийната машина

Вече можем да поискаме от нашата Пролог система да генерира подходящи планове за доставки от точно определен доставчик и/или за точно определен продукт (кашкавал). Можем също да поискаме планове за постигане на всеки друг ефект, който е представен в модела. Например, можем да поискаме план, който да ни предостави достатъчно суровини, както следва:

```
?- достатъчно_суровини(Plan).
Plan = [заем, поръчка_суровини];
No
```

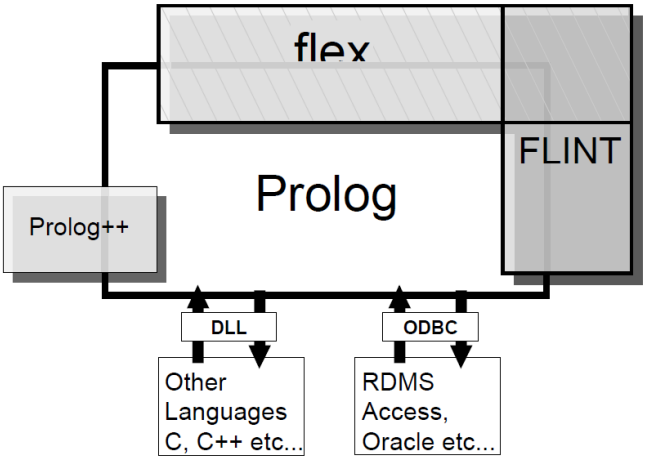
ЧЕТВЪРТА ГЛАВА: ИЗПОЛЗВАНИ ТЕХНОЛОГИИ

В тази глава са описани използваните технологии и тяхната интеграция. В тази глава правим подробен анализ на инструментариума на flex expert system. Разглеждаме как flex се интегрира в софтуерния пакет LPA и неговата връзка с езика Пролог. Главата е структурирана в няколко раздела, които обхващат основните компоненти на системата (flex expert system), нейната архитектура, функционалности и приложения в реални сценарии. Flex expert system представлява мощен инструмент за разработка на експертни системи, създаден от екип от специалисти в областта на изкуствения интелект и логическото програмиране. Той се отличава със своята гъвкавост и интеграция с езика Пролог (Фигура 15).

Интеграцията на flex в пакета LPA се осъществява чрез няколко ключови механизма:

- Споделена база от знания: Flex и Пролог споделят обща база от знания, което позволява безпроблемен обмен на информация между компонентите.
- Унифициран синтаксис: Въпреки че flex предоставя свой собствен синтаксис за дефиниране на правила и факти, той е съвместим със синтаксиса на Пролог, което улеснява интеграцията.

- Двупосочна комуникация: Flex може да извиква предикати на Пролог, а Пролог може да използва функционалности на flex, създавайки гъвкава среда за разработка.
- Общи структури от данни: Компонентите на LPA използват съвместими структури от данни, което позволява лесен обмен на информация между тях.
- Интегрирана среда за разработка: LPA предоставя единна среда за разработка, която обединява всички компоненти, включително flex.



Фигура 15. Пакетът от продукти на LPA

Тази интеграция позволява на разработчиците да използват пълния потенциал на логическото програмиране, комбинирайки го с мощните инструменти за създаване на експертни системи, предоставени от flex. Резултатът е гъвкава и мощна платформа за разработка на интелигентни приложения. Връзката между flex expert system и езика Пролог е фундаментална за разбирането на цялостната архитектура на системата. Flex се интегрира с Пролог по начин, който позволява взаимно допълване на техните силни страни.

Пролог, като език за логическо програмиране, предоставя мощен механизъм за представяне на знания и логически извод. Той работи на принципа на унификация и резолюция, позволявайки декларативно програмиране, където разработчикът описва какво трябва да бъде решено, а не как точно да се реши. Flex надгражда върху тази основа, предоставяйки по-високо ниво на абстракция, специално проектирано за създаване на експертни системи. Архитектурата на flex е проектирана да бъде разширяема, позволявайки на разработчиците да добавят

нови функционалности и да адаптират системата към специфичните изисквания на различни домейни. За да разберем по-добре позицията на flex expert system в екосистемата на инструментите за разработка на експертни системи, е направено сравнение с други популярни инструменти. Това сравнение показва, че flex expert system има своето уникално място в екосистемата, предлагайки комбинация от гъвкавост, мощ и интеграция с логическото програмиране, която го прави особено подходящ за сложни експертни системи, изискващи богато представяне на знания и логически извод.

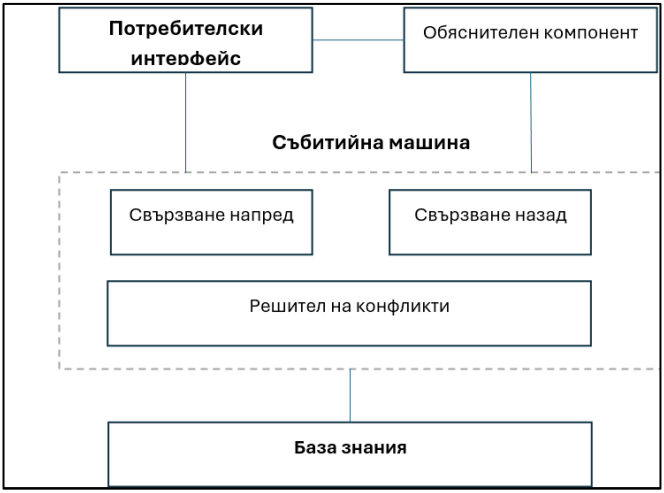
Компонентът за взаимодействие с потребителя във Flex е ключова част от системата, тъй като позволява динамично събиране на знания, които не са налични предварително. Чрез въпросите, меню структурите и възможността за обяснения, системата предоставя интуитивен интерфейс, съчетаващ мощта на продукционните правила с удобството на диалоговото взаимодействие.

ПЕТА ГЛАВА: ПРОТОТИП НА СИСТЕМА ЗА УПРАВЛЕНИЕ НА ВЕРИГА ЗА ДОСТАВКИ

В тази глава представяме събитийна машина, в която събитията, включително конфликтите, се представят като производствени правила и фрейми. Прототипът на системата за управление на доставки, използващ новата събитийна машина, е реализиран с технологията Flex с версии на български и на английски език.

Архитектура на прототипа

На Фигура 16 е дадена общата схема на архитектурата на системата за управление на доставки.



Фигура 16. Обща архитектура на системата за управление на доставки

Системата е изградена на следните три архитектурни нива:

- База знания – включва необходимите за планиране на доставките знания.
- Събитийна машина – активен компонент на системата. Събитийната машина е вид машина за извод, която отчитайки моменталното състояние на проблема (фактологията в базата данни) избира и активира подходящи за решаване на конкретния случай производствени правила, съхранявани в базата знания.
- Потребителски интерфейс – този компонент осъществява връзка с потребителите на системата. В потребителския интерфейс е интегриран и обяснителен модул.

Събитийната машина е адаптирана за генериране на разписание за доставки на хранителни продукти за детски градини. В този пример се разграничават методите за предно и обратно верижно свързване за генериране график за извършване на доставки. Автоматично генериране на графици и разписания е класически проблем в изкуствения интелект, който включва както компоненти за търсене, така и стратегически компоненти за вземане на решения. Той има сходни характеристики с разпределението на ресурсите, както и с общо и специфично планиране.

Свързването напред и свързването назад са два контрастиращи метода за генериране на графици предвид различните ограничения, съпътстващи доставката на хранителни продукти за детски градини.

В този пример се използват информационни обекти, представляващи доставки, детски градини, доставчици, артикули и периоди. Доставчиците могат да доставят определени продукти за определени детски градини и да имат периоди на свободно време, когато не са на разположение. Като се имат предвид тези данни, основното изискване е да се планират ежедневните графици на доставките, така че: Един доставчик може да доставя само една доставка през даден период; За една детска градина се прави само една доставка на ден.

Характерно за решаване на такъв вид проблеми е възникване на конфликтни ситуации. В примерната събитийна машина конфликтите представяме като производствени правила. Двата метода се различават по това как се справят с тези ситуации.

Метод на извод: **свързване напред.** Свързването напред е техника за извод, базирана на данни. Тя започва с наличните данни и прилага правила за извод за нови данни, докато не бъде постигната целта. Този метод обикновено се използва в ситуации, когато началният набор от данни е обширен и целта е да се извлекат заключения от него. Механизмът за извод се изпълнява на следните стъпки: Започва с известните факти в базата данни (фактологията); Прилагане на правила; Извеждане на нови факти; Повторение - този процес се повтаря, докато не могат да се прилагат повече правила или не се постигне определена цел. Предимства на

свързването напред: Простота; Автоматична обработка на данни; Изчерпателност; Ефективност в определени сценарии. Като недостатъци мове да се идентифицират: Неефективност при задачи, ориентирани към конкретни цели; Интензивно използване на паметта; Сложност с големи набори от правила. Обобщавайки можем да заключим, че това е най-ефективният метод, въпреки че няма гаранция, че винаги ще бъде намерено потенциално съществуващо решение. При този подход конфликтите се разрешават чрез размяна на предварително назначени доставчици или предварително назначени доставки. По този начин конфликтите се разрешават директно, когато възникнат.

Смесеното множеството правила (`ruleset forward_chaining_timetable`), представящо програмната реализация на свързването напред на Flex (Фигура 17), включва три базови производствени правила: `extend_timetable` (Фигура 18), `resolve_supplier_conflict` (Фигура 19) и `resolve_product_conflict`, както и мета-правилата `select rule using first come first served, update ruleset by removing any unsatisfied rules и initiate by doing restart`. Производствените правила представят знания от проблемната област и посредством тяхното прилагане могат да се търсят отговори на заявките (въпросите) към системата. Мета-правилата са указания към събитийната машина за това, как да се селектират и прилагат производствените правила.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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 .
```

Фигура 17. Свързване напред

```
% 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 .
```

Фигура 18. Правило за частично запълване на график за доставки

Освен това, множеството съдържа мета-правила, управляващи работата на събитийната машина в режим „свързване напред“. Мата-правилото `using first come first served` определя начина на избор и прилагане на проиводствените правила – в случая се избира първото (в реда на съхраняване в базата знания) подходящо правило и то се активира. Второто мета правило `update ruleset by removing any unsatisfied rules` казва на събитийната машина как да бъде актуализирано множеството от производствени правила – тук се премахват всички правила, които не могат да бъдат удовлетворени. Третото мета-правило `initiate by doing restart` специфицира начина за иницииране на производствените правила.

```
% 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 .
```

Фигура 19. Правило за разрешаване на конфликти, свързани с доставчиците

Основното производствено правило `extend_timetable` се активира многократно за да генерира възможен график за доставки. Правилото работи с частично запълнен график, който се допълва на всяка итерация. Второто производствено правило `resolve_supplier_conflict` се използва за разрешаване на конфликти, предизвикани от проблеми с доставчиците. Третото производствено правило resolve_product_conflict се използва за разрешаване на конфликти, свързани с доставяните продукти.

Метод за извод: **свързване назад.** Свързването назад е техника, основана на цели. Тя започва с целта в правилата (then-частта) и ги прилага обратно за да определи кои факти трябва да са верни (дадени в if-частта на правилото) за да се постигне целта. Този метод е по-малко ефективен от свързването напред, но ако съществува решение, този подход в крайна сметка ще го намери. При този подход конфликтите се разрешават чрез сляпа отмяна на последния разпределен доставчик и доставка, замяната им с алтернативни и след това опит за попълване на останалата част от графика. Ако това ново разпределение не доведе до решение, тогава този процес се повтаря, докато всички комбинации доставчик - доставка бъдат изчерпани.

Програмната реализация на свързването назад е дадена на Фигура 20. Базовото правило `backward_chain_timetable` се изпълнява рекурсивно, като запомнянето на детска градина, стока, доставчик и период се отменя, когато не

могат да бъдат предложени повече доставчици за оставащите детски градини и периоди.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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 .
```

Фигура 20. Свързване назад

При свързването назад събитийната машина решава три вида конфликти. Във втория случай се моделира потенциална конфликтна ситуация между доставчици поради това, че те вече са разпределени за определена детска градина за желания период (Фигура 21). При третия вид се идентифицират потенциални конфликти, породени от продуктите за доставка (Фигура 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 .
```

Фигура 21. Решаване на втори вид конфликти

```
% 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 .
```

Фигура 22. Решаване на трети вид конфликти

Стартиране и изпълнение на събитийната машина. Събитийната машина може да бъде активирана с помощта на базовото мета-правило `action run` (Фигура 23). В това правило са вградени производственото правило `timetable_method`, както и двете нови мета-правила `forward_chaining_timetable` и `backward_chaining_timetable`. В тази реализация на системата за управление на доставки на потребителя се предоставя възможност да избере метода, който системата ще използва за генериране на графика за доставки. След като методът е избран в мета-правилото се активира съответното правило за активирането и управлението му. Иницииране на свързването назад се извършва от мета-правилото

backward_chaining_timetable (Фигура 24). Тялото на правилото е верига от последователно активиране (restart) на свързване назад (backward_chain_timetable) и показване на генерирания график (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 .
```

Фигура 23. Стартиране на събитийната машина

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Иницииране 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 .
```

Фигура 24. Иницииране на метода за извод „свързване назад“

Аналогично се извършва иницииране на свързването напред.

Базата знания е изградена на следните три нива: Производствени правила; Мета-правила; База данни.

В изкуствения интелект, продукционните правила, известни още като правила за условие-действие или правила „ако-тогава“, са основен начин за представяне на знания и разсъждения. Те описват как една система трябва да се държи въз основа на набор от условия. По същество те заявяват, че ако е изпълнено определено условие, тогава трябва да се предприеме конкретно действие. В езиците за логическо програмиране (като Prolog, CLIPS, Jess и др.) производствените правила са основен механизъм за представяне на знание и извеждане на нови факти. Характеристики на работа с производствени правила: Поддържат декларативен стил на програмиране; Поддържат механизъм на дедукция; Работят върху база от факти – набор от твърдения, които в даден

момент се считат за истина; Могат да се изпълняват итеративно – едно правило може да задейства друго правило (верига от заключения).

В изкуствения интелект мета-правилата са правила, които управляват прилагането на други правила (често наричани правила на обектно ниво или производствени правила). Те действат като контролен механизъм, насочващ системата с изкуствен интелект как да използва своята база знания.

Потребителски интерфейс. На Фигура 25 е представен сегмент от потребителския интерфейс на системата. Потребителският интерфейс може да бъде значително по-атрактивен. От една страна нашата цел е да се демонстрира работата на събитийната машина, като се използва опростен потребителски интерфейс. От друга страна в логическото програмиране необходимите данни за конкретния случай (фактологията) обикновено се приема, че са налични в базата данни на системата. В актуалната прототипна имплементация потребителят може да избира метода, по който да се генерира графика за доставки. В потребителския интерфейс е вграден модула за пояснения, който се активира с помощта на правилото `because`.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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!' .
```

Фигура 25. Потребителски интерфейс

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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 ^ .
```

Фигура 26. Шаблони на събитийната машина

Използване на шаблони. Подобно на синоним, шаблонът подпомага четливостта на KSL изразите. В системата за управление на веригата за доставки се използват шаблони за подпомагане работата на събитийната машина (Фигура 26). Шаблон се използва също за подобряване четливостта и разбирането на правилата и релациите.

Шаблонът на Фигура 27 подпомага процеса на назначаване доставчик и хранителен продукт за определена детска градина и за определен период според спецификациите на изходните данни.

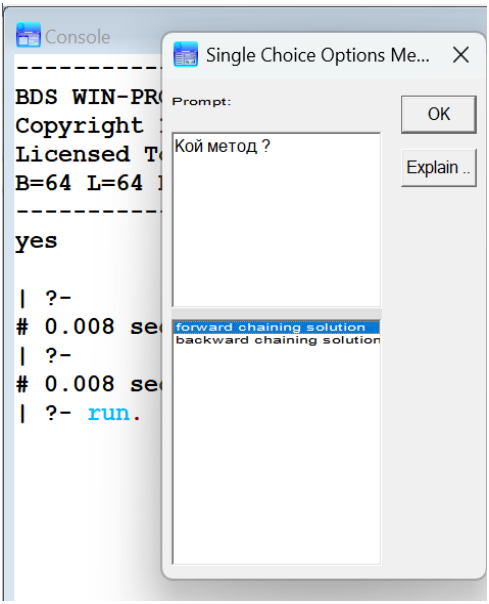
```
% Генерира всеки доставчик и стока за дадена детска градина и
% период според спецификациите на изходните данни.
% ------------------------------------------------------------
relation T може достави S to детска градина C в период P
   if T is an instance of доставчик
   whose свободно_време does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

Фигура 27. Шаблон, подпомагащ назначаване на доставчик, продукт и период

Примерни сесии със системата за управление на доставки. За демонстриране работоспособността и функциите на реализирания прототип на събитийна машина, в тази точка ще разгледаме примерни сесии за генериране на график за доставки на хранителни продукти на детски градини. Данните за детските градини, доставчиците и продуктите са опростени и символични, понеже целта е да се представи по-ясно механизмът на работа на машината. Основно са разгледани два примера, прилагащи получени със свързване напред и съответно свързване назад решения. За да се стартира примерна сесия със събитийната машина се въвежда целта run в командния ред (Фигура 28).

След старта на събитийната машина потребителят може да избере метода, който да се използва за генериране на графика за доставки (Фиг.29). В редуцирания вид на графика са включени три детски градини, доставяните хранителни продукти, съответните доставчици, както и периодите през деня, в които ще бъдат осъществени доставките.

Вторият демонстрационен пример показва избор на свързване назад за метод, който събитийната машина ще използва за генериране на график за доставки при същите условия – състоянието на базата данни (фактологията) остава непроменена. Резултатът от изпълнението е даден на Фигура 30. Сравнявайки двата резултата се вижда, че събитийната машина генерира един и същ резултат при еднаква фактология, моделираща конкретен реален сценарий.



Фигура 28. Стартиране на сесия

```
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
```

Фигура 29. Генериран график за доставки със свързване напред

```
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
```

Фигура 30. Генериран график за достаки със свързване назад

ЗАКЛЮЧЕНИЕ

Настоящият дисертационен труд изследва анализа, моделирането и разработването на интегрирана система за управление на веригата за доставки с фокус върху обществените поръчки и публичните институции. Постигната е синергия между теорията на управлението на веригата за доставки, управлението на риска и разработването на интелигентен софтуерен прототип, базиран на събитийна архитектура и експертни технологии.

Предложената методика за интеграция на събитийния модел в платформата ЗЕМЕЛА доказва възможността за ефективно прилагане на изкуствен интелект, логически извод и адаптивност в реална логистична среда. Чрез моделиране на взаимодействията между възложители, доставчици и потребители се потвърждава концепцията за интелигентна верига за доставки, която реагира проактивно на събития и рискове.

Резултатите имат както теоретична, така и практическа стойност, като предлагат рамка за изграждане на устойчиви, прозрачни и дигитално управлявани логистични системи в публичния сектор. Прототипът позволява оптимизиране на планирането и разпределението, по-бърза реакция при риск, подобрена комуникация и повишена проследимост при изпълнение на обществени поръчки.

Изследването потвърждава тезата, че веригата за доставки трябва да се разглежда като динамична, адаптивна и базирана на знания система, в която информационните технологии и експертните решения са ключови за устойчивостта и ефективността. Разработената архитектура представлява принос към информатиката чрез интеграция на компютърни науки, логистика и управление.

Бъдещото развитие включва интеграция на машинно обучение и предиктивна аналитика, изграждане на облачна и микросървисна архитектура, внедряване на блокчейн за прозрачност и сигурност, разширяване към други сектори, разработване на симулационен модул за обучение и създаване на национална платформа за интелигентно управление на обществените доставки.

В заключение, изследването показва как съвременните информационни технологии могат да трансформират управлението на обществените доставки и поставя основа за изграждане на интелигентна, устойчива и адаптивна верига за доставки.

ОСНОВНИ РЕЗУЛТАТИ В ДИСЕРТАЦИОННИЯ ТРУД

1. Разработен е концептуален събитиен модел и архитектура за интелигентно управление на верига за доставки като надстройка на платформата ЗЕМЕЛА.
2. Създадена е методика за интеграция на събитийния модел, демонстрираща използване на изкуствен интелект, логически извод и адаптивно поведение в реална логистична среда.
3. Реализация на прототипа и тестването му в реални условия.

Дисертационният труд постига съчетание между теоретично моделиране и практическа реализация, като предлага научно обоснована архитектура и работещ прототип за интелигентно управление на вериги за доставки в публичния сектор. Резултатите имат както научен принос в областта на информатиката, така и директна приложимост в управлението на обществени доставки.

БЛАГОДАРНОСТИ

Искам да изразя своята благодарност към личностите, без които този дисертационен труд не би бил възможен.

Огромна благодарност дължа на научния ми ръководител доц. д-р Емил Дойчев. Освен това благодаря на проф. д-р Ася Стоянова-Дойчева и проф. д-р Станимир Стоянов за доверието в мен и всеотдайността във всяка ситуация, идеите и съветите, които оценявам изключително много.

Изказвам специална благодарност към доц. д-р Венета Табакова-Комсалова за позитивната енергия и сърдечно отношение, ценните насоки и безрезервна помощ.

БИБЛИОГРАФИЯ

[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 (7 изд.). (Pearson, Ред.).

[6] Martin Christopher, Helen Peck. (1 07 2004 г.). 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. и 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, (стр. 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), (стр. 1-5). Warsaw, Poland. doi:10.1109/IS57118.2022.10019674.

[12] Стоянов, И. (06 2023 г.). Изследвания за създаване на виртуален оператор в инфраструктура за интелигентно земеделие. Дисертация. Пловдив, България: Пловдивски университет „Паисий Хилендарски“.

[13] S.J.Russell, P. Norvig. (2016). Artificial Intelligence. A Modern Approach. Pearson. Извлечено от 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. Извлечено от 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.