ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

МАШИННО-ТЕХНОЛОГИЧЕН ФАКУЛТЕТ

Катедра „Технология на машиностроенето и металорежещи машини“

маг. инж. Костадин Филипов Стоянов

РАЗРАБОТВАНЕ НА ПОДХОД ЗА ОПЕРАТИВНА СЪВМЕСТИМОСТ В РЕКОНФИГУРИРАЩИ СЕ ПРОИЗВОДСТВЕНИ СИСТЕМИ ЗА МАШИНОСТРОЕНЕТО

А В Т О Р Е Ф Е Р А Т

на дисертация за придобиване на образователна и научна степен

"ДОКТОР"

Област: 5. Технически науки

Професионално направление: Машинно инженерство

Научна специалност: Металорежещи машини и системи

Научни ръководители: проф. д.т.н. Георги Попов
проф. д-р Идилия Бачкова

СОФИЯ, 2018 г.

Дисертационният труд е обсъден и насочен за защита от Катедрения съвет на катедра „Технология на машиностроенето и металорежещи машини“ към Машинно-технологичен факултет на ТУ-София на редовно заседание, проведено на 30.04.18 г.

Публичната защита на дисертационния труд ще се състои на 10.09.18 г. от 15:00 часа в Конферентната зала на БИЦ на Технически университет – София на открито заседание на научното жури, определено със заповед № ОЖ-168 / 21.05.2018 г. и заповед № ОЖ-187 / 05.06.2018 г. на Ректора на ТУ-София в състав:

1. проф. д.т.н. Сашо Гергов – председател
2. проф. д.т.н. Георги Попов – научен секретар
3. проф. д.т.н. Стоян Стоянов
4. проф. д-р Жозеф Теллалян
5. проф. д-р Коста Бошнаков

Рецензенти:

1. проф. д.т.н. Сашо Гергов
2. проф. д.т.н. Стоян Стоянов

Материалите по защитата са на разположение на интересуващите се в канцеларията на Машинно-технологичен факултет на ТУ-София, блок № 3, кабинет № 3230.

Дисертантът е редовен докторант към катедра „Технология на машиностроенето и металорежещи машини“ на Машинно-технологичен факултет. Изследванията по дисертационната разработка са направени от автора, като някои от тях са подкрепени от научноизследователски проекти.

Автор: маг. инж. Костадин Стоянов
Заглавие: Разработване на подход за оперативна съвместимост в реконфигуриращи се производствени системи за машиностроенето
Тираж: 30 броя
Отпечатано в ИПК на Технически университет – София

I. ОБЩА ХАРАКТЕРИСТИКА НА ДИСЕРТАЦИОННИЯ ТРУД

Актуалност на проблема

Постоянното нарастване на технологичното знание повишава комплексността на производствените процеси и едновременно с това повишава в значителна степен сложността на средствата за производство и обслужващите ги информационни и управляващи системи. Липсата на оперативна съвместимост между информационните и управляващи системи на предприятията, която е един от основните движещи принципи на Индустрия 4.0, е един от основните софтуерни проблеми и очевидна пречка както за ефективната работа на предприятията, така и за постигането на висока степен на гъвкавост в случаите на налагаща се реконфигурация. Този основен проблем води до употребата на твърде много устройства и внедряване на различни софтуерни решения с цел да се осигури връзка и правилен обмен на информацията между отделните устройства и машини в системите. Това води до допълнително увеличаване на времето за производство и крайната цена на готовия продукт.

Цел на дисертационния труд, основни задачи и методи за изследване

На базата на определените в първа глава изводи са формулирани следните цел и задачи на дисертационната работа:

Цел: Разработване на подход за оперативна съвместимост на РПС в машиностроенето.

Задачи:

1. Да се разработят онтологични модели за РПС;
2. Да се разработи софтуер за генериране на управляващи програми базирани на стандарта IEC 61499 чрез употребата на онтологии;
3. Да се апробира подхода за генериране на управляващи програми върху комплекс работни станции FESTO;
4. Да се обобщи подхода за оперативна съвместимост в РПС за машиностроенето.

Научна новост

Предложен е подход за осъществяване на оперативна съвместимост на РПС в машиностроенето чрез употребата на онтологии и Java десктоп приложение за генериране на управляващи програми, базирани на стандарта IEC 61499.

Предложени са пет домейн онтологии („Екипировка“, „Материали“, „Процесни сегменти“, „Продуктови дефиниции“, „Персонал“) базирани на стандарта IEC/ISO 62264 и отговарящи на едноименните мета класове от мета онтология базирана на същият стандарт. Интеграцията на тези онтологии с мета онтологията създава комплексна система за управление на знанията, която може да се използва за разрешаване на различни по сложност и обхват производствени и бизнес задачи.

Практическа приложимост

Разработените онтологии отговарят на езика OWL2 DL и са верифицирани чрез различни системи за логически анализ и извод. Всяка една от тях може да се използва и самостоятелно като база от знания за съответната област.

Разработено е десктоп софтуерно приложение за генериране на управляващи програми базирани на стандарта IEC 61499. Софтуерът позволява работа както с локални онтологични файлове, така и с файлове, локализирани в интернет пространството, а също

така предоставя и две допълнителни възможности за типа на генерираните управляващи програми - IEC 61131 и CNC. Софтуера предоставя възможност за генериране на управляващи програми, базирани на стандарта IEC 61499 чрез разработена статична библиотека, която позволява употребата неговата употреба и при липса на онтологичен файл.

Генерираните управляващи програми, базирани на стандарта IEC 61499 за комплекса работни станции на фирма FESTO, са внедрени в лекционния материал на следните специалности:

- „Компютърно проектиране и технологии в машиностроенето" към МТФ на ТУ София за образователно-квалификационна степен бакалавър и магистър.
- „Информационни технологии в индустрията" към ФКСТ на ТУ София за образователно-квалификационна степен бакалавър.

Апробация

Апробацията на резултатите е извършена върху комплекса работни станции на фирма FESTO (Обслужваща работна станция, Обработваща работна станция и Сортираща работна станция) собственост на катедра „Технология на машиностроенето и металорежещи машини".

Публикации

Основните постижения и резултати от дисертационния труд са публикувани в 7 на брой научни статии докладвани на следните конференции и семинари:

- Eighth international conference of on Electrical and Control Technologies ECT – 2013, Kaunas, Lithuania, 2013;
- 8^{ма} международна научна конференция за млади изследователи, Варна, България 2014;
- 28^{ма} международна научна конференция на Машинно-технологичен факултет на ТУ - София, Созопол, България, 2015;
- International congress - Machines. Technologies. Materials., Borovets, Bulgaria, 2016;
- САИ XXIV международен симпозиум, Банкя, България, 2016.

Структура и обем на дисертационния труд

Дисертационният труд е в обем от 148 страници, като включва увод, 4 глави за решаване на формулираните основни задачи, списък на основните приноси, списък на публикациите по дисертацията и използвана литература. Цитирани са общо 171 литературни източници, като 139 са на латиница и 32 на кирилица, а останалите са интернет адреси. Работата включва общо 83 фигури и 7 таблици. Номерата на фигурите и таблиците в автореферата съответстват на тези в дисертационния труд.

II. СЪДЪРЖАНИЕ НА ДИСЕРТАЦИОННИЯ ТРУД

ГЛАВА 1. АНАЛИЗ НА СЪВРЕМЕННОТО СЪСТОЯНИЕ НА РЕКОНФИГУРИРАЩИТЕ СЕ ПРОИЗВОДСТВЕНИ СИСТЕМИ И ПРОБЛЕМА ЗА ОПЕРАТИВНА СЪВМЕСТИМОСТ

На базата на направения литературен обзор, свързан с проблемите при постигането на оперативна съвместимост в областта на реконфигуриращите се производствени системи и методите за нейното постигане, могат да бъдат направени следните изводи и заключения:

1. Към настоящия момент не съществуват разработени стандарти, които да подпомогнат оперативната съвместимост в РПС и да облекчат работа на проектантите;
2. Съществуват редица проблеми, касаещи проектирането на софтуер и хардуер за РПС;
3. Налице е необходимост от разработване на концепция за постигането на оперативна съвместимост в РПС;
4. Постигането на оперативна съвместимост дава възможност за намаляване броя на различните интерфейси, които са необходими за интегрирането на нови системи;
5. Направен е анализ на средствата, чрез които оперативната съвместимост в РПС би била постижима;
6. Съществува разнообразие от стандарти, които са насочени към различни аспекти на оперативната съвместимост, но в областта на производството добра основа за развитие на производствени системи имат стандартите IEC 62264, ISO 10303 (STEP) и IEC 61499;
7. Онтологиите са в основата на многократно използваемите бази знания и могат да бъдат успешно използвани за различни видове оперативна съвместимост поради тяхната висока изразителност и съществуващите системи за извършване на логически анализ;
8. Езиците за разработване на онтологии са много и се класифицират по различни признаци, но най-предпочитан е OWL-DL;
9. В областта на РПС не е известен подход за постигане на оперативна съвместимост;
10. Употребата на онтологии подпомага разработването на подходи за оперативна съвместимост в областта на РПС;

ГЛАВА 2. РАЗРАБОТВАНЕ НА ОНТОЛОГИЧНИ МОДЕЛИ ЗА РЕКОНФИГУРИРАЩИ СЕ ПРОИЗВОДСТВЕНИ СИСТЕМИ

2.1. Мета онтология базирана на стандарта ISO/IEC 62264

На базата на основните идеи, принципи, дефиниции и модели, в съответствие с обхвата и функциите на стандарта ISO/IEC 62264 е създадена мета онтология за интеграция на информационно-управляващите системи в предприятията, която представлява рамка (скелет) за създаване и интегриране на различни по функции, цел и предназначение домейн онтологии. Чрез мета онтологията са формализирани най-общите аспекти на функциониране на предприятията, свързани с понятия, релации и основни правила. Мета онтологията позволява създаване, анализиране, изпълняване, проверка и

валидиране на модели, а също така и има възможност за осъществяване на проверка за взаимната съгласуваност на моделите.

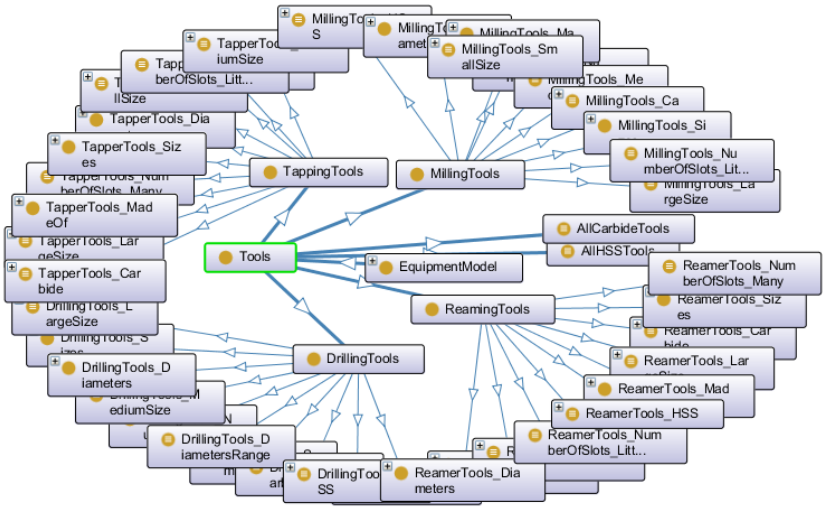
2.2. Домейн онтология „Екипировка“

2.2.1. Класове

Основният клас на тази онтология е класа „Enterprise“, а допълнителния клас „ValuePartitions“ се използва като шаблон при проектирането на домейн онтология-та. Класът „ValuePartitions“ съдържа покриващи аксиоми, които се използват за раз-лични класификации, сортиращи и сглобяващи операции, чрез употребата на сис-темата за логически анализ HermiT. Самият клас „Enterprise“ следва структурата, за-дадена за производствени предприятия в първа част на стандарта IEC/ISO 62264 и има следната йерархия от класове - „Enterprise“, „Site“, „Area“, „ProductionLine“, „WorkCell“, „EquipmentModule“ и „ControlModule“. Класът „ControlModule“ стои на най-ниското ниво, дефинирано от стандарта и съдържа класовете - „Sensor“ и „Actuator“. Класът „Sensor“ съдържа колекция от класове, свързани с различните типове сензо-ри. Физическите обекти (самите сензори) в класовете са представени чрез индиви-ди, чиито физически характеристики са описани с помощта на свойства тип данни и са присвоени към съответстващия им клас.

Класът *„EquipmentModule“* съдържа няколко допълнителни примитивни класа, освен класа „ControlModule“, които са *„Accessory“*, *„Module“* и *„Tools“*. Примитивният клас *„Accessory“* съдържа колекция от класове за различни аксесоари като стиски, предпазители, лампи и т.н., които биха могли да бъдат добавяни към различни машини. Наименуваният клас *„Module“* съдържа класовете *„FESTOModule“* и *„CNCMachineTool_Module“*, където първият съдържа колекция от класове, отговаряща на различните модули, които са необходими за изграждането на обработващата, обслужващата и сортиращата работни станции на фирма FESTO. Подобно на сензорите, всеки модул е представен чрез индивиди, а физическите характеристики са описани чрез свойства тип данни. Наименуваният клас *„Tools“* съдържа колекция от примитивни и дефинирани класове, които притежават ограничения и могат да класифицират различните индивиди по различни признаци. Например, дефинираните класове *„AllCarbideTool“* и *„AllHSSTool“* сортират всички инструменти, чиито материал е от карбид или бързорезна стомана и имат супер клас *„Tools“*, когато онтологията бъде проверена за консистентност от системите за логически анализ и извод. Всеки един от инструментите в онтология е описан по начин, който е сходен с този при сензорите и модулите. Другите под класове на супер класа *„Tools“* са *„DrillingTool“*, *„MillingTool“*, *„ReamingTool“* и *„TappingTool“*. Всеки един от тези класове съдържа индивиди, които описват различните по тип инструменти и са разпределени към съответният им клас. Чрез употребата на дефинирани подкласове, онтологията има възможността да извършва различни сортиращи операции в зависимост от материала на инструмента, предварително зададен диаметър на инструмента или габаритните размери на самият инструмент (малък, среден и голям). Например, наименуваният клас *„DrillingTool“* съдържа наименуваните класове *„DrillingTool_Diameters“*, *„DrillingTool_MadeOf“* и *„DrillingTool_Sizes“* и един дефиниран клас, който разпределя всички пробивни инструменти спрямо предварително зададен диапазон на диаметъра на пробивните инструменти - *„DrillingTool_DiametersRange“*. Класът *„DrillingTool_Diameters“* съдържа наименувани класове за различните диаметри на свредлата и всеки един от индивидите е разпределен в съответстващия му клас, кореспондиращ на собствения му диаметър. Например, примитивният клас *„DrillingTool_Fi_1.6“* съдържа всички свредла от онтологията, които притежават диаметър равен на 1.6 [mm]. Целта на тази класификация е да създаде контейнер, в който и индивидите да могат да бъдат разпределени на базата на някои от техните основни

свойства, като в този случай това е диаметъра на инструмента (за цилиндрични инструменти). След като съществуват примитивни класове, които да съдържат създаваните индивиди, е добра практика да се премине към създаването на дефинирани класове чрез които да се извършват различните сортиращи операции спрямо свойствата на индивидите и нуждите на онтологията, тъй като е възможно един индивид да притежава десетки свойства и извършването на подобен тип операции ръчно е неефективно и излишно. Примитивният клас *„DrillingTool_MadeOf“* съдържа два дефинирани класа (*„DrillingTool_Carbide“* и *„DrillingTool_HSS“*), които могат да преразпределят свредлата спрямо материала им. Последният пряк подклас - *„DrillingTool_Sizes“* - на супер класа *„DrillingTool“*, преразпределя примитивните класове за диаметъра на инструментите в по-широк смисъл, като ги разделя на групи (малки, средни и големи) според предварително зададен диапазон на диаметъра за всяка група. Класовете и операциите, които се извършват за останалите типове инструменти са аналогични на описаните за наименувания клас *„DrillingTool“*.



Фиг. 2.2: Йерархия на класа „Tools“

На фиг. 2.2 е представена йерархията на класа *„Tools“*. На йерархичното ниво на примитивния клас *„WorkCell“* е възможно да се създават групи от примитивни и дефинирани класове, които чрез употребата на обектни свойства и свойства тип данни могат да извличат информация от по-ниските йерархични нива, и да свързват индивидите в машини (подробен пример е представен в точка 2.2.2). От това йерархично ниво към супер класовете (*„ProductionLine“*, *„Area“*, *„Site“*) на наименувания клас *„WorkCell“* е по-добре да се използват покриващи аксиоми за създаването на различни конфигурации от производствени линии и работни зони.

2.3. Домейн онтология „Материали“

2.3.4. Ограничения

За наложените ограничения в онтологията е представен пример с клас *„AluminiumAlloys“*. Този клас съдържа осем дефинирани класа и съдържа всички индивиди, които описват алуминиевите сплави въведени в онтологията. Един от дефинираните класове е класът *„AluminumAlloys_CoefficientOfCrossContraction“* с наложено екзистенциално ограничение (5). Това означава, че този клас съдържа всички индивиди, които са алуминиеви сплави и имат коефициент на напречно свиване точно 0,34. Друг клас, който е с наложено ограничение (6) е *„AluminumAlloys_HighDensity“*, а класът *„AluminiumAlloys_HighSpecificHeatCapacity“* има наложено ограничение (7).

AluminiumAlloys
and *(hasCoefficientOfCrossContractionValue* ***value*** *0.34)* (5)

AluminiumAlloys
and *(hasDensityValue* ***some*** *xsd:double [>= 2650.0])* (6)

AluminiumAlloys
and *(hasDensityValue* ***some*** *xsd:double [>= 2650.0])*
and *(hasHighSpecificHeatCapacityValue* ***some*** *xsd:integer [>= 950])* (7)

Класът „*AluminumAlloys_HighDensity*“ съдържа всички алуминиеви сплави в онтологията, които притежават плътност по-голяма или равна на 2650 [kg/m^3], а вторият клас - *AluminiumAlloys_HighSpecificHeatCapacity* добавя към това ограничени второ, което гласи, че за да бъде даден индивид член на този клас той трябва да е алуминиева сплав, да притежава плътност по-голяма или равна на 2650 [kg/m^3] и да притежава специфичен топлинен капацитет по-голям или равен на 950 [J/kgK]. Поради тази причина, когато се приложат системите за логически анализ, класът „*AluminiumAlloys_HighSpecificHeatCapacity*“ става подклас на класа „*AluminumAlloys_HighDensity*“. Ако се добави ново ограничение към вече съществуващите и тази нова група от ограничения бъде наложена към друг клас, тогава въпросният клас се явява под клас на класа *AluminiumAlloys_HighSpecificHeatCapacity* и индивидите трябва да преминат през допълнително класифициране. След като онтологията бъде подложена на логически анализ се създава йерархия с осем поднива на клас „*Final*“.

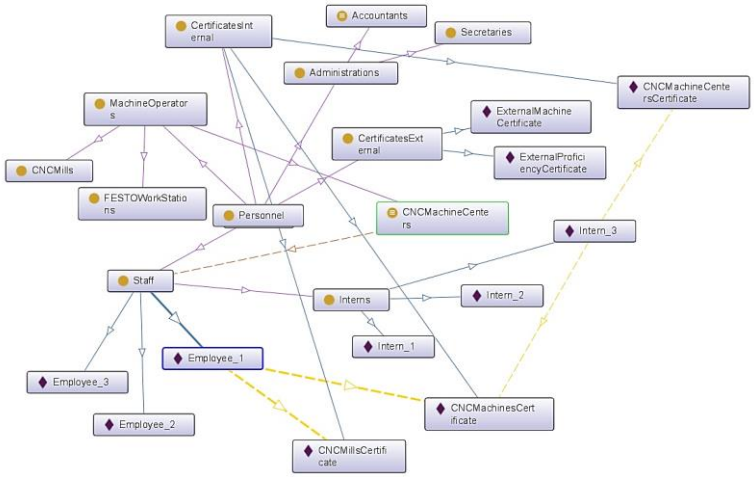
2.4. Домейн онтология „Продуктови дефиниции“

В домейн онтологията „Продуктови дефиниции“ се описват детайлите за производство. За разлика от домейн онтологиите описани в т. 2.2 и т. 2.3, където описанието на конкретен обект се осъществява чрез употребата на индивиди, то в настоящата домейн онтология описанието на конкретен детайл се извършва на ниво клас. Описанието на даден детайл на ниво индивид би довело до редица проблеми като някои от тях са увеличаване на времето за въвеждане на информацията, увеличаване възможността за грешка, значително увеличаване на обектните свойства и свойствата тип данни, което от своя страна би довело до увеличаване на взаимоотношенията между отделните индивиди, а също така и увеличаване на времето, необходимо на системите за логически анализ да извлекат знанието от наложените връзки. На фиг. 2.12 са представени 3D изгледи на тестовите детайли на фирма FESTO за комплекса работни станции. Детайлите се различават по цвят и материал като тези разлики се отчитат от сензорите на самите работни станции. Комплексът работни станции, маршрутния технологичен процес и управляващите програми генерирани от разработения софтуер (представен в глава 3) са представени подробно в глава 4.

а) черен *б) червен* *в) сребърен*

Фиг. 2.12: 3D изгледи на тестови детайли на фирма FESTO

2.5. Домейн онтология „Персонал“

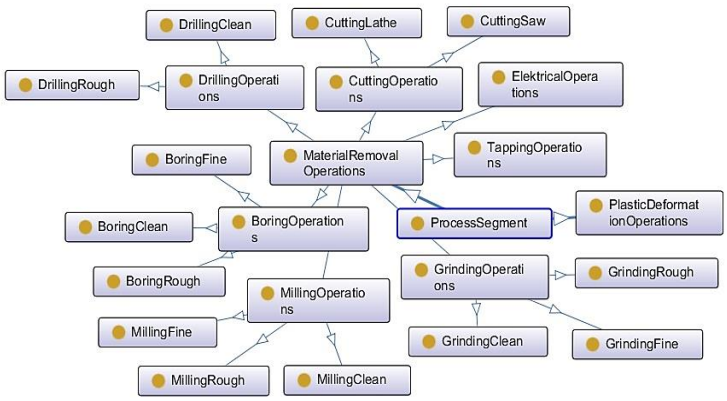


Фиг. 2.14: Придобити сертификати в домейн онтологията „Персонал“

Домейн онтология „Персонал“ структурира информация за наетите служители, структурата на персонала и квалификацията на служителите, като задава някои основни свойства и класове, които да илюстрират взаимоотношенията между отделните служители. Самите служители са описани на ниво индивиди, като за целта са разработени групи от свойства тип данни, чрез които да бъде извършено описанието на съответния индивид. Например, групата на личностните свойства съдържа свойства, чрез които се описва персонална информация за индивида като лично и фамилно име, телефонен номер, години и т.н. Това са свойства, чиито обхват е от примитивен тип - целочислено число, число с десетична запетая, буквен низ и т.н. Друга група от създадените за онтологията съдържа свойства, свързани с опи-та на съответния служител като образование, предишна заемана позиция, квалификация, трудов стаж, старшинство и т.н. Обхватът на тези свойства е дефиниран като ограничения от примитивен тип.

За настоящата онтология е разработена отделна група от свойства, която касае различни сертификати, отнасящи се до придобитата квалификация на даден служител. Самите сертификати са описват на ниво индивид и се разделят в два примитивни класа - единият от наименуваните класове съдържа сертификати, които са вътрешни за дадено предприятие, а вторият съдържа такива, които се явяват външни (обучението на служителя е извършено от друга фирма) за същото предприятие. Връзката между даден служител и определен сертификат се осъществява чрез обектни свойства и е представена на фиг. 2.14.

2.6. Домейн онтология „Процесни сегменти“



Фиг. 2.16: Класове от домейн онтология „Процесни сегменти“

В домейн онтологията „Процесни сегменти“ са създадени класове, чрез които да се опишат различните по вид и режими технологични операции. На фиг. 2.16 е представена йерархичната структура на класовете в онтологията, като двата основни примитивни класа са „*MaterialRemovalOperations*“ и „*PlasticDeformationOperations*“.

Примитивният клас „*MaterialRemuvalOperations*“ съдържа като подкласове някои от основните видове технологични операции като струговане, фрезоване, свредловане, шлифоване и т.н. От своя страна всеки един от тези

наименувани класове съдържа дефинирани подкласове, които допълнително да класифицират съответната технологична операция спрямо вида обработка - груба, чиста, полу-чиста и фина, а в някои случаи и супер-фина - в зависимост от наложените върху класовете ограничения. Самите технологични операции със съответното знание за режимите на рязане се описват на ниво индивиди.

Въпреки, че между самите индивиди в домейн онтологията „Процесни сегменти“ не са наложени ограничения или допълнителни връзки чрез обектни свойства, то за онтологията е създаден набор от такива. Тези обектни свойства подобно на групата от външни обектни свойства в домейн онтологията „Материали“ се използват след обединяването на отделните онтологии с цел представяне на взаимоотношенията *технологична операция > инструмент*.

2.7. Обединение на разработените домейн онтологии с мета онтологията

След като разработените онтологични модели достигнат определена комплексност и пълнота е необходимо те да бъдат обединени с мета онтологията, представена в т. 2.1. От една страна обединението с мета онтологията е необходимо, т.к. разработените онтологии не могат да обхванат всички взаимоотношения между отделните индивиди, когато те работят самостоятелно, което би довело до грешки и неправилно или непълно извлечено знание. Обединението с мета онтологията позволява да се сведат тези грешки до минимум, да се верифицират самите онтологии, а също така и да се създадат по-добре формулирани взаимоотношения между отделните индивиди, връзки между отделните класове и мета класове и по-добро структурира-не на йерархията на онтологията, което би довело до намаляване на времето, необходимо на системите за логически анализ да направят класификация и да извлекат необходимото знание. Благодарение на обединяването с мета онтологията е възможно прилагането на основните правила, понятия и релации, заложени в стандарта ISO/IEC 62264.

От друга страна, обединението на разработените домейн онтологии с мета онтологията е необходимо заради значително по-малкия брой заети ресурси от разработения софтуер за генериране на управляващи програми, базирани на стандарта IEC 61499. Преди да бъдат обединени онтологиите е необходимо да бъде направена проверка за имената на схемите. Ако такива липсват е препоръчително да бъдат добавени, т.к. липсата им би довела до припокриване на свойства, класове и индивиди, което в последствие би довело до неконсистентност на онтологията или до грешно направени изводи. Препоръчително е, след обединяването на онтологиите, всички бъдещи промени да бъдат правени във вече обединената онтология, като това би спомогнало за намаляване на грешките в процеса на бъдещо развитие, кое-то намалява и цялостното време за разработване на нова версия на обединената онтология.

2.8. Изводи към втора глава

На базата на разработените онтологични модели за реконфигуриращи се производствени системи могат да бъдат направени следните изводи и заключения:

1. Основната насока на разработените онтологични модели е обхващането на основните производствени процеси за решаване на различни реконфигурационни задачи с цел решаване на проблемите на оперативна съвместимост в РПС;
2. Домейн онтологиите „Екипировка“, „Материали“, „Продуктови дефиниции“, „Персонал“ и „Процесни сегменти“ са проектирани за интеграция в рамката на представената мета онтология, базирана на стандарта IEC/ISO 62264, за да могат да работят съвместно с цел моделиране на комплексна система за управление на

знанията, както и за разрешаване на различни по сложност и обхват производствени и бизнес задачи;

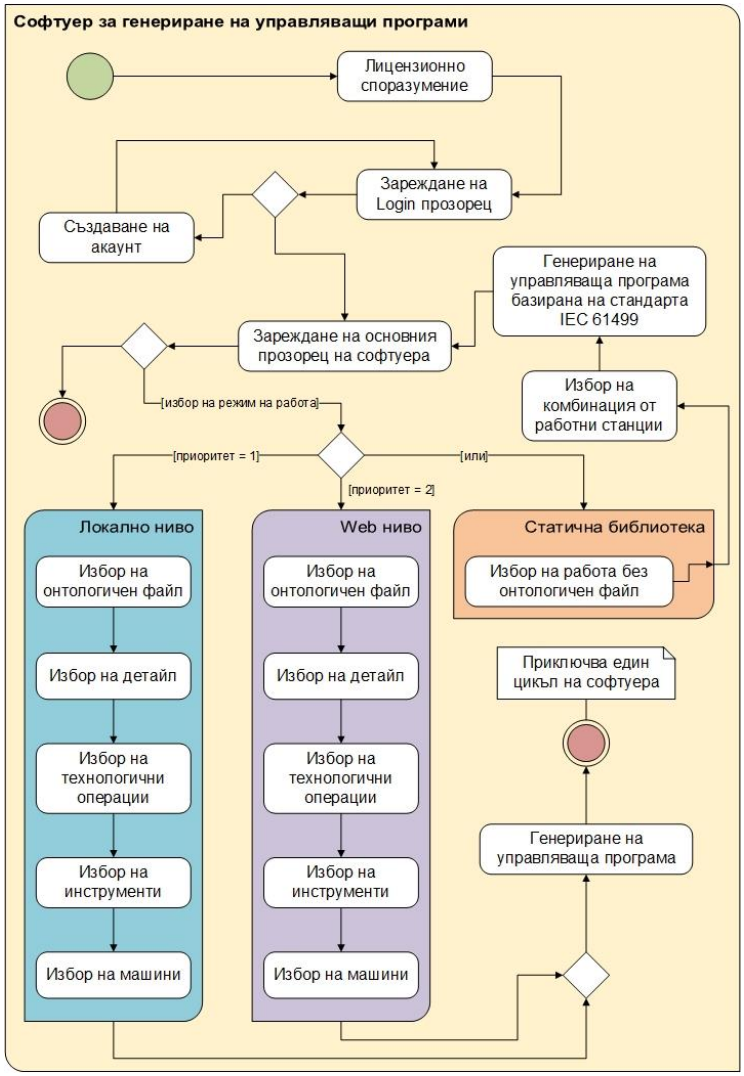
3. Разработените онтологии се придържат към езика OWL2 DL, а за тяхното разработване е използвана най-популярната свободно достъпна среда за създаване и редактиране на онтологии Protégé, съвместно със системите за логически анализ и извод HermiT, Pallet и FaCT++;
4. Чрез употребата на свойства, покриващи аксиоми и ограничения е възможно не само създаването и свързването на индивиди и класове в машини и системи, но също така е възможна и тяхната реконфигурация;
5. Разработените онтологии „Продуктови дефиниции“ и „Процесни сегменти“ предлагат гъвкаво решение за въвеждане на информацията за отделните детайли и последователността на извършваните технологични операции, като представят методология за изграждането на подредени списъци - концепция, за която няма разработени стандартни конструктори в езика OWL - като в същото време запазват изразителността на езика в рамките на OWL DL;
6. Всяка една от домейн онтологиите може да бъде използвана и като самостоятелни база от знания в областта на бизнеса и производствените системи, тъй като е възможно допълнителното им и бързо разширяване с нови данни, класове и свойства;
7. Предложени са насоки за обединението на домейн онтологиите с мета онтологията, а също така и насоки относно процеса по бъдещо развитие на обединената онтология.

ГЛАВА 3. РАЗРАБОТВАНЕ НА СОФТУЕР ЗА ГЕНЕРИРАНЕ НА УПРАВЛЯВАЩИ ПРОГРАМИ БАЗИРАНИ, НА СТАНДАРТА IEC 61499 ЧРЕЗ УПОТРЕБАТА НА ОНТОЛОГИИ

3.1. Общ преглед на разработваният софтуер

Основната цел на разработения софтуер е да генерира управляващи програми, базирани на стандарта IEC 61499. Необходимата за целта информация се извлича от разработените домейн онтологии, които са обединени с висшата онтология (всички онтологии са представени подробно във втора глава). На фиг. 3.1 е представена UML диаграма на дейността на разработения софтуер.

След като OntoMS бъде стартиран се зарежда графичния потребителски интерфейс, който представя общите условия за използване на софтуера. Потребителят не може да продължи докато не даде своето съгласие, касаещо правилата за употреба на OntoMS. След като потребителят се съгласи с общите условия, OntoMS зарежда потребителския интерфейс за вход в системата, който се явява и базова защита срещу непозволен достъп до функционалността на OntoMS. За целите на дисертационния труд е създадена възможност за създаване на потребителски профил, чрез който може да се влезе в системата. За удобство тази функционалност е имплементирана в допълнителен бутон, чрез който се отваря бланка за въвеждане на необходимата информация. При наличие на потребителски профил, и потребителят въведе коректното потребителско име и парола, която да отговаря на името в съответните полета за вход в системата, то на екрана се визуализира основният прозорец на разработения софтуер.



Фиг. 3.1: UML диаграма на дейността на разработения софтуер

Основният прозорец съдържа няколко допълнителни функционалности като визуализиране на времето на системата до секунда, представяне на основната информация на всички регистрирани потребители в табличен вид, а също така и подробна такава под формата на визитка. Целта за имплементация на тези функционалности е внедряването на базова административна дейност и улеснена комуникация между отделните потребители. Относно генерирането на управляващи програми потребителят има възможност да избере между употреба на локален онтологичен файл, да зареди онтологичен файл, намиращ се в интернет пространството или при липса на такъв да използва статичната библиотека на OntoMS. При избор на някой от първите два варианта на потребителя се предоставя възможност да избере желания детайл, за който да бъде генерирана управляваща програма, а също така и необходимите технологични операции, инструменти и машини. При избор от страна на потребителя за употреба на статичната библиотека се визуализира потребителски интерфейс, който предоставя възможност за избор на комбинация от работни станции, за които да бъдат генерирани управляващи програми. Генерирането на управляващи програми бележи краят на един работен цикъл от изпълнението на софтуера, след което потребителят бива препратен отново към основният прозорец на OntoMS.

3.2. Интерфейс за лицензионно споразумение

Тъй като OntoMS притежава функционалността да бъде изпълняван самостоятелно извън границите на IDE средата, в която е разработван, то след стартирането на изпълнимия .jar файл се визуализира лицензионното споразумение. Това, което оста-ва скрито от потребителя и се изпълнява на заден план, е подготовката на работно-то пространство, като за целта OntoMS прави проверка за състоянието на текущата директория. Ако необходимите файлове и директории не бъдат открити, то те биват създадени (препоръчително е .jar файла да се изпълнява в самостоятелна директо-рия). Файловете, които биват създадени са файлове за съхранение на потребителс-ките профили и файлове за регистрация на грешки и неочаквано поведение от страна на OntoMS. Създадените директории включват директории за съхранение на графични изображения, а също така и базовата директория за запис на генерираните управляващи програми.

В процеса на създаване на работните файлове и директории, OntoMS прави обръщение към системата от която е стартиран и променя визуализацията си спрямо стандартния графичен интерфейс на съответната система. При съгласие от страна на потребителя с всички лицензионни споразумения се активира бутона за прехвърляне към интерфейса за валидация на самоличността, чрез който се осигурява достъп до вътрешните функции на OntoMS.

3.3. Интерфейс за валидиране самоличността на потребителя и достъп до системата

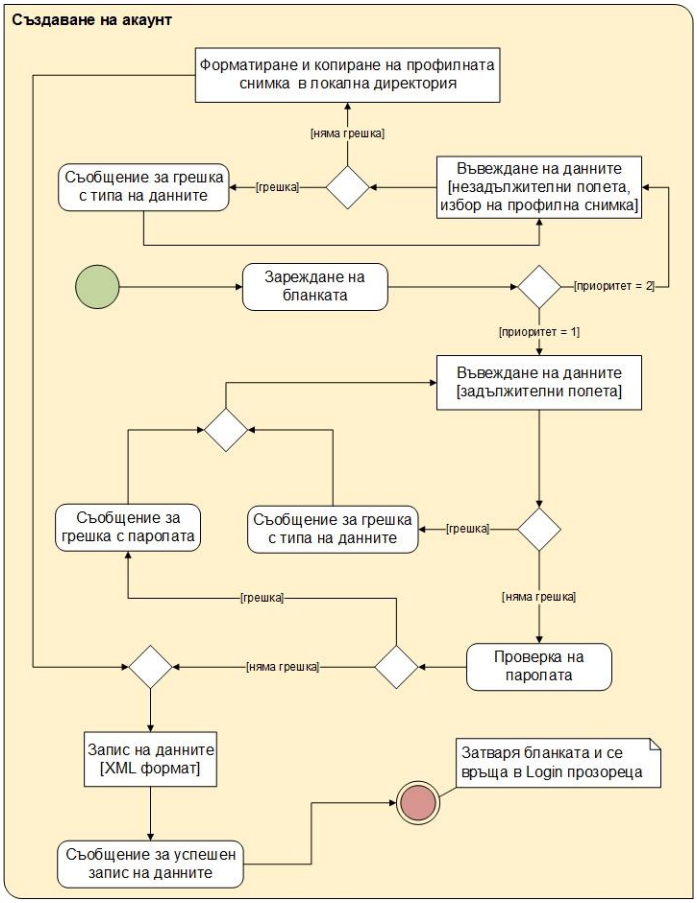
След като бъде зареден интерфейсът за вход към системата, потребителят има възможност да въведе своето потребителско име и парола. При сигнал от извършената проверка, че има несъответствие между име и парола, системата визуализира съобщение за грешка, в противен случай потребителят е допуснат до вътрешните функции на OntoMS. При липса на потребителско име и парола потребителят има възможност да създаде нов профил чрез натискането на съответния бутон.

Потребителското име и парола се валидират, като от съответните полета се извлекат въведените низове и се запазят програмно в локални променливи, след което се локализира файлът, който съдържа потребителските имена и пароли. Програмата извиква метод, който чрез XML трансформации извлича и попълва необходимата информацията от файла като ключ-стойност полета в HashMap структура от данни. След като целият файл бъде изчетен, методът връща попълнената вече структура от данни за извършване на финалните проверки от OntoMS. Ако въведеното от потребителя име не се съдържа във върнатата структура, OntoMS визуализира съобщение за грешно въведено потребителско име. Ако потребителското име се съдържа в извлечените данни, но паролата не отговаря, то тогава OntoMS визуализира съобщение за грешно въведена парола. Ако и двете проверки върнат положителен резултат, то тогава OntoMS допуска потребителя до своите вътрешни функции и зарежда основния прозорец.

3.4. Интерфейс за създаване на потребителски профил

При липса на потребителски профил, потребителят има възможност да създаде такъв, като натисне съответния бутон от потребителския интерфейс. След като бланката за въвеждане на необходимите данни бъде заредена, потребителят има възможност да въведе всички данни или само задължителните такива (отбелязани със знака *). При липса на въведена информация в някое от задължителните полета системата извежда съобщение за грешка и променя цвета на името на съответното текстово поле в червен. След като бъдат попълнени всички задължителни полета OntoMS извършва проверка на въведената информация. Ако съществува несъответствие между въведената информация и шаблона, на който трябва да отговарят данните от съответното текстово поле, то тогава OntoMS отново изкарва съобщение за грешка, като в съобщението се дават и насоки за нейното отстраняване. Полето, отговарящо за въвеждането на потребителско име, притежава и допълнителна проверка за дублиране на името с вече съществуващи такива. След като всички задължителни полета са проверени се извършва проверка за силата на зададената парола. При успешно преминаване на тази проверка OntoMS сравнява въведената парола с тази, която е въведена в полето за повтаряне на желаната парола. Ако има съответствие на стойностите на двете полета, въведените данни се записват в XML документ, който съхранява потребителската информация, а OntoMS извежда съобщение за успешния запис на данните и затваря бланката, като в същото време зарежда отново интерфейса за валидация на потребителя.

От полетата, които не са задължителни за създаването на потребителски профил, по-специално внимание заслужава полето за избор на потребителска снимка. След като бъде избрана такава от локалното дисково пространство на компютъра, OntoMS форматира снимката до предварително фиксирани размери и копира съдържанието на файла в съответната работна директория, служеща за съхранение на всички потребителски профилни изображения. При успешен запис на въведената информация, в XML документа се записва пътя до съответното графично изображение. Преди да се форматира и копира графичното изображение OntoMS извършва допълнителна проверка за дублиране на името на файла с цел да се избегне презаписване на файла, което от своя страна би довело до загуба на лична информация и дублиращи се изображения. Ако направената проверката даде положителен резултат, OntoMS изкарва съобщение за наличие на файл със същото име. Шаблоните, чрез които се проверяват въведените данни се задават чрез употребата на регулярни изрази. На фиг. 3.5 е представена UML диаграма на дейността за създаване на потребителски профил.



Фиг. 3.5: UML диаграма на дейността за създаване на потребителски профил

Алгоритъмът за проверка на данните от бланката бива задействан при всяко едно натискане на бутона за запис на данните. На първо място се задават шаблоните, чрез които се извършват проверките на данните от всяко едно поле. За да могат шаблоните да бъдат използвани от OntoMS като регулярни изрази, е необходимо да бъдат компилирани като такива. След като компилацията на шаблоните бъде завършена, OntoMS взима низа от всяко едно поле и го сравнява със съответващия му регулярен израз. При наличие на празно поле, което е маркирано като задължително или данните от полето не отговарят на зададения шаблон, изпълнението на проверката за всички останали полета се преустановява, а OntoMS визуализира съответното текстово съобщение за грешка и маркира името на полето в червен цвят с цел по-добра визуализация за потребителя. При успешно отстраняване на проблемите във въвежданата информация името на полето отново се задава с нормален цвят.

След като успешно преминат всички проверки за коректност на въведените данни, OntoMS започва проверка за дублиращи се потребителски имена. За целта се създава нова структура от данни – HashSet. Чрез XML трансформации, приложени върху файла, съдържащ потребителските данни, се извличат всички потребителски имена и се попълват в новосъздадения сет, които се връща на програмата за извършване на финалната проверка. Ако проверката премине успешно, OntoMS преминава към проверка на паролата, в противен случай се визуализира съобщение за повтарящо се потребителско име и записа на данните във файла е прекъснат до отстраняване на проблема.

Проверката на въведената парола преминава по аналогичен начин, като в случая е необходимо да се отбележи, че паролата не се проверява по един, а по четири шаблона, които са навързани в булево логическо условие. Преминаването и на четирите проверки гарантира на потребителя създаването на профил, защитен със силна парола – паролата би била валидна само и единствено при условие, че съдържа поне една малка и една голяма буква, цифра и специален символ. В допълнение към четирите условия е добавено и пето, което задава минималната дължина на паролата да не бъде по малка от 8 символа.

Последната проверка, която се извършва от OntoMS преди да бъдат записани данните, е за валидиране на въведената парола, която ако бъде успешна, се извършва запис на потребителския профил във файла, а потребителят бива препратен към интерфейса за въвеждане на потребителско име и парола.

3.5. Интерфейс за избор на файл

След като потребителят бъде допуснат до вътрешните функции на системата, OntoMS зарежда интерфейса за избор на файл. Със зареждането на интерфейса се зарежда и наличната информация в обобщен, табличен вид за всички потребители. При избор на даден потребител от таблицата или чрез натискане на командните бутони се визуализира визитка с подробна информация и профилна снимка (при лиса на профилна снимка системата слага такава по подразбиране) за желания потребител.

Възможностите на потребителя за генериране на управляващи програми са три. Първата възможност е да се зареди локален онтологичен файл, от който да бъде извлечена необходимата информация. Втората възможност е да се зареди онтологичен файл, намиращ се в интернет пространството. Разликата между първата и втората възможност, е че при избор на локален файл OntoMS притежава възможността да манипулира данните в самата онтология, докато втората възможност позволява само четене и извличане на информация от онтологията. При липса на он-тологичен файл потребителя има възможност да използва статичната библиотека на OntoMS, като в този случай, всички налични командни бутони се блокират и се позволява на потребителя да продължи напред към интерфейса за избор на комби-нация от работни станции.

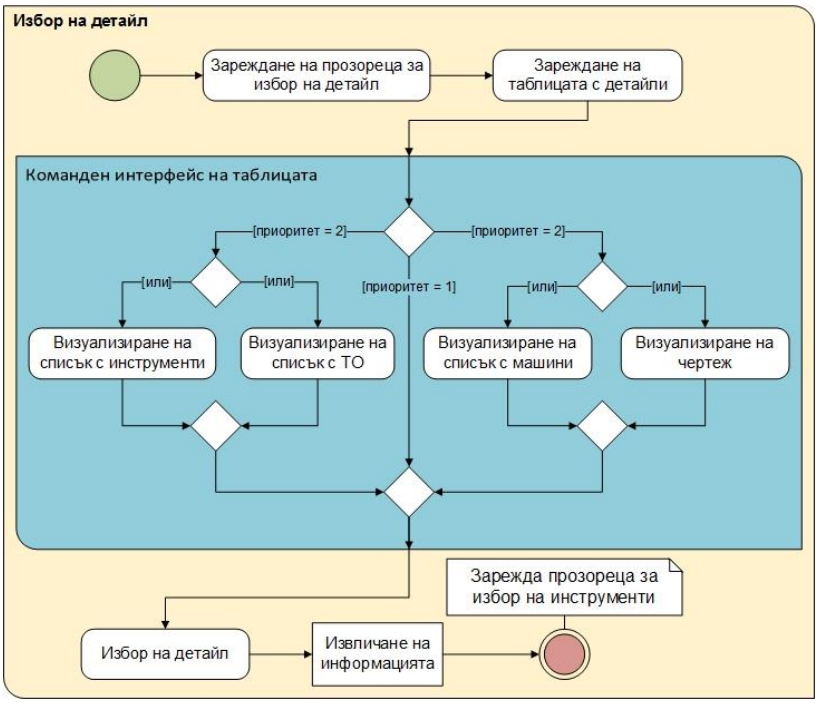
3.6. Интерфейс за комбинация от работни станции

При избор за употреба на статичните библиотеки OntoMS зарежда потребителския интерфейс за избор на комбинация от работни станции на фирма FESTO. Когато желаната комбинация бъде избрана, чрез натискане на командния бутон за генериране на управляващите програми се визуализира съобщение за избор на директория за запис на файловете. След като бъдат направени необходимите проверки за коректност на въведения файлов път, всички управляващи бутони от потребителският интерфейс биват блокирани от OntoMS с цел предотвратяване на неволен из-бор и последващо генериране на управляващи програми. Преди да бъдат генерирани самите програми, OntoMS проверява за съществуването на зададеният файлов път и ако такъв липсва, то той бива създаден, след което започва генерирането на самите ФБ за съответните работни станции. Когато индикатора за изпълнение достигане 100%, OntoMS визуализира съобщение за успешно генериране на файловете и възстановява командните бутони, след което връща потребителя в основния интерфейс за избор на файл.

3.7. Интерфейс за избор на детайл

При направен от потребителя избор за зареждане на локален онтологичен файл, OntoMS позволява на същия да продължи към интерфейса за избор на детайл, за който да бъде генерирана съответната управляваща програма. След като необходимият интерфейс

бъде зареден, OntoMS извлича необходимата информация от онтологичния файл и представя списъка с детайли в табличен вид. OntoMS позволява избор само на един детайл, по едно и също време.



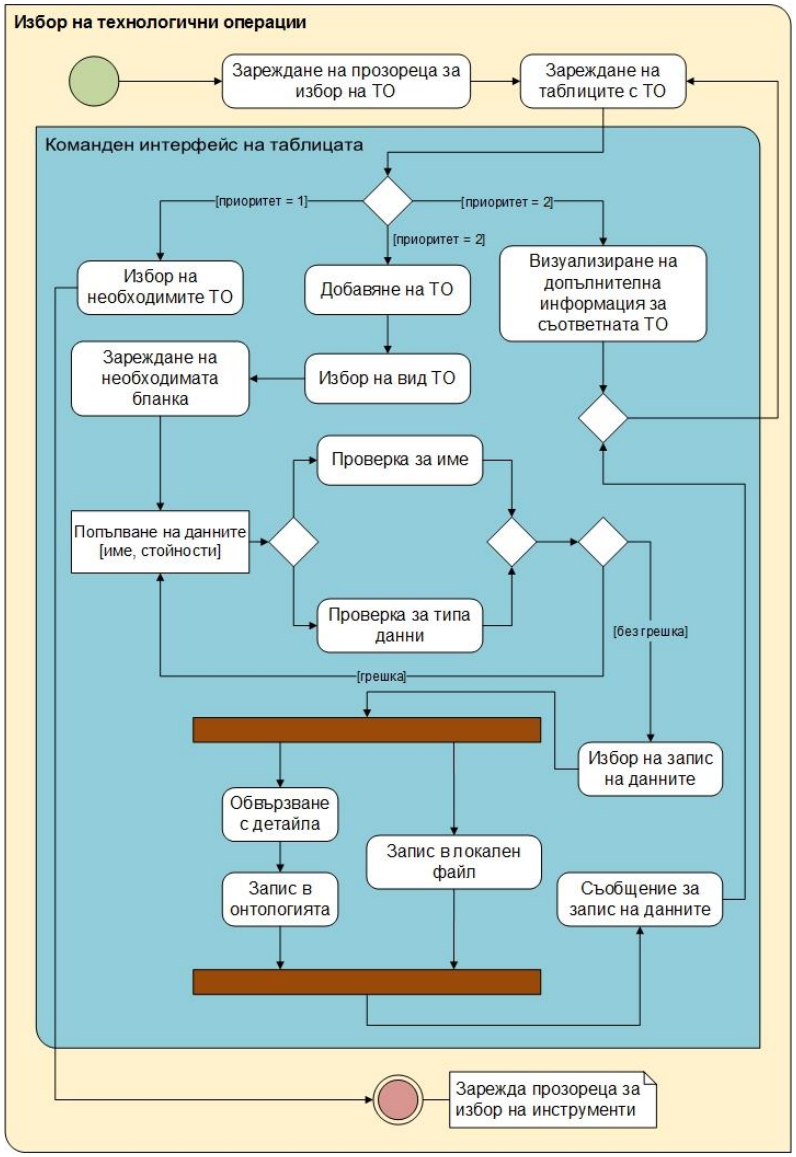
Фиг. 3.9: UML диаграма на дейността за избор на детайл

Ако потребителят желае да получи допълнителна информация относно използваните инструменти, машини или технологични операции, които са необходими за производството на конкретния детайл, то това е възможно чрез употребата на командните бутони на таблицата с детайли. При наличие на графично изображение, OntoMS прави опит да достигне до това изображение и ако е възможно да го форматира до необходимия размер, и копира в съответната локална директория за последваща употреба. Представяната от настоящия интерфейс допълнителна информация е в обобщен вид. Например, информацията относно инструментите за конкретен детайл, които се зареждат чрез командният бутон за инструменти, са представени единствено с имената си и тяхната необходима бройка. За да се получи подробна информация за конкретен инструмент е необходимо потребителят да премине към следващата стъпка, касаеща генерирането на управляващите програми. Когато желаният детайл бъде избран, OntoMS позволява на потребителя да премине към следващата конфигурационна стъпка - избор на инструменти. Базирайки се на вече извлечената информацията, която касае избрания детайл, OntoMS представя в отделните интерфейси като инструменти, машини и технологични операции само и единствено информация за избраният детайл. На фиг. 3.9 е представена UML диаграма на дейността на интерфейса за избор на детайл.

3.8. Интерфейс за избор на технологични операции

Когато интерфейсът за избор на технологични операции бъде зареден, основната информация бива представена в табличен вид. Потребителят има възможност да получи подробна информация за желана технологична операция като режими на рязане или времето, необходимо за изпълнението на съответната операция, чрез командния бутон или избор на самата технологичната операция от главната таблица. След като всички операции бъдат проверени, потребителят има възможност да премине към интерфейса за избор на машина, необходима за обработката на желания детайл. На фиг. 3.10 е представена UML диаграма на дейността за избор на технологични операции. За разлика от интерфейса за избор на детайл, където потребителят има възможност само и единствено да провери наличната информация, то в интерфейса за избор на технологични операции OntoMS позволява ръчното въвеждане на нови такива. За целта чрез съответният команден бутон се избира типа на желаната технологична операция. Веднъж избран, OntoMS зарежда

съответната бланка за попълване на необходимите данни. Когато всички данни бъдат попълнени в съответните текстови полета OntoMS извършва проверка за дублиране на името на технологичната операция и коректността на типа данни, а също така и тяхната стойност. Полетата със стойности от тип низ се проверяват с регулярни изрази. При откриване на несъответствие в текстовите полета спрямо зададените ограничения OntoMS визуализира съобщение за грешка.



Фиг. 3.10: UML диаграма на дейността за избор на технологични операции

След като всички ограничения бъдат спазени, е възможно записването на данните, като на потребителя бива предоставен избор за метода, по който да бъде осъществен записа. В първия случай потребителят има възможност да запише данните във външен файл, съхраняван в работната директория на OntoMS. Това означава, че технологичните операции, които се намират в този файл, се представят в спомагателна таблица като допълнителни технологични операции, които могат да бъдат извършвани независимо от избрания преди това детайл. Вторият вариант за запис на данните е чрез тяхното обвързване с конкретния детайл и манипулиране на самата онтология, като се създадат съответните индивиди и класове, и се наложат подходящите обектни свойства, свойства тип данни и ограничения. Когато информацията от новосъздадената технологична операция бъде записана, OntoMS визуализира информационно съобщение за успешен запис.

Интерфейсът за избор на инструменти има аналогична структура и действие, както интерфейса за избор на технологични операции, поради тази причина този интерфейс не е представен подробно в настоящия дисертационен труд. Необходимо е да се отбележи, че записаните локално инструменти се съхраняват в отделна директория и отделен файл от този на технологичните операции.

3.9. Интерфейс за избор на машина

Когато необходимите за обработването на желания детайл технологични операции и инструменти бъдат избрани, потребителят има възможност да премине към интерфейса

за избор на машина. Аналогично на описаните в предишните точки на настоящата глава интерфейси, машините, които могат да обработят желания детайл с необходимите за целта инструменти, са представени в табличен вид. В таб-лицата са представени единствено имената на наличните машини, но чрез командните бутони или чрез директен избор на съответната машина от таблицата, потребителят има възможност да визуализира общия вид на машината, а също така и техническите ѝ характеристики. След като желаната машина бъде избрана, OntoMS извлича необходимата информация от онтологичния файл и зарежда интерфейса за генериране на управляваща програма.

3.10. Интерфейс за генериране на управляваща програма

Настоящият интерфейс за генериране на управляващи програми се явява пресечна точка на двата възможни потока, чрез които се събира информацията, необходима за генериране на конкретна управляваща програма - чрез локален онтологичен файл или чрез онтологичен файл, намиращ се в интернет пространството.

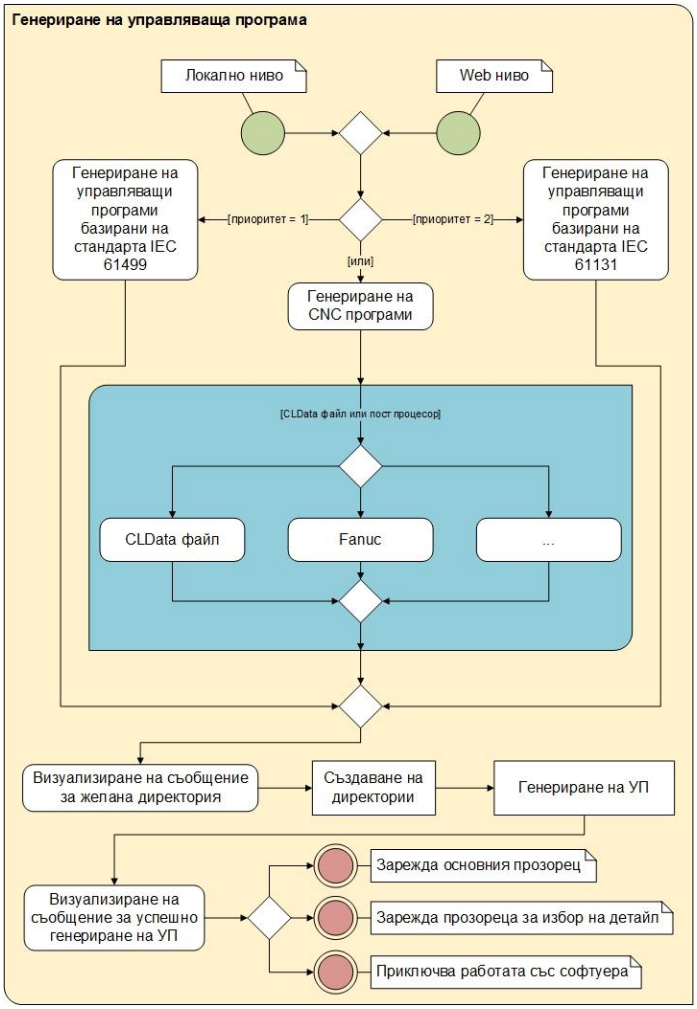
Когато интерфейсът бъде зареден, потребителят има възможност да избере типа на управляващата програма. Освен генерирането на управляващи програми, базирани на стандарта IEC 61499, OntoMS предлага възможност за генерирането на такава, базирана на стандарта IEC 61131 (процеса е аналогичен на IEC 61499) или генерирането на CNC програми. При избор за генериране на CNC програми потребителят може да избере желания формат. Въпреки, че е възможно да се генерира управляваща програма за конкретно CNC управление като Fanuc, HaidenHain, Siemens и др., то това би било некоректно и нефункционално без изричното съгласие и споразумение със съответният производител. Поради тази причина, настоящата функционалност на разработения софтуер е ориентирана към поддържането и генерирането на управляващи програми, базирани на CLData файлове и ISO код, които в последствие лесно могат да бъдат транслирани от съответния пост-процесор.

Когато типът за генерираната управляваща програма бъде избран, OntoMS визуализира съобщение за въвеждане на желания от потребителя файлов път, където да бъдат записани генерираните файлове. Когато бъдат направени всички необходими проверки за коректност на въведения файлов път, OntoMS проверява дали желаната директория съществува. Ако не съществува, директорията бива създадена, след което управляващите програми биват генерирани. Ако желаната директория вече съществува, OntoMS извършва проверка дали директорията е празна. При наличие на съществуващи файлове операцията по генериране на управляващите програми се прекратява и се визуализира съобщение за грешка. Когато управляващите програми бъдат генерирани, OntoMS визуализира съобщение за успешното създаване на необходимите файлове и предоставя на потребителя възможност за връщане към интерфейса за избор на онтологичен файл, интерфейса за избор на детайл или да излезе напълно от програмата. На фиг. 3.12 е представена UML диаграма на дейността за генериране на управляваща програма.

3.11. UML диаграми на методите от класа FileManager

Настоящата точка представя част от алгоритмите, които стоят в основата на методи-те на спомагателният клас FileManager от класовата йерархия на OntoMS. Това е базов, спомагателен клас, а представените алгоритми извършват голяма част от операциите с файлове и директории като създаване, изтриване, копиране, а също така трансформация и манипулация на данни. Необходимо е да се отбележи, че по-ради малките вариации между отделните алгоритми в настоящата точка са представени само трите най-базови. Първият от базовите алгоритми отговаря за създаването на вътрешните ресурси на OntoMS, т.е. отговаря най-вече за първоначално-то изграждане на работните

директории на OntoMS. Този алгоритъм, както и негови-те разновидности, се изпълняват при операции с файлове и директории като копиране, изтриване, извършване на проверки за съществуващи файлове, правилно зададени пътища до отделните ресурси на дисковото пространство, проверка на права за четене и писане и др.



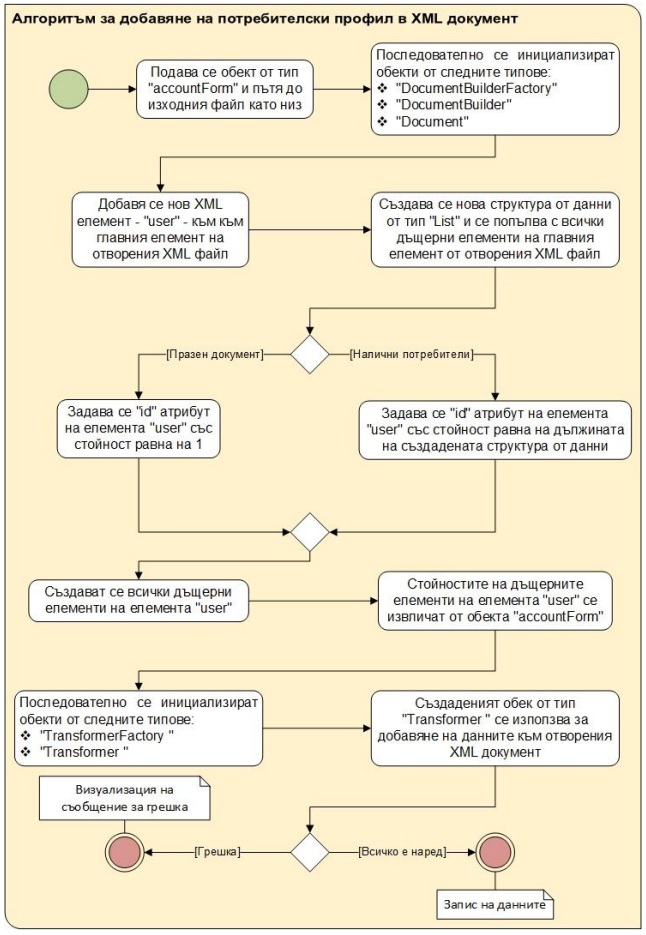
Фиг. 3.12: UML диаграма на дейността за генериране на управляваща програма

Първата стъпка от изпълнението на алгоритъма е да бъде зададен пътя до оригиналния файл, който е обект на по-нататъшни операции под формата на низ. Низът, се подава като входен параметър на конкретен метод и бива присвоен на вътрешна за метода променлива. Следващата стъпка от изпълнението на алгоритъма е създаването на буферирани входни и изходни потоци, които имат по-добра производителност при работа с твърдия диск на компютъра отколкото небуферираните такива. В процеса на създаване на потоците се създават и допълнителни променливи, които да служат като различни булеви флагове. Преди да започне прехвърлянето на информация от оригинални към изходния файл се извършва проверка за валидността на пътят до изходния файл както и дали може да бъде създаден. При наличие на грешка, операцията върху съответния ресурс се преустановява. В противен случай, пътя до изходния файл се подава на изходящия поток и започва прехвърлянето на информация от оригиналния файл. Копирането на информация-та е извършвано в цикъл, като условието за изход от цикъла е -1, което означава, че е достигнат края на оригиналния файл.

Вторият основен алгоритъм отговаря за добавянето на потребителските профили в XML документа. Този алгоритъм се изпълнява при всеки успешен запис на нов потребител във XML документа, като самият запис се извършва чрез съответните XML трансформации. За да започне изпълнението на алгоритъма е необходимо да се подаде обект от тип *accountForm* като входен параметър на обгръщащия го метод. По този начин цялата налична информация от бланката е достъпна за алгоритъма. За трансформацията на XML документа се използва DOM (Document Object Model).

Необходимо е да се отбележи, че при големи файлове DOM не подходящ избор за модификация на XML документи, т.к. зарежда целия документ във временната паметта на компютъра, което при системи с недостатъчно количество RAM памет би довело до фатални грешки. Когато обектът от тип *accountForm* бъде достъпен, то тогава започва и инициализирането на необходимите DOM обекти като *DocumentBuilderFactory* и *Document*, след което желаният XML документ се обработва за да се зареди в паметта или

се създава нов такъв, ако документът не е наличен. Следващата стъпка от алгоритъма е добавянето на нов *user* елемент към вече отворения XML документ.



Фиг. 3.14: UML диаграма на дейността за създаване на вътрешни ресурси

За да може да се различава еднозначно от останалите потребители, към новия елемент се добавя и атрибут - *ID*. За да се определи стойността на атрибута се проверява дали XML документът е новосъздаден, като в този случай първоначалната стойност на атрибута е единица. В случаите, когато документът не е празен, се създава нова структура от данни от тип List, която се попълва с всички налични потребители в документа. Стойността на *ID* атрибута в този случай е равна на дължината на новосъздадената структура от данни. Когато новият елемент (*user*) може да бъде еднозначно определен в XML документа по неговия атрибут, то тогава се създават и всички негови дъщерни елементи. Стойностите на новосъздадените елементи се извличат от подадения в началото на алгоритъма обект от тип *accountForm*. Следващата стъпка е инициализация на обекти от тип *Transformer* и *TransformerFactory*, чрез които се добавя крайният резултат към отворения документ. При успешен запис на данните, потребителят има възможност да използва своите потребителско име и парола за достъп до системата, в противен случай OntoMS визуализира съобщение за грешка и записът на данните се прекратява. На фиг. 3.14 е представена UML диаграма на дейността, визуализираща алгоритъма за добавяне на потребителски профил в XML документ.

3.12. Изводи към трета глава

На базата на разработения софтуер за генериране на управляващи програми, базирани на стандарта IEC 61499, чрез употребата на онтологии могат да бъдат направени следните изводи и заключения:

1. Разработен е софтуер за генериране на управляващи програми базирани на стандарта IEC 61499;
2. Информацията, необходима за генерирането на управляващите програми, се извлича от специално разработени домейн онтологии, обединени от висша онтология (представени във втора глава);
3. Имплементирани са базови нива на сигурност в разработения софтуер, осигуряващи защита на вътрешните функции на софтуера и информацията за отделните потребители;
4. Имплементирана е базова функционалност на административна и комуникационна дейност;

5. Разработеният софтуер притежава функционалността за изтриване, създаване и копиране на файлове и директории както и възможност за манипулация на данните във вече съществуващи такива;
6. Разработеният софтуер притежава базовата функционалност за поддържане на собствена нерелационна база от данни;
7. Разработеният софтуер имплементира възможност за работа с локален онтологичен файл, онтологичен файл, намиращ се в интернет пространството или разработената статична библиотека, с която разполага;
8. Разработеният софтуер имплементира възможност за избор на типа на генерираната управляваща програма - IEC 61499, IEC 61131 или CNC.

ГЛАВА 4. АПРОБАЦИЯ НА ПОДХОДА ЗА ГЕНЕРИРАНЕ НА УПРАВЛЯВАЩИ ПРОГРАМИ ВЪРХУ КОМПЛЕКС РАБОТНИ СТАНЦИИ FESTO

4.1. Предназначение и конструктивни особености на комплекса работни станции FESTO

4.1.3. Сортираща станция FESTO

Сортиращата станция FESTO служи за сортиране на изделия на базата на разпознаване чрез оптични и индуктивни сензори. Разпределянето на постъпилите на станцията детайли се класифицират по материал (метал – неметал) и цвят (червен и черен). Върху подвижната маса на станцията са монтирани профилната плоча (с разположените върху нея разпределителен конвейерен модул, магазинен модул и част от пневматичната система), контролната конзола и таблото за управление. В таблици 4.5 и 4.6 са представени входовете (сензорите) и изходите (изпълнителните механизми) на сортираща станция FESTO, а също така и съответните им означения в интерфейсните функционални блокове.

4.1.4. Последователност на работа на комплекса работни станции FESTO

Комплексът от работни станции на фирма FESTO е изграден от обслужваща станция, обработваща станция и сортираща станция. Самият комплекс е представен на фиг. 4.1.

Начална позиция (за обслужваща станция FESTO)

- Повдигащият цилиндър е прибран (хващачът е вдигнат);
- Хващачът е отворен;

Начални условия

- Наличие на детайл в гнездото за детайли;

Последователност

- Наличието на детайл се засича от оптичен сензор;
- Повдигащият цилиндър се спуска надолу;
- Хващачът се затваря;
- Повдигащият цилиндър се изкачва нагоре;
- Осъществява се преместване по линейната ос към първа позиция на обработващата станция;

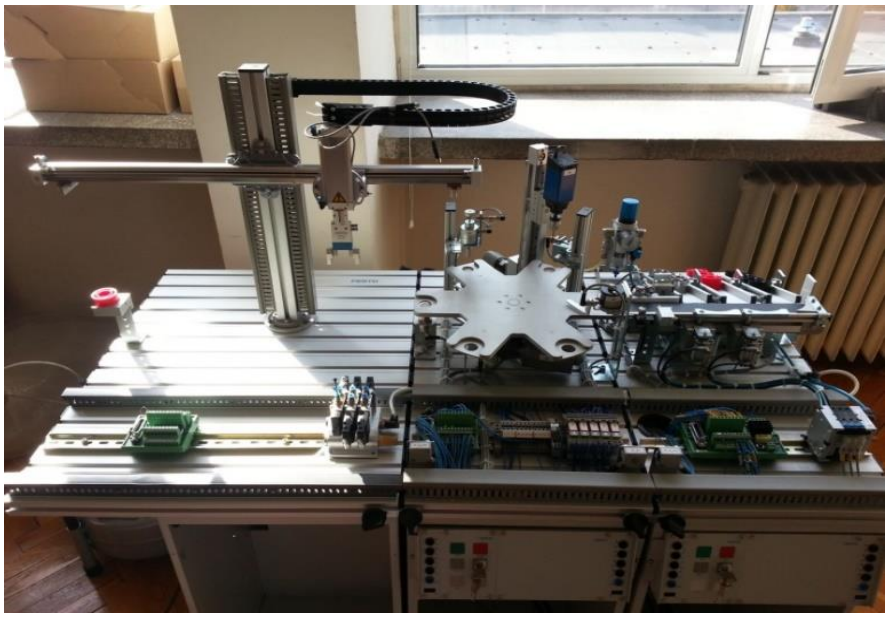
- Повдигащият цилиндър се спуска надолу;
- Хващачът се отваря и детайлът се поставя на първа позиция на обработващата станция;
- Повдигащият цилиндър се изкачва нагоре;
- Осъществява се преместване по линейната ос към началното положение.

Начална позиция (за обработваща станция FESTO)

- Въртящата се маса е позиционирана;
- Буталото за проверка на правилното разположение на детайла е прибрано;
- Пробивната машина е във вдигнато положение;
- Моторът на пробивната машина е изключен;
- Захващащото бутало е прибрано;
- Електрическият палец не е задействан;

Начални условия

- Наличие на детайл на първа позиция;



Фиг. 4.1: Комплекс работни станции на фирма FESTO

Последователност

- Въртящата маса се завърта на 60^0, при наличието на детайл на първа позиция;
- Буталото за проверка се спуска надолу;
- При правилно поставен детайл буталото се прибира и масата се завърта на 60^0;
- Захващащото бутало излиза напред и захваща заготовката;
- Моторът на пробивната машина се включва;
- Пробивната машина се придвижва по линейната ос надолу;
- След като извърши обработващите операции, пробивната машина се придвижва по линейната ос нагоре;
- Моторът на пробивната машина се изключва;
- Захващащото бутало се прибира и отпуска заготовката;
- Масата се завърта на 60^0;
- Електрическият палец избутва детайла върху сортиращата станция.

Начална позиция (за сортираща станция FESTO)

- Спиращият механизъм пречи на придвижването на детайла;

- Отсекателя на магазин 1 е прибран;
- Отсекателя на магазин 2 е прибран;
- Моторът на конвейера е изключен;

Начални условия
- Наличие на детайл в началото на конвейера;

Последователност
- Наличието на детайл се засича от оптичен сензор PART_AV;
- Включва се мотора на конвейера;
- Разпознаване на цвят/материал от двата сензора преди спиращия механизъм;

Наличие на черен детайл; сортиране в края на конвейера
- Прибиране на спиращия механизъм;
- Придвижване на детайла;
- Засичане на детайла от светлоотразителният сензор B4;
- Изключване на мотора на конвейера, включване на спиращия механизъм, прибиране на отсекателя на магазин 1 или магазин 2 в случай, че някой от тях е бил отворен;

Наличие на сребрист детайл; сортиране в началото на конвейера
- Прибиране на спиращия механизъм;
- Отсекателя на магазин 1 се отваря;
- Придвижване на детайла;
- Засичане на детайла от светлоотразителния сензор B4;
- Изключване на мотора на конвейера, включване на спиращия механизъм, прибиране на отсекателя на магазин 1 или магазин 2 в случай, че някой от тях е бил отворен;

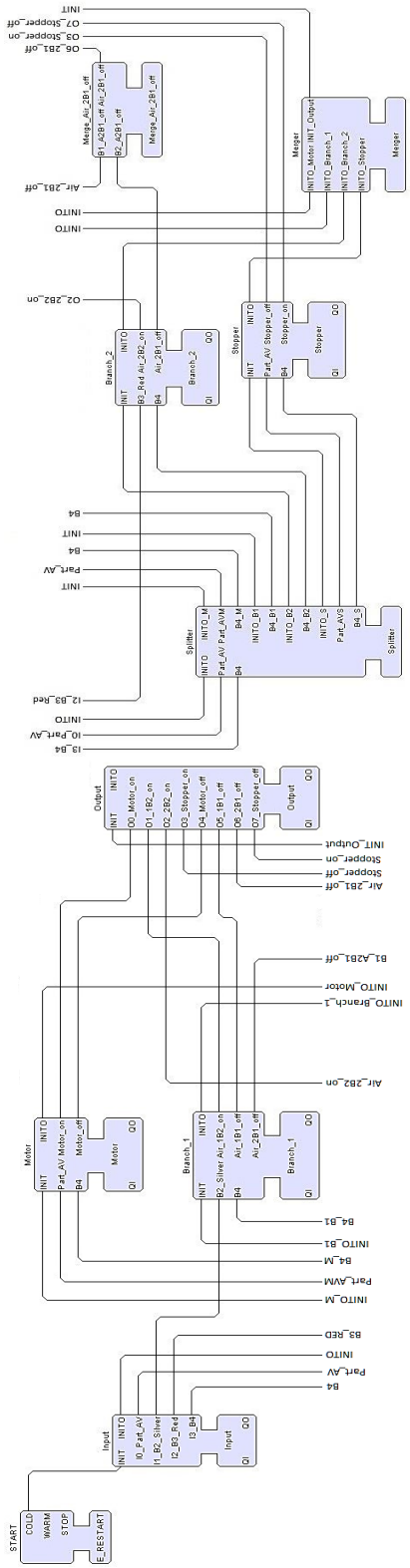
Наличие на червен детайл; сортиране в средата на конвейера
- Прибиране на спиращия механизъм;
- Отсекателя на магазин 2 се отваря;
- Придвижване на детайла;
- Засичане на детайла от светлоотразителният сензор B4;
- Изключване на мотора на конвейера, включване на спиращия механизъм, прибиране на отсекателя на магазин 1 или магазин 2 в случай, че някой от тях е бил отворен;

4.2. Генериране на управляващи програми за комплекса работни станции, базирани на стандарта IEC 61499

4.2.3. Сортираща станция (Sorting workstation)

Чрез система „*Sorting station“* сортиращата станция сортира детайлите в следната последователност - сребърен детайл, червен детайл и накрая на конвейерния модул черен детайл. Разработеният ресурс за описаната последователност е представен на фиг. 4.9. За сортиращата станция са разработени шест варианта на сортиране на детайлите. В

настоящата глава е представена системата за сортиране в последователност сребърен детайл, червен детайл и черен детайл.



Фиг. 4.9 (б): Ресурс от функционални блокове на система „Sorting station“

Фиг. 4.9 (а): Ресурс от функционални блокове на система „Sorting station“

След като бъде стартирана системата се подава входно събитие INIT към интерфейсният функционален блок „*Input“* от функционалния блок „*E_RESTART“* (този блок се генерира автоматично при създаването на нова система). След като получи това събитие, блокът „*Input“* го предава като изходно събитие (INITO) към функционалния блок *„Splitter“*, като този блок разделя събитието и го предава като изходни събития (INITO_M, INITO_B1, INITO_B2, INITO_S) към блоковете „*Motor“, „Branch_1“, „Branch_2“* и „*Stopper“*, за които е входно събитие (INIT). Накрая блоковете подават изходно събитие (INITO) към обединяващия функционален блок „*Merger“*, който обединява събитията (INITO_Motor, INITO_Branch_1, INITO_Branch_2, INITO_Stopper) и подава изходно събитие (INITO_Output) към интерфейсния функционален блок „*Output“*. По този начин всички функционални блокове са активирани.

Получавайки сигнал от сензора Part_AV, интерфейсният блок *„Input“* генерира изходно събитие (I0_Part_AV) към блока *„Splitter“*, който го разделя и генерира изходни събития (Part_AVM и Part_AVS) към блоковете *„Motor“* и *„Stopper“*. От входните събития двата блока генерират изходни събития (Motor_on и Stopper_off) към интерфейсният блок *„Output“* за да включат електромотора на конвейера и приберат спиращия механизъм. Получавайки сигнал от сензора B2, интерфейсният блок *„Input“* генерира изходно събитие (I1_B2_Metalic) към блока *„Branch_1“*, който генерира изходни събития (Air_1B2_on и Air_2B1_off) към блоковете *„Output“* и *„Merge_Air_2B1_off“* (подава изходно събитие към блок *„Output“*) за да включи разпределителен палец 1 и изключи разпределителен палец 2. Получавайки сигнал от сензора B3, интерфейсният блок *„Input“* генерира изходно събитие (I2_B3_Red) към блока *„Branch_2“*, който генерира изходно събитие (Air_2B2_on) към блока *„Output“* за да включи разпределителен палец 2. Получавайки сигнал от сензора B4, интерфейсният блок *„Input“* генерира изходно събитие (I3_B4) към блока *„Splitter“*, който го разделя и генерира изходни събития (B4_M, B4_B1, B4_B2 и B4_S) към блоковете *„Motor“, „Branch_1“, „Branch_2“* и *„Stopper“*. От входните събития блоковете генерират изходни събития (Motor_off, Air_1B1_off, Air_2B1_off и Stopper_on) към интерфейсният блок *„Output“*, за да бъде

изключен електромоторът на конвейера, да се активира спиращият механизъм и да се приберат в начално състояние отсекателите 1 и 2 (ако има включени такива).

4.3. Симулация на функционалните блокове

На всеки един от функционалните блокове, който е генериран от разработения софтуер, описан в глава 3, е направена симулация, за да се провери неговата логика и функционалност. Също така, в допълнение, е проверена и съвместимостта на генерираните блокове с по-нови версии на софтуера FBDK, което не само позволява работа с най-новите възможности на средата за разработване на функционални блокове, но и позволява използването на най-новите технологии, имплементирани в програмния език Java. Програмите са качени успешно на контролери Netmaster II, а самите тестове върху комплекса работни станции са положителни

4.4. Изводи към четвърта глава

На базата на направената апробация на подхода за оперативна съвместимост, могат да бъдат направени следните изводи:

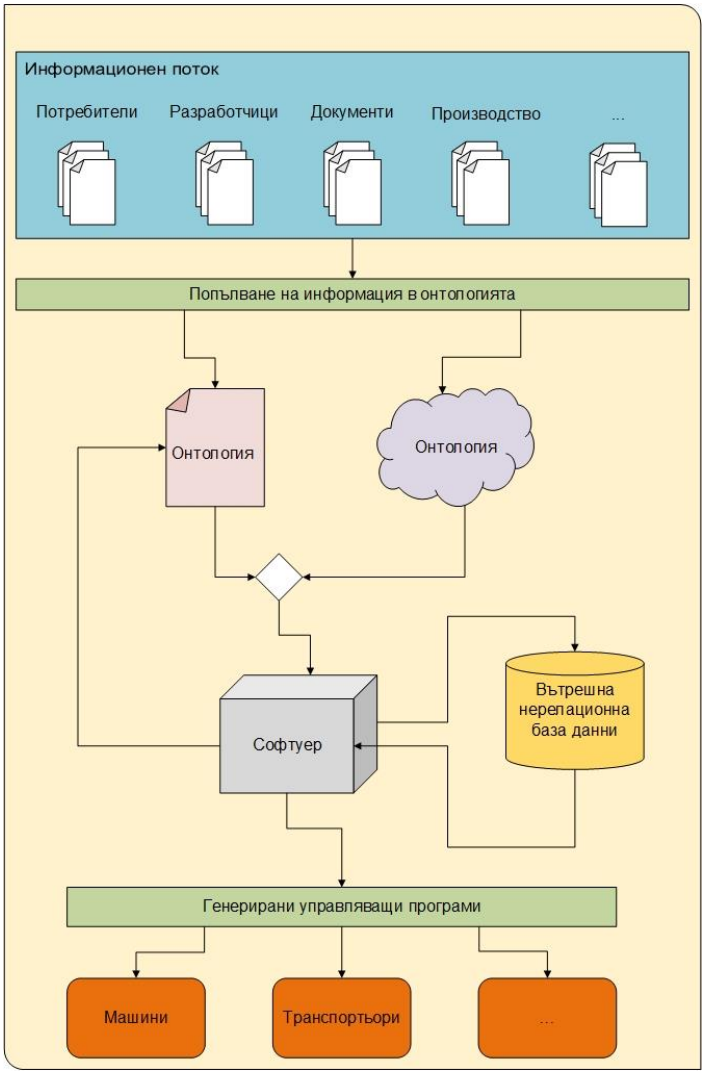
1. Извършените практически тестове за реализация на различни технологични операции, като подаване на заготовка, тестване, обработване и последващо сортиране на базата на физични характеристики, са извършени на три от работните станции на фирма FESTO – обслужваща станция, обработваща станция и сортираща станция;
2. Направен е подробен анализ на технологичните възможности на трите работни станции на фирма FESTO, а също така и на съпътстващите ги модули и сензори;
3. Комуникационната мрежа между работните станции и генерираните функционални блокове е осъществена с контролери Netmaster II;
4. Генерираните функционални блокове са тествани за съвместимост с най-новата версия на средата за разработване на функционални блокове FBDK версия 2.5, която е съвместима с Java 8 и позволява имплементация на най-новите възможности на програмния език;
5. Всеки един от генерираните функционални блокове е верифициран чрез симулация;
6. Генерираните от разработения софтуер управляващи програми са тествани успешно върху комплекса от работни станции на фирма FESTO.

ГЛАВА 5. ОБОБЩАВАНЕ НА ПОДХОДА ЗА ОПЕРАТИВНА СЪВМЕСТИМОСТ В РЕКОНФИГУРИРАЩИ СЕ ПРОИЗВОДСТВЕНИ СИСТЕМИ (РПС) ЗА МАШИНОСТРОЕНЕТО

5.1. Обобщаване на подхода за оперативна съвместимост до настоящия момент

Разработването на подход за оперативна съвместимост в машиностроенето е една изключително комплексна и трудна задача. Базирайки се на три дименсионната референтната рамка за постигане на оперативна съвместимост представена в ISO 11354-1:2011, настоящият дисертационен труд има за цел да постигне оперативна съвместимост чрез употребата на подхода федерация. Това означава, че за постигането на оперативна

съвместимост е нужно да бъде разработена онтология. На фиг. 5.1 е представен схематично разработеният подход.



Фиг. 5.1: Подход за постигане на оперативна съвместимост

Данните в една онтология могат да бъдат извличани от различни източници. Например, първоначалните данни могат да бъдат попълнени от разработчиците на онтологии, като в последствие информационният поток може да се разшири до попълване на данни от потребителите на разработената система, от самото производство или да се извлича информация директно от документи и релационни бази данни. Настоящият дисертационен труд е базиран на разработена висша онтология, отговаряща на стандарта за обмен на данни между информационните системи и системите за управление на производството - ISO/IEC 62264. От разработената висша онтология са избрани базови мета класове, чрез които да бъдат илюстрирани основните процеси в едно производство.

За да бъде възможно извличането на информация от онтологичния файл, е разработен софтуер за генериране на управляващи програми, базирани на стандарта IEC 61499. Софтуерът предлага възможност за извличане на информацията както от локален онтологичен файл, така и от такъв, намиращ се в интернет пространството. Независимо, кой от двата варианта е избран, чрез поредица от интерактивни графични елементи потребителя има възможност да избере детайл, намиращ се в онтологията, както и съответстващите му технологични операции, а също така и необходимите за неговото производство инструменти и машини. След като цялата необходима информация бъде извлечена от онтологията, софтуерът предоставя на потребителя възможност за генериране на управляващи програми. В софтуера е имплементирана и базовата функционалност за поддържане на нерелационна база данни. Разработеният софтуер е представен в трета глава.

5.2. Насоки за бъдещо развитие на подхода за оперативна съвместимост

5.2.1. Общи насоки за бъдещо развитие

С цел да се повиши ефективността, функционалността и бъдещата поддръжка на разработения софтуер, а също така и на разработените домейн онтологии, е необходимо да бъдат имплементирани следните подобрения:

- В настоящата си версия, разработеният софтуер директно генерира управляващите програми в работната директория без предварително да визуализира генерираните функционални блокове. Имплементирането на графичен редактор, чрез който да бъдат представени и редактирани при необходимост генерираните функционални блокове, би намалило времето за отстраняване на възникнали грешки от страна на потребителя.
- Имплементирането на графичен редактор за функционалните блокове дава възможност за създаване на функционалност за представяне на потоците от данни и събития между отделните функционални блокове, чрез което потребителят може да редактира XML файла, отговарящ за описанието на дадена система от функционални блокове.
- Необходимо е, да се внедри защита от стартирането на повече от една инстанция на софтуера върху конкретна машина, т.к. това би довело да възникването на грешки, които да доведат до загубата на ценна информация.
- Добавяне на ефективна система за съобщения и привилегии, чрез която да се подобри комуникацията между отделните потребители.
- Имплементиране на функционалността, веднъж вече генерирани програми да бъдат добавяни към статичната библиотека, като готови решения, чрез което би се разширила функционалността на софтуера в офлайн режим или при липса на онтологичен файл.
- Разширяване на отделните функции за защита, удостоверяване и криптиране на информацията чрез употребата на стандартните библиотеки за сигурност на Java и готови криптографски такива.
- При четене и писане DOM моделите да се заменят с JAXB такива, което би довело до по-четим и лесен за поддържане код, и намаляване на обема на заетата памет (в процес на четене на големи документи).
- Имплементация на многонишково програмиране в процеса на генериране на управляващите програми, което би довело до повишаване на производителността в случаите, когато процесорът притежава повече от едно ядро.
- Домейн онтологиите да се разбият на допълнителни подобласти, а тяхното обединение да се извършва чрез функцията за вмъкване, а не функцията за обединение, т.к. по този начин се увеличава модулността на самата система, а също така се намалява и последващото време за обработка и извличането на изводи от вече съществуващите факти.

5.2.2. Миграция от Java Swing към JavaFX

JavaFX представлява съвкупност от графични и мултимедийни пакети, които позволяват на разработчиците да проектират, тестват, проверяват и разработват висококачествени приложения, които са способни да работят стабилно на различни платформи. Тъй като JavaFX библиотеката е писана като Java API, то кодът на JavaFX може да реферира интерфейси от всяка една Java библиотека. Визуализацията на JavaFX приложения може да бъде широко персонализирана чрез каскадните стилове (Cascading Style Sheets, CSS), които отделят визуализацията и стила от имплементацията, за да могат разработчиците да се концентрират върху разработването на вътрешната функционалност на разработвания софтуер. Тъй като програмните интерфейси на JavaFX са напълно

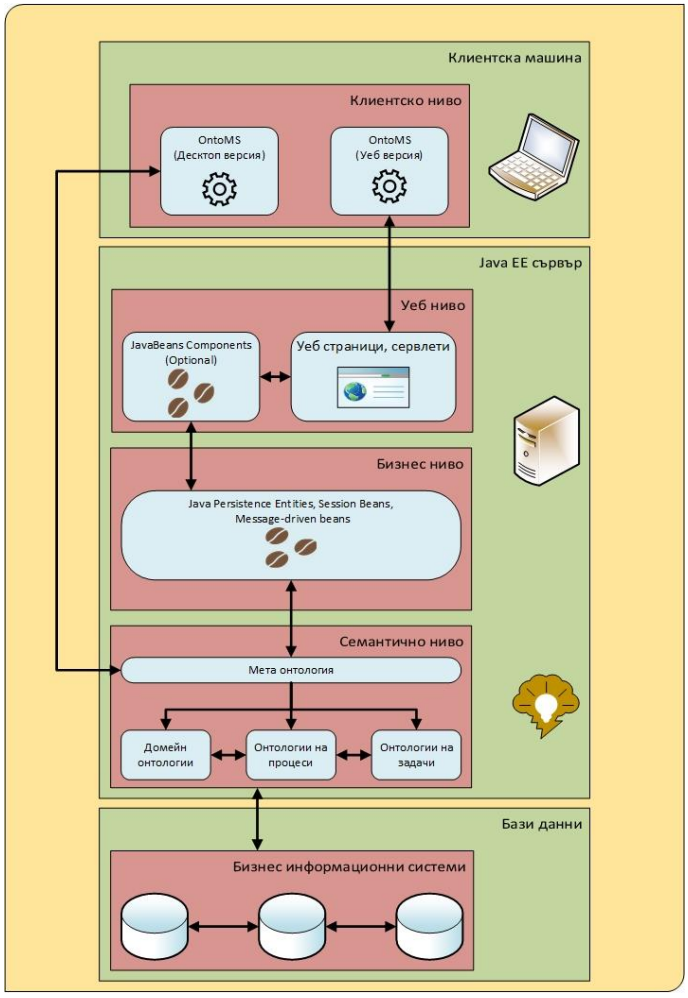
интегрирани към Java SE Runtime Environment (JRE) и Java Development Kit (JDK), а JDK е налична за всички основни платформи (Windows, Mac OS X и Linux), то всички JavaFX приложения, които са компилирани към седма версия на JDK или по-нова, са достъпни на същите платформи.

5.2.3. Разработване чрез тестване

Основните предпоставки при разработването на софтуер е поддръжката на бизнес процесите и улесняване управлението на самите организации, в които съответния софтуер е интегриран. Дейността на софтуерните разработчици трябва да бъде фокусирана в предоставянето на висококачествени системи, които са гъвкави и лишени от грешки. За съжаление, дори след няколко десетилетия, напредък в софтуерната индустрия, качеството на разработваните софтуерни продукти все още е проблем. Дефектите в разработения софтуер създават нежелани загуби, като правят системите нестабилни, непредсказуеми или напълно неизползваеми. Сами по себе си те намаляват стойността на софтуера, а в някои случаи са способни да донесат многократно повече негативи, отколкото позитиви. Дефектите са най-очевидния проблем при некачествено написан програмен код. Такъв код е труден за разбиране, труден за поддръжка и модификация, а също така бавен и скъп за разработване.
някои от най-популярните към настоящия момент помощни средства за изграждане на тестова среда са:

- **Системи за контрол на версията**
  - *Git*
- **Виртуални машини**
  - *VirtualBox*
  - *Vagrant*
  - *Docker*
- **Средства за управление жизнения цикъл на проектите**
  - *Ant*
  - *Maven*
- **IDE**
  - *Eclipse*
  - *IntelliJ IDEA*
- **Тестови среди**
  - *Junit*
  - *TestNG*
  - *TestFX*
  - *Selenium*
  - *Selenide*
- **Среди следящи за покритието на тестовете**
  - *Java Code Coverage (JaCoCo)*
- **Маскиращи средства**
  - *Mockito*
  - *EasyMock*
  - *PowerMock*

5.2.4. Миграция от десктоп към уеб базирано бизнес приложение



Фиг. 5.9:Интеграция на направената разработка с бизнес технологии

Разработеният в рамките на настоящата дисертация Java базиран софтуер за генериране на управляващи програми, базирани на стандарта IEC 61499, е подходящ за работа с малки обеми от данни, а разработените домейн онтологии могат работят единствено с примитивни типове от данни. Един възможен подход за решаването на този проблем е миграция от десктоп към уеб базирано приложение чрез имплементацията на Java Enterprise Edition технологиите, обединени със семантичните възможни и функционалността на Oracle база данни. На фиг. 5.9 е представена интеграцията и информационните потоци между отделните компоненти на разработените онтологии и софтуер от дисертационния труд с оглед на представените в настоящата точка технологии за миграция от десктоп към уеб базирано приложение. Необходимо е да се направи уточнение, че представената на фиг. 5.9 многослойна структура е валидна за общия случай, в който не се използва Oracle база данни. При този сценарии, съхранението на онтологиите и семантичните данни от семантичното ниво се съхраняват в т.н. семантични хранилища, а връзката с релационните данни от базата се осъществява чрез допълнителни програмни интерфейси, които не са обект на разглеждане в настоящия дисертационен труд.

5.3. Изводи към пета глава

На базата на разработения подход за оперативна съвместимост в реконфигуриращи се производствени системи (РПС) за машиностроенето могат да бъдат направени следните изводи и заключения:

1. Предложена е практическа реализация на разработения подход чрез употребата на онтологии и Java базирано десктоп приложение за генериране на управляващи програми базирани на стандарта IEC 61499;
2. Предложени са насоки за бъдещото развитие на разработеното десктоп приложение, чрез които да се подобри неговата функционалност, производителност и сигурност на информацията;
3. Предложени са насоки за по-добро разработване на домейн онтологиите, чрез което да се подобри тяхната функционалност и увеличи производителността на цялата система;

4. Предложена е миграция от десктоп към уеб базирано бизнес приложение, чрез технологичните възможности на Java EE платформата и семантичните възможности на Oracle база данни.

НАУЧНО-ПРИЛОЖНИ И ПРИЛОЖНИ ПРИНОСИ

Научно-приложни приноси

1. Предложен е подход за осъществяване на оперативна съвместимост на РПС в машиностроенето чрез употребата на онтологии и Java десктоп приложение за генериране на управляващи програми, базирани на стандарта IEC 61499.
2. Предложена е домейн онтология „Екипировка“, която отговаря на мета клас „Оборудване“ (*Equipment*) от стандарта IEC/ISO 62264 и обхваща заложените в него свойства, атрибути и класове, а също така и съответните релации между тях. Онтологията позволява свързването на индивиди и класове в машини и системи, а също така позволява и тяхната реконфигурация.
3. Предложена е домейн онтология „Материали“, която отговаря на мета клас „Материали“ (*Material*) от стандарта IEC/ISO 62264. Чрез наложени ограничения върху класовете изграждащи онтологията се осъществява сортировка на отделните материали на база техните физични свойства.
4. Предложена е домейн онтология „Продуктови дефиниции“, която отговаря на мета клас „Продуктови дефиниции“ (*Product Definition*) от стандарта IEC/ISO 62264. Освен заложените свойства, атрибути и класове, които са заложени в стандарта онтологията предоставя възможност за описание на отделните производствени детайли на база техните геометрични характеристики.
5. Предложена е домейн онтология „Персонал“, която отговаря на мета клас „Персонал“ (*Personnel*) от стандарта IEC/ISO 62264. Онтологията структурира отделните типове информацията за наетите служителите на бизнес предприятието, като йерархична структура, отдели, квалификация и налага ограничения на служителите относно възможностите за работа с определен тип машини спрямо тяхната настояща квалификация.
6. Предложена е домейн онтология „Процесни сегменти“, която отговаря на мета клас „Процесни Сегменти“ (*Process Segment*) от стандарта IEC/ISO 62264. Онтологията дефинира класове, които отговарят на основните технологични операции, а също така дефинира и ограничения относно последователността на тяхното изпълнение. Онтологията използва концепция за подредени списъци, за която не съществуват стандартни конструктори в езика OWL2 като в същото време запазва своята изразителност в рамките на езика OWL2 DL.
7. Предложени са насоки за интеграция на разработените домейн онтологии с мета онтологията, които водят до увеличаване производителността на цялата система. Интеграцията на тези онтологии с мета онтологията създава комплексна система за управление на знанията, която може да се използва за разрешаване на различни по сложност и обхват производствени и бизнес задачи.

8. Предложени са насоки за по-добро разработване на онтологичните модели, чрез които да се постигне по-голяма гъвкавост, а също така да се подобри и тяхната функционалност.
9. Предложен е подход за извличане на информация от онтологии за генериране на управляващи програми, базирани на стандарта IEC 61499.

Приложни приноси

1. Разработени са онтологии с използване на езика OWL2 DL, които са верифицирани чрез различни системи за логически анализ и извод. Всяка една от тях може да се използва и самостоятелно като база от знания за съответната област.
2. Разработено е десктоп софтуерно приложение за генериране на управляващи програми базирани на стандарта IEC 61499. Необходимата информация за генериране на управляващи програми се извлича от разработените онтологични модели. Софтуерът позволява работа както с локални онтологични файлове, така и с файлове, локализирани в интернет пространството, а също така предоставя и две допълнителни възможности за типа на генерираните управляващи програми - IEC 61131 и CNC. За разработката на софтуера е използван програмния език от висше ниво Java.
3. Разработена е статична библиотека за генериране на управляващи програми, базирани на стандарта IEC 61499, която позволява употребата на функционалността на софтуера и при липса на онтологичен файл.
4. Генерираните управляващи програми, базирани на стандарта IEC 61499 за комплекса работни станции на фирма FESTO, са внедрени в лекционния материал на следните специалности:
    - „Компютърно проектиране и технологии в машиностроенето" към МТФ на ТУ София за образователно-квалификационна степен бакалавър и магистър.
    - „Информационни технологии в индустрията" към ФКСТ на ТУ София за образователно-квалификационна степен бакалавър.
5. Предложени са насоки за бъдещото развитие на разработения софтуер чрез които да се подобри неговата архитектура, функционалност, производителност и сигурност.
6. Предложена е миграция от десктоп към уеб базирано бизнес приложение чрез технологичните възможности на Java EE платформата и семантичните възможности на Oracle база данни.

СПИСЪК НА ПУБЛИКАЦИИТЕ ПО ДИСЕРТАЦИОННИЯ ТРУД

1. G. Popov, K. Stoyanov, G. Stambolov, "Automation control of sorting station based on IEC 61499 function blocks", Eighth international conference of on Electrical and Control Technologies ECT – 2013, Kaunas, Lithuania, 2013, pp. 13 – 18, ISSN 1822-5934.

2. K. Stoyanov, "Approach for software development for generation of control programs based on IEC 61499 standard", International journal for science, technics and innovations for the industry – MTM, Issue 5, Borovets, Bulgaria, 2016, pp. 51 – 53, ISSN 1313-0226.

3. K. Stoyanov, D. Gocheva, I. Batchkova, G. Popov, “Domain ontology of materials in reconfigurable manufacturing systems”, САИ XXIV международен симпозиум, Банкя, България, 2016, стр. 105 – 108, ISSN 1313-2237.

4. K. Stoyanov, D. Gocheva, I. Batchkova, G. Popov, “Domain ontology of the equipment in manufacturing systems”, International journal for science, technics and innovations for the industry – MTM, Issue 5, Borovets, Bulgaria, 2016, pp. 54 – 57, ISSN 1313-0226.

5. К. Стоянов, Г. Попов, И Бачкова, „Съвременно състояние на проблема за оперативна съвместимост на реконфигуриращи се производствени системи“, 8ма международна научна конференция за млади изследователи, Варна, България 2014, стр. 31 – 35, ISSN 1310-3946.

6. К. Стоянов, Д. Гочева, И. Бачкова, Г. Попов, „Онтологии и стандарти в реконфигуриращи се производствени системи“, 28ма международна научна конференция на Машинно-технологичен факултет на ТУ - София, Созопол, България, 2015, стр. 343 - 348, ISBN: 978-619-167-178-6.

7. К. Стоянов, Д. Гочева, И. Бачкова, Г. Попов, „Подход за създаване на домейн онтологии на системи за обработка на призматично корпусни детайли“, 28ма международна научна конференция на Машинно-технологичен факултет на ТУ - София, Созопол, България, 2015, стр. 349 - 354, ISBN: 978-619-167-178-6.

SUMMARY

DEVELOPING AN APPROACH FOR INTEROPERABILITY IN RECOFIGURABLE MANUFACTURING SYSTEMS

M.Sc. eng. Kostadin Stoyanov

In the dissertation thesis, an approach is proposed in order to implement interoperability for reconfigurable manufacturing systems through the usage of ontologies and Java desktop application for the generation of control programs based on the IEC 61499 standard.

Five domain ontologies ("Equipment", "Materials", "Process Segments", "Product Definitions", "Staff") that are based on the IEC/ISO 62264 standard have been proposed. Each one of the developed domain ontologies is corresponding to a meta-class of a meta-ontology that is based on the same standard. The integration of these ontologies with the meta-ontology creates a complex knowledge management system that can be used to solve production and business tasks with different scope and complexity.