Софийски университет “Св. Климент Охридски”
Факултет по Математика и информатика
Катедра "Информационни технологии"

ДИПЛОМНА РАБОТА

Тема: Приложение на информационните технологии в телероботиката

Дипломант: Нурани Путра Игусти
Специалност: Информатика
Факултетен номер: 42148
Научен ръководител: доц. д-р. Антоний Попов

Дипломант: .....
/Нурани Игусти/

Рецензент: .....
/ст.н.с. Недко Шиваров – ЦЛМП -БАН/

София,
06. 2007 г.

Научен ръководител: .....
/доц. д-р. Антоний Попов/

Съдържание

Въведение

Целта на тази дипломна работа е запознаване с телероботиката и връзката ѝ с информационните технологии. Използвани са някои основни термини и понятия свързани с роботите, описанието и приложението им. Накратко е представено историческото развитието на езиците за програмиране и визуализация на роботи, както и примери за техните приложения. От значение е, да се покаже ефективността и наложителното използване на телероботите.

Изложение:

Глава 1 запознаване с термина робот, видовете роботи и телероботиката. **Глава 2** обяснява нуждата от възникване на един универсален език за комуникация с телеробота. Кратко запознаване с езика XML, който е базисен за езиците за управление (RoboML, XRCL) и обработка на изображения (SVG, VRML) на роботите. В **глава 3** описава хирургически робот и няколко вида системи. **Глава 4** Сравнява две от най-известните системи (ZEUS, da Vinci), тяхното приложение и полза. В **глава 5** читателят може да тества новите си знания за роботите и телероботите.

1. Въведение в роботиката и телероботиката

1.1. Описание на робот

Терминът робот произлиза от чешката дума ”robota”, означаваща промишлен труд и за първи път е употребен от чешкия писател Карел Чапек през 1920 г. в пиесата му „R.U.R.“ (Росумски универсални роботи).

Трите закона на роботиката в научната фантастика са правилата за поведение на роботите, описани от Айзък Азимов. За първи път те се споменават в разказа му „Runaround“ от 1942 г. Тези закони са вградени във всеки робот в книгите му и гарантират безопасното му използване [1]. Законите гласят:

1) Роботът не може да нарани човешко същество или с бездействието си да позволи нараняването му.
2) Роботът трябва да изпълнява заповедите, дадени му от човешко същество, освен ако това не противоречи на Първи Закон.
3) Роботът трябва да защитава собственото си съществуване, освен ако това не противоречи на Първи и Втори закон.

Най-общо под робот се разбира голям набор от машини, общото между които е, движението и способността да се извършват механични операции. Роботите се проявяват в множество форми, вариращи от хуманоид, който имитира човешката форма и начина му на движения, до индустриални, чиито вид се определя от функцията, която изпълняват. Те могат да бъдат групирани на мобилни роботи, манипулаторни роботи (индустриални роботи) и реконфигурируеми роботи - приспособяващи се към задачата, която изпълняват.
От практическа гледна точка роботът е механично устройство, което изпълнява задачи чрез директен или частично управление от страна на човека, както и напълно автономно. Обикновено се използват за задачи прекалено еднообразни, мръсни или опасни за човека – например унищожаващи бомби роботи, роботизирани ръце и други или действащи в съответствие със заложена в тях способност за вземане на решения, предоставена от изкуствен интелект (ИИ).

Някои роботи се управляват от програмирани компютри, така те могат да „създават” обратна връзка, за да взаимодействат със средата, но не притежават действителна интелигентност.

В повечето случай роботите се състят от:

- Механични устройства като задвижваща платформа, ръка или друга конструкция способна да взаимодейства със заобикалящата среда.
- Сензори на или около устройството, който „усещат” средата и връщат смислени отговори.
- Системи, които препредават информация засечена от сензорите и управляват устройството на базата на получената информация.
- Накой имат ИИ.

1.2. **Видове роботи**

Важно е да правим разграничение между различните видове роботи [2]:

1.2.1. **Индустриални**

Индустриалният робот е машина, която може да бъде програмирана да изпълнява предварително поставени задачи с висока степен на автономност, в предварително зададена среда и повтаряемост на задачите. Обикновенно се използват във фабриките на поточните линии.

1.2.2. **Домакински**

Домакинският робот е устройство програмирано или контролирано в по-голяма или по-малка степен от човек. Той изпълнява задачи, който са често променяеми и неповторяеми в минимално управлевана и непредвидима среда. Примери за домакински роботи са тези използвани за чистене на прозорци и подове, сортиране на пакети и др.

1.2.3. **Телероботи**

Телероботите са тези, които функционират в среда отдалечена от човека-оператор или в обикновенно опасна или недостъпна среда. Например телероботите за космическите мисии, за дълбоко океанските изследвания и

медицината. Те действат в съответствие със заложените в тях способности за вземане на решения, предоставени от ИИ.

1.3. **Въведение в телероботиката**

Роботите под една или друга форма присъстват в живота ни от десетилетия. Манипулаторите „господар-подчинен” са специално звено на роботите, това в което човек е директно в процеса на управление определяйки действията му. Телероботите са модерна версия на манипулаторите „господар-подчинен”, които вече се използват успешно и в критични ситуации. Ранните механични „господар-подчинен” системи еволюират до днешните комплексни електро-механични телероботни системи и по този начин отварят вратите на операционната интервенция в медицината.
Днешните автономни роботи могат да бъдат програмирани, да се движат с висока прецизност и скорост, да извършват комплексни задачи като боядисване и сглобяване. Но дори така те не могат да извършват комплексни физически взаимодействия и да реагират адекватно в неочквани ситуации. От друга страна телероботите, поставяйки човека в процеса на управление, комбинират предимството на човешката съобразителност и механичната прецизност на роботите. Така се решават невъзможни до сега ситуации и проблеми.

В общия случай телероботът се състои от редица физически устройства като робот, камери, сензори и други механизми, които взаимодействат помежду си в сложна система от софтуерни програми и системи за визуализация, предаващи данни към човека-оператор и обратно, и които могат да се разпределят на различни машини, отдалечени на големи разстояния.

2. XML в телероботиката

2.1. История

Бързото развитие на телероботите през годините и големият брой фирми и научни организации, занимаващи се с изобретения и производство, довежда до нуждата от създаване на нов универсален език за комуникация с различните съставни системи на телеробота. Един от тези езици е *Record Definition Language* (RDL), който е добра стартова позиция, но не успява да се утвърди като стандарт поради неговата невъзможност за надграждане. Следват и многото други неуспешни опити за създаване на подобни езици като *Industrial Robot Language* (IRL), *Robo Script*. Поради голямото разнообразие на телероботите както и задачите за които те са предвидени, тези езици се оказват недостатъчни. Нужният език е *Extensible Markup Language* (XML), който се утвърждава като основен стандарт за описание на данни главно поради неговата опростеност и възможност за надграждане съобразно спецификата на задачите. Добре развитите формализми на езиците за програмиране, включително и на тези за програмиране на роботи, правят възможно тяхното превеждане в XML-базирани езици. Затова в днешно време при езиците за програмиране и пренос на данни при телероботите ние правим две основни разграничения: базирани на XML и други.

2.2. Описание на XML

2.2.1. **Дефиниция**

XML е разработен от World Wide Web Consortium (W3C) [3]. Използва се за структуриране на текст, който е четим, както за машини, така и за хора и представя прост формат за размяна на информация през Интернет между компютри.

2.2.2. **Деклариране**

XML декларацията е първия ред от документа. Тя го идентифицира като XML

документ. Декларация също указва версията на XML, използвана в документа. За сега това е 1.0.

```
<?xml version=”1.0”?>
```

Декларацията може да съдържа други атрибути, които да поддържат различни характеристики, като например множество от символни кодове. Тя не е задължителна, ако обаче е включен декларативен ред, той трябва да започва с първия символ на първия ред. Рекомендацията на XML препоръчва да се включи декларация във всички XML документи.

2.2.3. **Отварящ / затварящ таг**

Най-простият начин за кодиране на значение на част от текст в XML е заграждането от отварящ и затварящ таг. Отварящият таг съдържа името на тага между знаците за по-малко (<) и по-голямо (>), а съответстващия затварящ таг има знак за наклонена черта пред името на тага.

```
<tag> Телероботика </tag>
```

Един добре оформен XML документ има затварящ таг, който съответства на всеки отварящ.

2.2.4. **Елементи**

Изграждащият блок е елементът, който е и съдържанието на XML документа. Всеки елемент се състои от име и съдържание.

```
<country> Bulgaria </country>
```

Съдържанието на елемента е заградено от отварящ и затварящ таг. То може да бъде обикновен текст, други елементи (наследници) или комбинация от текст или елементи - такова съдържание се нарича смесено. Един XML документ е дърво от елементи с един коренен елемнт.

```
<root_element>
        <sub_element1> info </ sub_element1>

        <sub_element2> info </ sub_element2>
</root_element>
```

2.2.5. **Атрибути**

Друг начин да се добавят данни в XML документ е прибавянето на атрибути в началния таг. Стойността на един атрибут обикновено е проектирана, да се свърже като данни със съдържанието на съответния елемент. Празното пространство се използва като разделител на атрибутите на един таг.

Всеки атрибут има име следвано от знак за равенство и стойност на атрибута. Стойността на атрибута е заградена в кавички /единични или двойни/. В примера по-долу <my_tag> има два атрибута: att_1 и att_2.:

```
<my_tag att_1=”1” att_2=”2”> the text data </my_tag>
```

2.2.6. **Празен елемент**

Ако един елемент няма съдържания тогава затварящия таг може да се пропусне. В този случай наклонената черта се добавя към отварящия таг, за да индикира, че това е празен елемент. Елементното съдържание е всичко, което XML спецификацията позволява да се появи между началния start-tag и крайния таг end-tag, като текст, поделементи, коментари и обработващи инструкции. Празен елемент може също да има атрибути, както е показано в следващия пример:

```
<my_empty_element att_1=”1” att_2=”2” />
```

2.2.7. **Дефиниция на DTD**

Уникалният идентификатор на ресурси (The Uniform Resource Identifier - URI) в декларацията на документен тип сочи към документ, известен като дефиниция на документен тип (Document Type Definition –DTD). Форматът на DTD е дефиниран в

XML спецификацията и не е същият както за XML документа. DTD съдържа съвкупност от правила, които определят механизма, при който различни тагове в

документ могат да се използват заедно и атрибутите, които може да има всеки таг. Повечето XML процесори дават възможност за проверка на един XML документ спрямо DTD, позволявайки на приложенията бърза и безпроблемна проверка на структурата на един XML документ дали е коректна.

2.2.8. **Коментари**

Авторът на XML документа може да разполага коментари в документа, да добави бележки, индикации за собственост и др. Те са предназначени за четене от човек и се ингнорират от XML процесора. Коментарите започват със знак за по-малко, последвани от удивителен знак и две тирета и завършва с две тирета и знак за по-голямо, както е показано в долния пример. Коментарите не могат да се вмъкват в таговете. Трябва да се поставят преди или след тях.

```
<my_tag> content <!—poiasnenia --> </my_tag>
```

2.2.9. **Пространства от имена**

Namespace - логическо групиране на типове, по които можем да разграничаваме елементите на различни XML документни един от друг, когато ги комбинираме заедно в други документи. Те са свързани с лексиката, а не с типа на документа - пространството определя кои имена му принадлежат, а не какво означават и как се съчетават. Целта на namespace е да позволи едновременното съществуване на множество набори от тагове в един и същи документ. Всеки може да създава XML документи, като използва свои термини. Но това поражда проблеми, тъй като

съществуват много думи и всеки влага собствен смисъл в тях. Най-добрия начин да се реши проблемът е за всеки един елемент в документа да има точно определено име. Може всеки елемент от моя собствен XML документен тип да получава мой собствен префикс.

```
<my:name>Nury</my:name>
```

Имената и префикса се разделят от двуточие. Префиксите обаче не решават проблема, защото всеки може да създава префикси. Администрирането на му би

действало така, както администрирането на домейн имената в Интернет: човек трябва да се обърне към администраторите с префикса, който желае да използва. Ако още не е използван, той може да се използва, иначе трябва да се измисли друг префикс.

Следните онлайн ресурси дават подробна информация за XML докумнтацията:

• http://www.w3.org/XML/
• http://www.xml.com/axml/testaxml.htm

2.3. Езици за управление базирани на XML

2.3.1. **Описание на RoboML**

2.3.1.1. **Въведение**

RoboML (Robotic Markup Language) е експеримаентален XML базиран език използван за комуникация между автономни мобилни роботи и други компоненти на робота. Той е създаден от Мюнхенския технически университет и първата му поява е през пролетта на 2005г.

2.3.1.2. **Приложение на RoboML в Интерфейс човек-робот използващ комуникация на Агенти**

За избор на език за нуждите на Интернет-базираните Human Robot Interface (HRI) приложения би трябвало се избере език, който да позволява манипулация чрез относително лесен софтуер или чрез директна човешка намеса и да е подходящ за междуплатформени приложения. За да се удовлетворят тези изисквания се използва XML за развитието на спецификацията на RoboML. Създаденият RoboML е като общ език за програмиране на роботи, комуникация между агенти и за представяне на знания

HRI архитектура и RoboML чрез софтуерен HRI прототип е изграден от:

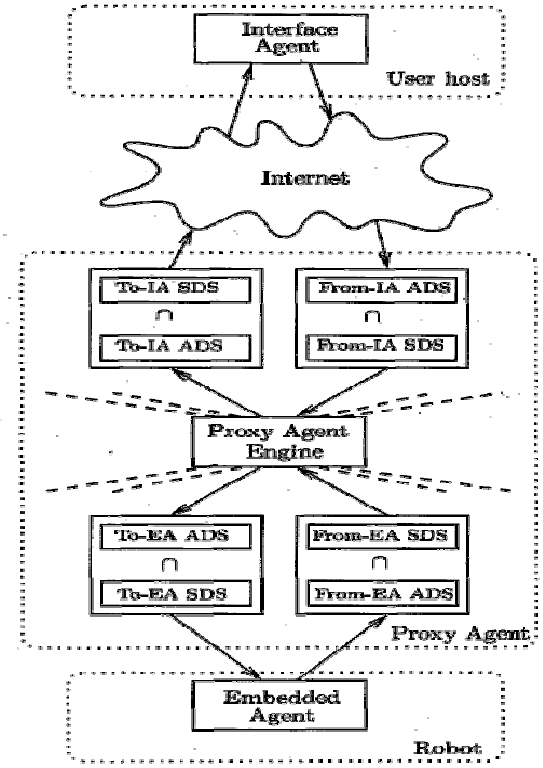
- агент на потребителския интерфейс (Interface Agents - IA)

- прокси агент
- embedded агент

Изисквания към HRI-и агентите:

- embedded агентът трябва да работи в реално време
- да има ограничени изчислителни ресурси

За да се улесни комуникацията между embedded агентите и агентите на потребителския интерфейс, в реално време чрез комуникационен канал с несигурно качество, се използва прокси агент като медиатор. Целта на прокси агента е намаляване на натоварването на комуникацията между агентите и използваните изчислителни ресурси за комуникационни нужди от embedded агентите и от агентите на потребителския интерфейс.
Ще разгледаме архитектура, която включва агенти на потребителския интерфейс (IA) и embedded агентите (Embedded Agents – EA), комуникиращи чрез Интернет [4]. (*Фигура 1*)



(Фигура 1) Proxy-Mediated Human-Robot Interface Architecture

2.3.1.3. **Proxy-Mediated Human-Robot Interface Architecture**

- Динамично създава и премахва връзки към IA-ти и EA-ти, изгражда и поддържа представянето на онтологиите на съответните агенти чрез XML схема.
- Извършва превода на между-агентната комуникация, улеснява асинхронната двупосочна комуникация между IA-ти и EA-ти.
- Информацията трябва да бъде обработена предварително, за да се намали натоварването на комуникацията.
- Колкото се може повече от натоварването, свързано с някакъв вид комуникация трябва да се извършва извън компютрите, които съдържат EA-и и IA-и.

За да се предложат истински Интернет базирани HRI-и също така е полезно да се избягват допълнителни натоварвания на изчислителните ресурси, използвани от агента на потребителския интерфейс. Често в Интернет-базираните HRI-и съществува отделна система, която да отговаря за Интернет услугите със слаби ограничения за изчислителните ресурси в сравнение със EA/IA компютрите.

Прокси агент наричаме съдържател на агента медиатор, който ще предлага някои допълнителни услуги за намаляване на изчислителното и комуникационното натоварване на другите агенти. Той поддържа база от знания, за да улесни асинхронната комуникация между IA-и и EA-и . За да се намали натоварването на междуагентната комуникация и да се предложат различни нива на достъп, прокси агентът поддържа схема на данните, които са достъпни във всеки един момент за IA-ите и за EA-ите. Схемите представляват подмножества на онтологиите, съответстващи на IA-и и EA-и. В прокси агента съществуват четири схеми на данни за всеки от IA-и/EA-и: available data schema (ADS) и subscribed data schema (SDS) за потоците от данни “към-агент” и “от-агент”

Възможен е следния сценарий, използвайки прокси агент:

IA-ите и EA-ите изпращат заявки и данни към прокси агента в собствените си онтологии. Тези заявки могат да бъдат изразени чрез get или subscribe методите на RoboML, а самият трансфер на данните да се осъществява чрез set метода. Заявките от тип subscribe от агент A подновяват записа „от-агент SDS” отговарящ на агент B, създаден за да върне необходимата информация. Заявките от тип get от агент A могат да не причинят постоянна или временна смяна на неговия „към-агент SDS”, но искането трябва да бъде анонсирано в „от-агент SDS” на агент B,

избран да отговаря на заявката. Едно съобщение със set метода от агент A може да поднови записа за неговия „от-агент ADS” и съдържанието на съобщението може да бъде запазено в базата от знания на прокси агента за следващо използване.
Връзката между ADS и SDS на фигурата е представена чрез знака за сечение на множества ∩, едновременно за потоци от тип „към-агент” и „от-агент”, с предположението че в повечето случаи потокът от данни е филтриран да удовлетворява едновременно SDS и ADS. Условието потокът от данни да бъде едновременно филтриран за SDS и ADS, не винаги може да бъде вярно за “от-агент” потоците . Възможно е прокси агентът да избере да не игнорира незаявената информация, а да я запази за бъдещо използване.
Механизмите на по-сложното поискване и запазване на незаявена информация могат да позволят асинхронното взаимодействие между IA-и и EA-и, вземайки в предвид ограниченията в реално време, наложени върху EA-и (и евентуално върху IA-и).

Синтаксисът на съществуващите схеми на данните и схемите на абонираните данни е базиран на XML и RoboML. Изборът на XML за основа, върху която да се изгради RoboML е основан на:

- XML е подходящ за описване на различни типове структурирани данни.

Възможността да се използва синтактичния модел намалява работата върху дизайна на един нов език.

- Удобството да се манипулира от човек, чрез прост софтуер или директно.
- XML съвместимият код е лесен за синтактичен разбор и лесен за генериране от софтуер.
- Възможността за междуплатформени приложения.
- Като последствие от адаптирането на XML като стандарт за уеб и поддръжката му от браузърите, той се явява като междуплатформен стандарт.

2.3.1.4. **Агент-комуникатор**

За да се дефинира семантиката на елементите на агентната комуникация в RoboML се използва KQML (Knowledge Query and Manipulation Language) модела [6].

Следните методи се представят като елементи на RoboML:

- set като аналог на tell
- get като аналог на ask

- subscribe като аналог на subscribe от KQML

Всеки от тези елементи може да има незадължителни атрибути sender, receiver и ontology:

- атрибутите sender и receiver съответстват на комуникационния слой на KQML модела
- ontology съответства на слоя за предаване на съобщения от KQML модела

За да се представи съдържанието на едно RoboML съобщение се разглеждат следните алтернативи:

- Съдържанието е вече представено чрез език, който има XML-базиран аналог .
- Хардуерът на роботите може да се представи като семантично дърво според физическата и логическата взаимна връзка на единиците.

- Софтуерните компоненти и потребителският интерфейс също могат да бъдат представени локално чрез дървовидни структури.
- Представянето е особено подходящо за вградени една в друга markup структури на XML-базирани езици. Използват се така наречените контейнер елементи.

RoboML използва контейнер елементи, за да представи съставните обекти в областа. За хардуерната онтология са дефинирани следните контейнерни елементи:
робот, колело, мотор, сензор, контролер и т.н. Тези контейнерни елементи могат да бъдат по-късно специфицирани чрез атрибутите „име" и „онтология". „Името" служи да идентифицира уникално елементите в съответния слой на вложеност. Контейнерът елемент може да има своя собствена онтология, която прави възможно описването на данни от различни области в един RoboML документ. Той може да има определен брой елементи-деца в съответствие с онтологията на областта. Действителните данни могат да бъдат показани в token елементите, с други думи елементите, които са от прости типове и не съдържат поделементи или атрибути. Токените специфицирани за хардуерната онтология на RoboML са:

- „позиция", „скорост", „ускорение", общия токен „стойност" и др.

Разработена е комуникацията, използваща хардуерната онтология и онтологията на потребителския интерфейс :

- Използване на XML схеми за реализацията на ADS и SDS.
- Показва се примерен „от-EA ADS", генериран по време на оперирането на HRI"от-EA ADS"
- Представя информацията достъпна в „от-поток"-а на EA агента. В този случай той се отнася към позициите на управляващите мотори и скоростите на движещите мотори. [5]

2.3.1.5. **Приложение**

Използване на RoboML (XML) схеми за реализацията на ADS и SDS

```
<xsd:element name="set" type="SetPerformative"/>
<xsd: complexType name="SetPerformative">
   <xsd:element name="robot" type="Robot"/>
   <xsd:attribute name="sender" type="xsd:string"/>
   <xsd:attribute name="receiver" type="xsd:string"/>
   <xsd:attribute name="ontology" type="xsd:string" fixed="Hardware"/>
</xsd:complexType>
<xsd:complexType name="Robot">
   <xsd:element name="wheel" type="Wheel" minOccurs="4" maxOccurs="4"/>
   <xsd:attribute name="name" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="Wheel">
   <xsd:element name="motor" type="SteeringMotor" minOccurs="1" maxOccurs="1"/>
   <xsd:element name="motor" type="DrivingMotor" minOccurs="1" maxOccurs="1"/>
   <xsd:attribute name="name" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="SteeringMotor">
   <xsd:element name="position" type="xsd:decimal" minOccurs="1" maxOccurs="1"/>

</xsd:complexType>
<xsd:complexType name="DrivingMotor">
   <xsd:element name="velocity" type="xsd:decimal" minOccurs="1" maxOccurs="1"/>

</xsd:complexType>
```

RoboML съобщение

```
<set sender="AGV1" receiver="MyInterface" ontology="Hardware">
        <robot name="AG1">
                <wheel name="1">
                        <motor name="steering motor">
                                <position>2577</position>
                        </motor>
                </wheel>
                <wheel name="2">
                        <motor name="steering motor">
                                <position>-754</position>
```

```
            </motor>
            </wheel>
        </robot>
</set>
```

RoboML заявка

```
<get sender="MyInterface" receiver="AGV1" ontology="UserInterface">
…
</get>
```

2.3.1.6. **Заключение**

Предложен е общ интернет ориентиран език за комуникация на агента и програмиране на робота. XML базираният език за роботни апликации RoboML е заложен, за да се използва в комбинация с други интернет стандарти включително и други базирани на XML езици. Предложената архитектура и език са включени в HRI за автономно управляван прототип. Хардуерът и юзер интерфейсът са създадени за експериментална и HRI конфигурация. Въпреки, че

предложените методологии са тествани успешно, все още има много за надграждане за ясна спецификация на прокси медиираната HRI архитектура и RoboML.

Проектът *Интерфейс човек-робот*, използващ комуникация на Агенти чрез XML базирани езици е тясно свързан със създаването на шаблонни интернет стандарти за XML базирани езици и протоколи, както и за усъвършенстване на HRI, комуникацията на агентите, представяне на знания и програмиране на роботите. Прокси-медиираният HRI модел цели да намали натоварването на комуникацията и изисленията на embedded агентите и агентите на потребителския интерфейс. Да предостави средства за асинхронна обмяна на данни, подпомагайки embedded агента работещ в реално време и може би агента на потребителския интерфейс. Съществуващата схема на данните и абонаментната схема на данните, изразени чрез XML схемата, се поддържат от прокси агента за всеки от потоците от данни. Предлага се един общ Интернет-базиран език за агентна комуникация, представяне на данните и програмиране на роботите – RoboML.

- Проблеми при използването на Интернет като среда за комуникация на интерфейсите човек-робот:

- Комуникацията между системите в реално време чрез канал с ниски гаранции за качеството на услугата.
- Ограничени или несигурни изчислителни ресурси достъпни за интерфейсния клиент и за embedded софтуера.
- Съгласуваност с Интернет стандартите и съвместимост със съществуващия софтуер.

- Възможни следващи направления за работа върху RoboML включва:

- Спецификация на онтологиите на архитектурите за управление на роботите, мултимодални потребителски интерфейси и програмиране на роботи.
- Развитие на поддръжката на RoboML от приложения чрез аплети и plug-in-и за браузъри.

2.3.2. **Описание на XRCL**

2.3.2.1. **Въведение**

XRCL (The Extensible Robot Control Language) е относително прост, модерен език базиран на XML. Дизайнът му е съобразен със средата, за да позволява на изследователите на роботи да споделят идеи чрез предоставане на кода. Това е open source проект под протекцията на GNU Copyleft (http://www.gnu.org). Визията на езика XRCL е смесица между XML and C++.

XRCL е две неща:
- спесификация на приложен език
- среда за интерпретиране на езика

Три са основните причини XRCL да се базира на XML
- възниква като един стадартен метод за структурно представяне
- осигурява много помощни инструменти (например: parsers, editors)
- станал известен и популярен сред програмисти и не прогрмисти

2.3.2.2. **Дизайн**

XRCL е хибридна система състояща се от „поведения” на размита логика (fuzzy logic) от една страна и глобални "крайни състояния на машините" („Finite State Machines" - FSM) от друга страна. На ниско ниво тези „размити” правила се описват по следния начин :

If too_close_to_wall THEN Speed STOP;
If ! too_close_to-wall THEN Speed FAST;

„Размитите” правила са обединени в малки nuggets наричани "пакети". Те не са нищо повече от събраните „размити” правила. От друга страна, ние имаме така наречената Finite State Mashine (FSM), която всъщност е само графи на възли и указатели. Възлите, които ние ще наричаме "състояния" представляват набор от поведения, а указателите представляват прехода от едно състояние в друго.

Преходите са съставени от правила (възможно и „размити”), които инструктират контролния механизъм, кога да напусне едно състояние и да влязе в друго. „Размитите” правила казват на робота как да се оправя с малки неща като: какво трябва да се прави, когато се натъквне на препятсвие. FSM очертава общ план, който робота трябва да следва.

2.3.2.3. **Поведение базирано на контролер на „размита” логика**

Традиционната логика ни позволява да извличаме неща от верните изявления. Например:

If too_close_to_wall THEN Speed STOP;

If ! too_close_to_wall THEN Speed FAST;

Можем да определим колко бързо трябва да върви робота. Това е, „if too_close_to-wall” е „TRUE”, тогава скоростта ше бъде „STOP”, но ако „too_close_to_wall” е „FALSE”, тогава скоростта ше бъде „FAST”. Традиционната логика може да ни заведе доста далече, но е много безкомпромисна. Как да пресмятаме какво „too_close_is”? Роботът може внезапно да се движи прекалено бързо, когато се отдалечава от нещата, но след това рязко да спре, когато наближава обекта. Ако

се интерпретират тези правила с „жива” логика, следователно тогава наистина само се нарежда какво да направи в определен момент - казва „спри”, когато стига на определено разтояние от обекта.

Така няма много за управление. Но как роботът ще тръгн е отново?
В XRCL, ние използваме логика с „малък тласък” (slight twist). Вместо да изискваме абсолютна стойност на „TRUE” и “FALSE”, ние използваме условна истина. Следователно ако условието е само частично вярно, тогава правилото има само частични последствия. В този смисъл, ако приближаваме стена „too_close_to_wall” става частична истина и скоростта на робота се забавя частично.
В XRCL „размитите” правила са предствавени в секциите поведение на котролера. Тези поведения са малки възли котролен код, написани за да изпълняват специфични задачи и са замислени, за да се използват в комбинация с други поведения. Всяко поведение се съчетава основно от C++ код и компилация от „размитите” правила.

Всички „размити: правила заемат формата

- IF [fuzzy value] THEN [affecter] [amount]
  [Fuzzy value] е “размита” стойност 0 или 1

2.3.2.4. **Семантика**

Както в XML декларацията така и тук всеки отварящ таг се асоциира с затварящ таг.[7]

- Основни тагове

XRCL	определящ таг
ROBOT	определя типа на робота
BEHAVIOR	указва ново „поведение”
OBJECT	дефинира обект (като невронна мрежа, blob и др.)
MEMBERS	добавя променлива стойност към „поведението”

ARG добавя аргумент към „поведението”
UPDATE код, който да се стартира на всеки времеви интервал
ACTIVATE код, изпълняващ се при стартиране на „поведението”
DEACTIVATE код, изпълняващ се при спиране на „поведението”
TIMEOUT код, който се стартира при изтичане на времевия интервал

- Общи тагове

INCLUDE използва се за зареждане на C++ хедъри
LIB използва се за зареждане на C++ библиотеки
CLEANUP деконструктор, за изтриване на „поведение”
PREINIT стартира код, когато е създадено „поведение”
PO STINIT код, стартиращ се след създаването на всики „поведения”, и точно преди стартирането на системата

2.3.2.5. **Пример**

```
<controller>
<robot type="B21R">elektro</robot>
<behavior name="Goto">
        <arg default_value="0.0"
        type="double">my_x</arg>
<arg default_value="0.0"
type="double">my_y</arg>
<arg default_value="0.15"
type="double">my_close</arg>
<update>
        static int done = 0;
        if (! done) {
        xcSay("Here I go!");

        done = 1;
        }
Fuzzy euclidean(0.0, 1.0);
// градусни на завъртане на ляво:
```

```
Fuzzy left(5, 20);
// градуси на завъртане на дясно:
Fuzzy right(-5, -20);
double phi = xcAngle(my_x, my_y);
double dist = xcDistance(my_x, my_y);
Fuzzy too_left = right >> phi;
Fuzzy too_right = left >> phi;
Fuzzy near_goal = euclidean << dist;

// правила на „размита” логика

IF too_left THEN Turn -.30;
IF too_right THEN Turn .30;
IF ! (too_left || too_right)
THEN Speed xcSLOW*1.1;
IF (near_goal || too_left || too_right)
        THEN Speed xcSTOP;
TurnEffect(!near_goal);
SpeedEffect( 1.0 );
// within my_close cm
if (dist < my_close) {
xcPrint("\n[I'm there!]");
xcSay("I am now at the
specified location.");
xcReturn();
}
</update>

</behavior>
</controller>
```

2.3.2.6. **Употрева на XRCL в проекта *Elektr***

Elektro (фигура 3) е екипиран с 3 главни сензора: инфраред, лазер, сензор.
Също така притежава и стерео визуална система екипирана с две камери с висока резолюция монтирани на плот въртящ се на 180 градуса. „Мозъкът” представлява стандартно PC с 400MHz Pentium II с 128MB RAM и 9 GB хард диск монтирани на роботът. Компютърът също притежава стандартни периферни устройства като звукова карта и колонки.

(*Фигура 3)*

Изборът на XRCL в проекта *Elektro* се базира на неговата „размита” логика и способност да бъде надграждан над XML [8].

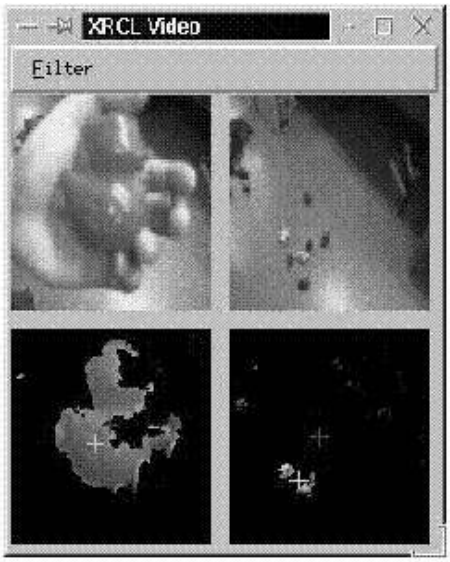
За изграждането на робота са използвани следните системи:

- Blob object system
- Neural network object system
- XRCL GUI Interface

2.3.2.6.1. **Blob object system**

Системата е създадена да осъществи лесен достъп до видео данните от робота. Blob представя двумерен масив от черни и бели пиксели получени с филтри, чрез които ние получаваме видео данните в реално време. В белите секции на blob е визуализаирана информацията, която ни трябва. Докато черните секции са

основата. Филтърът може да бъде прост, изобразявайки даден цвят или комплексен за детектване на нелинейно невронна схема на лице. *(Фигура 4)*. показва пример за такива филтри.



Фигура 4: Фигурата показва синьо пате в човешка длан

Веднъж създаден, филтърът може да намира асоциираният blob навсякъде в XRCL чрез асоциации към специфични полета на blob-а. Например ако имаме набор от филтри, които засичат синьото пате, тогава на базата на тях могат да бъдат създадени „размити" логически правила базирани на филтър „синьо пате – център на тежестта" и филтър „ синьо пате – присъствие ".

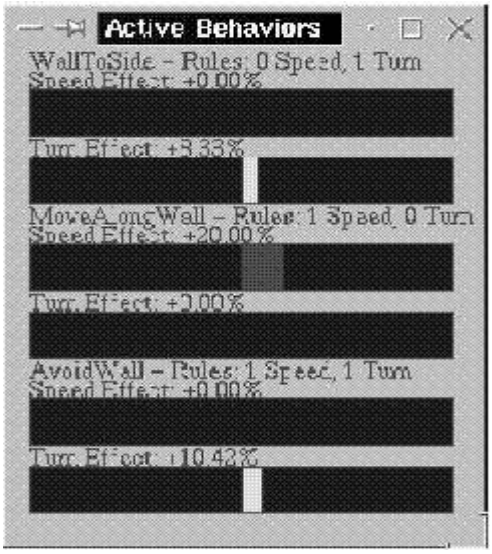
2.3.2.6.2. **Neural network object system**

Като допълнителен тест за гъвкавостта на XRCL е изградена невронна мрежова система, което е позволило да се създаде интерфейс за мощна самообучаваща система. Тази система би могла да бъде използвана в комбинация с филтър от blob системата или като данни на „размитите" правила.

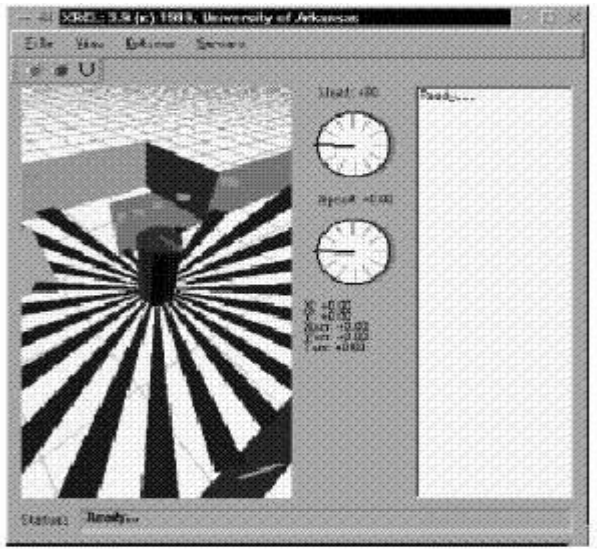
2.3.2.6.3. **XRCL GUI Interface**

За визуализация и управление на контролерите в XRCL се използва графична среда. За тази цел е създаден интерфейс с множество нива на изображение и

вградена X Windows system (*Фигура 5*).



Фигура 5 :
XRCL Система за графично изобразяване



3D графичен поглед получен чрез X Windows system

2.3.2.6.4. **Заключение**

Потребителският интерфейс и комуникацията на робота с хората е интерактивна концепция, която употребява общия наблюдаван виртуален свят. И тъй като *Elektro* е сбор от системи - манипулатор, навигационна система, сензорни системи и.т.н., то целта е да се направят тези субсистеми да работят заедно, управлявани от обща система, като да е лесна за използване и прилагане и в други проекти.

2.4. Езици за обработка и изображения базирани на XML

2.4.1. **Описание на SVG**

2.4.1.1. **Въведение**

Scalable Vector Graphics (SVG) е графичен формат за векторни графики, създаден от консорциума W3C. SVG е XML-базиран език за описание на двуизмерна графика. Едно от основните предимстава при използването му е, че не се губи

качество при мащабиране на изображението или при промяна на размера. Освен това SVG документът е със сравнително малък размер, отворен стандарт, работи с Java технологии. SVG файловете могат да се четат и редактират с различни средства (например Notepad). Текстът в SVG може да бъде селектиран, избиран и може да се осъществява търсене, което дава отлична възможност за разработка на карти . Съвместим е и с други стандарти като EXtensible Stylesheet Language (XSL) и Document Object Model (DOM), поддържа Cascading Style Sheets2 (CSS2) и други ключови технологии. Той е проектиран за използване и като формат за автономни документи, като в този случай SVG файлът представлява зъвършено изчертаване, схема или диаграма и може да включва връзки към други разработки. Главният недостатък обаче на SVG е липсата на пълна поддръжка от страна на браузърите. В бъдеще и този проблем може да бъде решен, тъй като Mozilla и Microsoft планират пълната му поддръжка.
Основният конкурент на SVG е Adobe Flash. Двете технологии имат много подобни черти, но най-голямото предимство на SVG е съвместимостта му с други технологии - XSL и DOM. Flash не е отворен стандарт [9].

2.4.1.2. **Дефиниране на документ**

Има три начина за дефиниране на SVG в един уеб документ.

- Като самостоятелна (Standalone) SVG страница.
- Като прикрепен/допълнителен (embedded) елемент .
- В XHTML документ с namespace декларация.

2.4.1.3. **Standalone страница**

Следващият пример дефинира самостоятелен SVG файл. Той трябва да бъде съхранен с svg разширение.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="300" height="300" x="0" y="0">
....
</svg>
```

Първият ред съдържа XML декларация, понеже SVG е XML приложение.

Вторият и третият ред дефинират DTD (Document Type Declaration). Където DTD описва езика и синтаксиса допустими в SVG.
Четвъртият ред съдържа тага <svg> - “указва” на браузъра, че това е SVG документ. Площта на SVG документа се дефинира чрез атрибутите width и height. Ако не се предварително дефинират тези атрибути ще запълнтя цялото пространство на браузъра, x и y определят къде ще бъде разположена svg площта в прозореца на браузъра. В петият, шестият и седмият ред се разполагат SVG елементите.
С последният ред </svg> затваря документа.

2.4.1.4. **SVG като прикрепен елемент**

Тъй като SVG е XML базиран, повече от машините за търсене няма да намерят самостоятелна SVG страница. За да разрешим този проблем можем да добавим SVG като прикрепен елемент в Extensible HyperText Markup Language (XHTML) страница.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<body>
<object data="test.svg" width="500"
height="500" type="image/svg+xml">
<embed src="test.svg" width="500"
height="500" type="image/svg+xml" />
</object>
</body>
</html>
```

Тук се употребяват и двата тага <object> и <embed>. Според XHTML стандарта трябва да използвате само <object> тага, но тъй като Netscape не го поддържа, най-добре е да използваме едновременно и двата. Използването им в документ ще допринесе за намирането им от машините за търсене. Той спомага за по-лесното интегриране и на други обекти (например: звук, музика, Flash) в същата уеб страница.
SVG разполага с предварително дефинирани форми, които могат да бъдат използвани и манипулирани от разработчиците.

2.4.1.5. **Форми**

- Правоъгълник - <rect>
- Окръжност - <circle>
- Елипса - <ellipse>
- Линия - <line>
- Многоъгълник -<polygon>
- Polyline - <polyline>
- Path - <path>

Правоъгълник

Тагът <rect> се използва, за да се създаде правоъгълник и вариации на правоъгълна форма.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="300" height="300">
<rect width="300" height="100"
style="fill:rgb(0,0,255);stroke-width:1;
stroke:rgb(0,0,0)"/>
</svg>
```

Обеснение на кода:

- Атрибутите width и height на елемента rect определят ширината и височината на правоъгълника;
- Атрибутът style се използва, за да дефинира CSS свойставата;
- CSS свойството fill дефинира цвета, с който ще се запълни правоъгълника (има стойност rgb или шестнадесетични числа);
- CSS свойството stroke-width определя ширината на рамката на правоъгълника;
- CSS свойството stroke определя цвета на рамката на правоъгълника’ КАРТИНКА

Окръжност

Тагът <circle> се използва за създаването на окръжност

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="300" height="300">
<circle cx="100" cy="50" r="40" stroke="black" stroke-width="2" fill="red"/>
</svg>
```

Обяснение на кода**:**

- Атрибутите **cx** и **cy** определят **x** и **y** координатите на центъра на окръжността. Ако **cx** и **cy** са пропуснати, центърът на октъжността е (0,0).
- Атрибутът **r** определя радиусът на окръжността.

Елипса

Тагът **<ellipse>** се използва, за създадване на елипса. Елипсата прилича на окръжността, но при нея **x** и **y** радиусите се различават един от друг, докато окръжността има равни **x** и **y** радиуси.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="500" height="500">
<ellipse cx="300" cy="150" rx="200" ry="80"
style="fill:rgb(200,100,50);
stroke:rgb(0,0,100); stroke-width:2"/>
</svg>
```

Обяснение на кода:

cx атрибутът определя x координатата на центъра на елипсата;

- cy атрибутът определя y координатата на центъра на елипсата;

- rx атрибутът определя хоризонталния радиус;
- ry атрибутът определя вертикалния радиус;

2.4.1.6. **Оцветяване на фигурите**

Всички SVG фигури използват два атрибута за оцветяване “stroke” и ”fill”
Stroke - използва се за очертаване на външната граница на фигурата.
Fill - определя вътрешния цвят на фигурата.
Тези атрибути имат стойности на цвета, който включва цветове и две “non-color” стойности:
none – не се вижда цвят
currentColor – използва се за дефиниране на цвят.

2.4.1.7. **Path**

Тагът <path> се използва, за да се дефинира път. Той е единственият най-полезен елемент за създаване на професионално изглеждащи SVG документи и е най-

трудният за ръчно писане в кодът на SVG. Той е доста труден и за визуализация.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="100%" height="100%">
<path d="M250 150 L150 350 L350 350 Z" />
</svg>
```

В този пример се дефинира път, който започва в позиция 250 150 и се изчертава линия до позиция 150 350 и от тук до позиция 350 350 и накрая пътят се затваря обратно в позиция 250 150.

2.4.1.8. **Филтри**

Един SVG филтър тряба да се дефинира в <defs> тага. Използват се за ефекти на множества от линии, които комбинирани с двумерната сила на SVG може да опишат повече от обикновена графика в уеб по такъв начин, че потребителското генериране и изменение да бъде извършено лесно. Ефектът на филтъра се

състои от серия графични операции на графики, които са приложени към даден източник, за да създаде променен графичен резултат.

2.4.1.9. **Градиент**

Един SVG градиент трябва да бъде дефиниран в <defs> тага. Градент е плавното преминаване от един цвят към друг. Няколко цветови преминавания могат да бъдат приложени към един елемнт.

В SVG има два типа градиенти:

- Линеарен градиент – тагът < linearGradient> се намира между таговете <defs> и </defs>. Defs е съкращение от definitions и дава възможност за дефиниране на специални елементи като градиентите. Линеарните градиенти могат да се дефинират като хоризонтални, вертикални и ъглови .
- Радиален градиент - тагът <radialGradient> се намира между таговете <defs> и </defs>.

2.4.1.10. **Приложения на SVG**

SVG има приложения в много области. През 2001 година индустрията на мобилните телефони избра SVG за основа на нейната графична платформа . Много водещи компании, сред които Nokia и Motorola, обединяват усилия, за да разработят SVG Mobile спецификация, чиято цел са ресурсно-ограничените устройства като мобилни телефони и PDA устройства (Personal Digital Assistant - мобилни компютри).

- Печат

Комбинацията от богати графични възможности, пълна текстова поддръжка и независимост от резолюцията превръщат SVG във формат подходящ за печат. Водещи компании в областта на производството на принтери понастоящем разработват SVG Print спецификация: версия на SVG подходяща hard-copy output.

- Уеб приложения

Разработчиците често са ограничени от липсваща функционалност и от несъвместимостта на браузърите. С мощността на скрипт и event поддръжка SVG може да се използва за платформа на базата, на която да се разработят

много/rich графични и потрребителски интерфейси. Освен това разработчиците използват съвкупност от отворени стандарти, което не ги обвързава със специфично изпълнение, компания или средства за разработка.

- GIS и картография

Goegraphic Information System (Географска информационна система) има много специфични изисквания: богати графични характеристики, поддръжка на векторно и растерно съдържание и възможностза за манипулиране на голямо количество данни. SVG предлага добри възможности за този пазар и много GIS системи осигуряват SVG експорт.
SVG е иделно допълнение към GML формата на OpenGIS консорциум. GML също е XML-базиран език, който описва географски елементи като реки и пътища и може да бъде конвертиран в SVG .

За да започнете да работите с SVG се нуждаете от текстов редактор и SVG Viewer

The Adobe SVG Viewer;
The Batilk SVG Toolkit
The CSIRO SVG Toolkit

2.4.2. **Описание на VRML**

2.4.2.1. **Въведение**

Езикът за моделиране на виртуална реалност Virtual Reality Modeling Language (VRML) е файлов формат, специално разработен за описание на интерактивни триизмерни обекти и светове. Първите разработки за създаване на триизмерен интерфейс за мрежата започва още през 1994 г. С времето проектът се усложнява и развива в пълноправен език, който е одобрен за създаване на 3D графика в Интернет. VRML е замислен за използване в Интернет за местни клиентски системи, както и за универсален формат за размяна на цялостни триизмерни графики и мултимедия. Крайният резултат, наричан още 3D свят, при интерпретирането на езика е обикновен текстов файл с разширение .wrl, който се изтегля от браузъра и се интерпретира на място. Потребителят има възможност да "оглежда" обектите, съставящи този "свят" интерактивно с помощта на мишката. Те могат да бъдат завъртани във всички посоки, отдалечавани или приближавани. Понеже VRML не е просто добавка към стандартния уеб, за гледането на "виртуални светове" не можем да използваме само обикновените и

познати на всички ни браузъри. Налага се да прибегнем до специални plug-in-и (пример) за тях или към специални VRML-"браузъри"[10].

2.4.2.2. **Самостоятелни VRML браузъри**

Те практически представляват външни програми, чрез които могат да се гледат виртуалните светове или какъвто и друг вид VRML-триизмерности из мрежата. Принципът им на действие е много подобен на известния RealAudio плейър. Ако при навигация с обикновен браузър стигнем до 3D, програмата автоматично се пуска.

2.4.2.3. **Замисъл**

VRML е замислен да изпълнява следните изисквания :

- Да позволява развитие за създаване на компютърните програми, редактиране, и поддържани файлове на VRML, както и автоматични

  програми на транслация, за да може да се превръщат други обикновено използвани триизмерни файлови формати в файлове на VRML.

- Да предоставя използването и комбинирането на динамични триизмерни обекти в рамките на света на VRML и по този начин да има преизползваемост.
- Да предостави възможност за добавяне на нов тип обект, не изрично дефиниран в VRML.
- Да бъде способен за реализация върху широк обхват на системи.

2.4.2.4. **Характеристики**

VRML е способен да представя статични и анимирани динамични триизмерни обекти и мултимедиини обекти с хипервръзки към друга медия като текст, звуци, филми, и изображения. За различните платформи са налични VRML браузери и инструменти за създаване на VRML файлове. VRML поддържа разтеглив модел (extensibility model), който позволява на нови обекти да бъдат дефинирани, за

разширение на основния стандарт. Има връзки между обекти на VRML и обикновеното триизмерно приложение (програмен интерфейс). VRML е език за описание на “сцена”. Той е компютърен език, а не е език за програмиране. VRML файловете не са компилирани, а са елементарни ASCII текстови файлове, които могат да бъдат анализирани от VRML интерпретатор. Тези интерпретиращи програми (parsers) са често наречени VRML браузери.

2.4.2.5. **VRML светове**

Терминът е получил названието си от основната цел на езика - споделени виртуални светове в Интернет. Те могат да бъдат отделни файлове или групи, които се зареждат едновременно. Могат да бъдат само обекти или сложни сцени.За създаване на VRML файл може да се използва текстов редактор (например Notepad) или средствата, предназначени за тази цел. Файлът е с разширение .wrl и се запазва като обикновен текст. Добре е да се работи с обикновен VRML, но така е трудно да се създадат сложни сцени и това налага използването на средство за създаване на VRML сцените или програма за моделиране на триизмерна графика. За тази цел е необходим транслатор за

превръщане на форматите от използваните програми в VRML формат (например Crossroads).

Файл на VRML се състои от следващите главни функционални компоненти :

1. заглавната част- Heder
2. сцена
3. прототипи
4. рутиране на събития

2.4.2.5.1. **Семантика**

Scene graph - или “сцена” съдържа възли, които описват обекти и техните свойства. Това е йерархично групирана геометрия, която осигурява аудиовизуално представяне на обекти, както и възли, които участват в event generation и routing mechanism (движещия механизъм).

Обектите в VRML файла се наричат възли**.** Когато са подредени йерархично те съставят “сцена”**.** Всеки VRML свят, колкото и прост да е, е граф сцена. Възлите са три типа:

- Shape nodes – описват геометрични фигури
- Property nodes – променят геометрични фигури
- Grouping nodes – групира обектите в един

Събирателният ефект на възлите в графа сцена се нарича „състояние” (state).

Групови възли:

- Anchor
- Billboard
- Collision
- Group
- LOD
- Switch
- Transform

Геометрични възли:

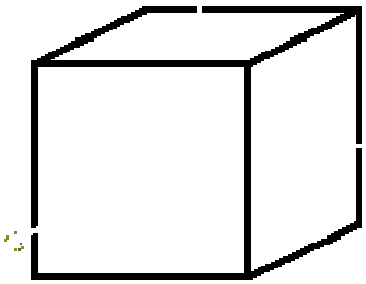
- Box
- Cone
- Cylinder
- ElevationGrid
- Extrusion
- IndexedFaceSet
- IndexedLineSet
- PointSet
- Sphere

Възли, които могат да се свързват:

- Background
- Fog
- NavigationInfo
- Viewpoint

Пример

Кутията е куб или кубоидна форма.



```
geometry Box {
    size 5.5 3.75 1.0
}
```

2.4.2.6. Уеб базирана симулация на Робот - манипулатор с помощта на VRML

За VRML не е нужен специализиран и скъп хардуер и софтуер. Произволен уеб браузър с VRML viewer е способен да поддържа тази програма което прави приложението независимо от хардуерната платформа. Заедно с Java и VRML External Authoring Interface (EAI) може да бъде използвана за визуализиране и телеуправление на реални роботи. Когато се говори за приложения на симулацията, обикновено се мисли за големи монолитни програмни пакети, които са инсталирани на работна станция (workstation), работеща самостоятелено (stand-alone). Когато VRML 1. 0 е заменен с по-гъвкавият VRML 2. 0, се предоставя за първи път възможността да се създават малки, но сложни триизмерни сценарии, които да работят в уеб среда. В този насока започва обсъждането на този проект, за да бъдат оценени възможностите да се имплементира уеб базирана среда за симулация и телеуправление на индустриални роботи.

Симулацията трябва да се възползва от всички предимства на езика. Резултатът трябва да бъде платформено независим софтуер, който работи изцяло в браузър, същевременно трябва да бъде достатъчно малък, за да бъде зареден през уеб сървъра всеки път когато е необходимо. С изключение на VRML plug-in не е необходима никаква друга инсталация. Тъй като пълният потребителски

интерфейс е внедрен в областта за визуализация, също така трябва да се възползва от допълнителна степен на свобода и да бъде по-интуитивен от съществуващите интерфейси. Отделено е специално внимание за изграждането на ефективен и лесно ползваем интерфейс за стандартни насочващи устройства. Функционалностите предоставени от 3D потребителският интерфейс са лесни за ползване, тъй като са използвани самообясняващи се символи вместо език. За дистанционното управление на индустриални роботи трабва да бъде установена връзка между контролната станция и реалния манипулатор. Най-удобният начин за достъп до стойностите в VRML сцена е посредством Java аплет. Също така може да се използва External Authoring Interface (EAI), който може да регистрира аплета и да бъде уведомен при настъпване на събития и съответно събития да бъдат генереирани и изпращани към виртуалният свят.

Механиченият манипулатор може да е моделиран като отворена верига с няколко твърди връзки свързани последователно с линейни или ротационни стави. Единият край е закрепен неподвижно върху носеща повърхност, а другият е свободен свързан с ефектор, (end-effector) за манипулиране на обекти. Движението на ставите е резултат от движението на връзките, позицията и ротацията на ефектора. [11]

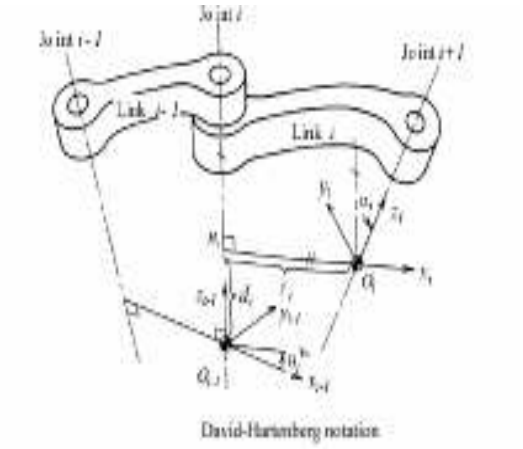
2.4.2.6.1. **Дефиниране на обекта робот**

За да дефинираме абстрактен обект е необходимо детерминирането на входните и изходните параметри, както и характеристиките, описващи един конкретен пример за този обект. Индустриалният робот е инструмент за позициониране в ограничено работно пространство. Инструментът се монтира на края на подвижната си ръка и когато говорим за положението на инструмента или на робота винаги ще се отнасяме към началото на координатната му система. Ръката на робота се състои от съвкупност от линейни или ротационни стави свързани от сегментите на ръката. Различните стави имат различна степен на свобода което позволява движение в различни посоки, за да бъдат достигнати всички точки в 3-мерното работно пространство.

Степените на свобода описват гъвкавостта на движението. Механизъм, който има пълна свобода на движение има 6 степени на свобода:
3 за транслация – способността да се движи във всяко от трите измерения
3 за ротация –способността да се променя ъгъла около трите оси.

В роботиката степените на свобода изразяват броят на посоките, в които роботът може да завърти или премести става. Човешката ръка има 7 степени на свобода. За да се движи свободно в пространството роботът трябва да има поне 6 стави. Тъй като линейните стави не са способни да променят ориентацията поне три от ставите трябва да бъдат ротационни. В този проект се предполага че роботите имат 6 ротационни стави, така наречените 6R манипулатори.

Роботът може да бъде дефиниран чрез David-Harteneberg (DH) параметри. Този набор от параметри описва разполагането на ставите и е базата на кинематичното пресмятане. За всяка става са необходими 4 стойности : a, d, α и θ.



- Дължина на връзката (an): най-късото разстояние между две ставни оси на двата края на Връзка n Завъртане на връзката (αn): ъгълът на завъртане между две z-оси на двата края на Връзка n, в посоката на xn осите
- Отместване на връзката (dn): разстояние между началото на фрейм n-1 до фрейм n по посока на z n-1 оста
- Ъгъл на ставата (Θn):ъгълът между x n-1 и xn осите, по посока на z n-1 оста.

Допълнително определяме стойностите на max и min на ставната ротация. Тъй като те не се променят можем да дефинираме област от шест float променливи за всеки параметър.

Съществуват 2 възможности да се премести механичната ръка в ново положение:

- стойностите на променливите да бъдат зададени директно (Joint Space), което ще се отрази на положението на инструмента и неговата ротация
- да бъде зададена целевата позиция или ротация и обектът да пресметне и зададе подходящите стойности(Motion in Cartesian Space).

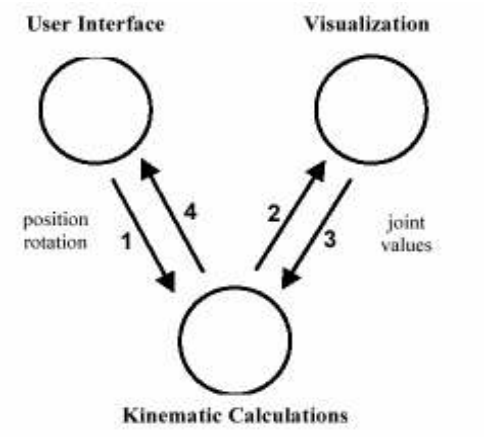
Състоянията на ставите са представени от 6-мерен вектор със стойности от тип MFFloat, вместо с матрица от 3x4 са използвани VRML типовете данни SFVec3f и SFRotation за позициите и стойностите на ротация. Когато е получено входно събитие eventIn, променените стойности ще бъдат изпратени от eventOut, тъй че поведението на обекта е следното:

```
1. IN set_position  >  OUT joint_changed
2. IN set_rotation  >  OUT joint_changed
3. IN set_joint     >  OUT position_changed +
                       OUT rotation_changed
```

2.4.2.6.2. **Имплементиране на VRML обекта робот**

Задачата трабва да бъде разделена на 3 отделни проблема:

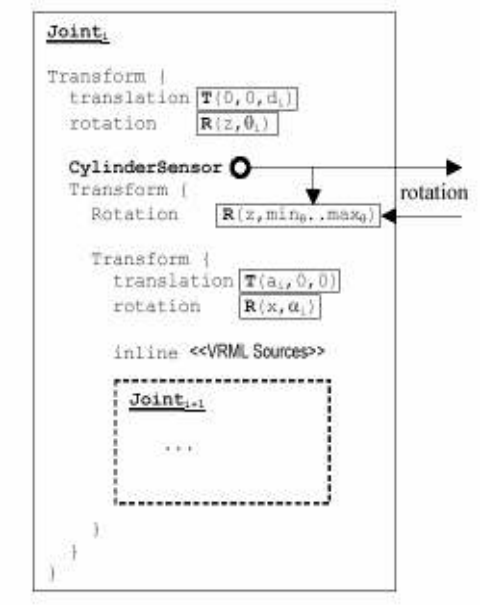
- обект за визуализация - да изобрази модела на екрана
- обект за пресмятане - да конвертира позицията и ротацията в ставни стойности с обратна кинематика
- интерфейс - който позволява на потребителя да управлява робота с 6 степени на свобода, използвайки стандартно 2D посочващо устройство. Предаването на данните между модулите е показано в следната диаграма:



2.4.2.6.3. **Визуализация**

Всеки сегмент на ръката се движи в координатна система определена от позицията на предишния сегмент. Веригата може да бъде моделирана чрез дърво от възли, където всеки сегментен възел е наследник на предишния. Разстоянията

за асемблиране на веригата са взети от DH параметри, които също трябва да бъдат подадени на модула като вход. Входът към VRML прототипа rob_geo е MFFloat масив от 6 стойности. Когато е получен eventIn се извиква скрип, който разпределя стойностите между ставите. За тази цел се генерира SFRotation eventOut и се изпраща към съответната става.



Структура на сегмент на ръка във VRML

Управлението на стойностите в ставите от потребителя се осъществява чрез хващане на ръката в произволен сегмент и завъртане на съответната става. Генерираните събития са директно насочени към ставата и също така ще бъдат изпратени към скрипта който ще впише отново стойностите в масив. Този масив е изхода на rob_geo модула.

Прототипът е дефиниран както следва:

```
PROTO rob_geo [
  eventIn  MFFloat    set_joint

  eventOut MFFloat    joint_changed

  field    MFFloat    a       [0,0,0,0,0,0]
  field    MFFloat    d       [0,0,0,0,0,0]
  field    MFFloat    alpha   [0,0,0,0,0,0]
  field    MFFloat    theta   [0,0,0,0,0,0]
  field    MFFloat    min     [0,0,0,0,0,0]
  field    MFFloat    max     [0,0,0,0,0,0]
]
```

2.4.2.6.4. **Кинематични изчисления**

Директна кинемтика - при дадени ъгли на ставите и дължини на всяка връзка, да се намери позицията на подвижния край.
Обратна кинематика - при дадена позиция на подвижния край и дължината на всяка връзка да се намерят ъглите на ставите.

Почти всяко взаимодействие с модела произвежда множество пресмятания извършени от този компонент. Въпреки математическата си сложност тази задача е много проста. Позицията и ротацията трябва да бъдат преобразувани в стави и ставите в позиции и ротация. т.е модула съдржа два скрипт възела: един за директна и един за обратна кинематика. Друг възел получава входните събития на модула и извиква нужните пресмятания. Всички те са базирани на DH параметрите.

```
PROTO rob_kin [
  eventIn  SFVec3f    set_position
  eventIn  SFRotation set_rotation
  eventIn  MFFloat    set_joint

  eventOut SFVec3f    position_changed
  eventOut SFRotation rotation_changed
  eventOut MFFloat    joint_changed

  field    MFFloat    a      [0,0,0,0,0,0]
  field    MFFloat    d      [0,0,0,0,0,0]
  field    MFFloat    alpha  [0,0,0,0,0,0]
  field    MFFloat    theta  [0,0,0,0,0,0]
  field    MFFloat    min    [0,0,0,0,0,0]
  field    MFFloat    max    [0,0,0,0,0,0]
]
```

2.4.2.6.5. **Потребителски интерфейс**

Потребителският интерфейс (ПИ) е колекция от 3D елементи със сензори, които позволяват на потребителя да манипулира робота в работното му пространство. Без значение кое от тези е използвано, изходът винаги е позициионна и/или ротационна стойност, която е подадена на подпрограмата за обратна кинематика. Зададената цел може да се намира извън работното пространство, стойностите на зададената позиция винаги се изпращат към интерфейса като обратна връзка. Тъй като не са необходими други променливи прототипът е много прост:

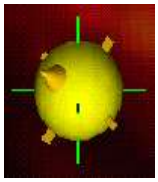
```
PROTO rob_ctrl [
  eventOut SFVec3f    position_changed
  eventOut SFRotation rotation_changed

  eventIn  SFVec3f    set_position
  eventIn  SFRotation set_rotation
]
```

2.4.2.6.6. **Схематична козола за управление**

Конзола за управление

Предоставя няколко независими от изгледа инструмента

връзка към страницата



променя ориен-тацията на коор-динатната с-ма на инструмента

променя ориентацията на обекта спрямо фиксирани оси

включва и изключва контролния елемент

ръката се завърта около xy равнината на текущото пространство

извиква интерактивните панели за уравление на тракекториите

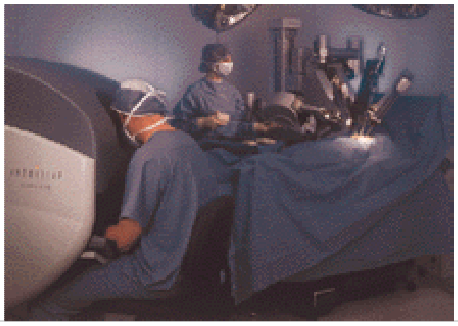
2.4.2.6.7. **Заключение**

Като език на много високо ниво, VRML може да работи на каквато и да е основна платформа. Единственото изискване за пускане на програмата е уеб браузерът да е с инсталиран plug-in, което е безплатно. И така, никакви разходи за допълнителен специализиран хардуер или софтуер. Напълно триизмерният интерфейс главно е самопояснителен и лесен за използване. Това прави програмата също интернационална, защото функционалността е показана от знаци вместо текст. Друг плюс е, че контролерът свободно избира неговата гледна точка. Въпреки че, в момента видеофилм предаващ движенията на истинският робот на потребителят е все още нужен, то в бъдещето е възможно изграждането на цялостна виртуална реалност, която от своя страна драстично ще ограничи изискванията към скоростта на връзката нужна за телеопериране

3. Хирургически телероботи

3.1. Описание

Хирургическите телероботи използват системата “господар-подчинен” при която човекът е директно част от цикъла на управление, командвайки действията на робота от дистанция. Първите механично свързани “господар-подчинен” системи еволюират в днешните комплаксни електро-механични телероботни системи и по този начин отварят пътя на сложни компютърно подпомагани операции. Ключово постижение при тази технология е, че отдалеченият робот изпълнява прецизно командите на оператора. С въвеждането на тази система, телероботите заемат мястото между хирургическите ръце и пациента, и повишават хирургическата сръчност в деликатни операции. *(фигура 1)*



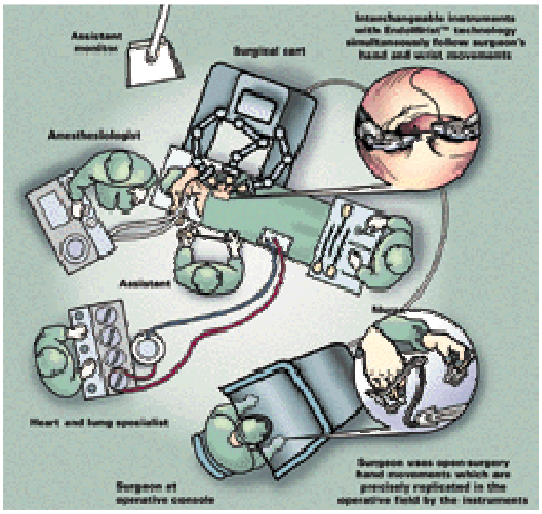
Фигура 10 Хирург оперираш чрез телеробот

Компютърно управляваните диагностични уреди са използвани в продължение на много години за предоставяне на жизнено важна информация, чрез ултразвук, компютърно спомагателна томография и други визуализиращи уреди. Наскоро са разработени роботни системи за позициониране на камерите при Minimally Invasive Surgery (MIS) – „минимално проникваща хирургия", за поставяне на неврохирургически инструменти (NeuroMate) за насочване на косторезачки (RoboDoc).

Нарстващата нужда от намеса MIS довежда до разработването на "интуитивни телероботни системи". При този метод, хирургът прави минимален разрез в човешкото тяло и вмъква ендоскоп - стрелообразен цилиндър свързан с камера.
Хирургическите инструменти влизат в човешкото тяло през допълнителни малки разрези и се стремят към миниатюризация. Въпреки, че хирургът не може да види директно раната, той може да проследи движенията на инструментите чрез видео монитор свързан към камерата в тялото. Пред хирурзите обаче бързо изникват проблеми, при опит за осъществяване на по-сложни операции. Първият проблем идва от противоположното движение на инструмантите.
Използваните в *MIS* инструменти са понякога повече от foot* дълги. Навлизайки в човешкото тяло през малък отвор, при движение на ръката на хирурга (извън тялото) в една посока инструмента реагира в противоположната (в тялото). Въпреки, че хирургът наблюдава движенията на монитора, този нов тип интуитивни движения изисква изучаването нов вид умения. Второ, поради намалената мобилност на инструментите в тялото, подобни инструменти имат ограничена свобода на движенията в сравнение с свободна ръка, опериращa отворена рана.
В резултат на това, операциите използващи метода на минимално проникваща хирургия или *MIS* са ограничени до прости операции. Допълнително, традиционните техники вмъкват ергономично неудобство. Визуалните възприятия и механичните деиствия са разграничени и неориентирани. Това довежда до допълнителено умствено натоварване по време на операция и изисква дълъг заучаващ период за усвояване на нови техники. Но въпреки тези ограничения, MIS става много популярен при отсраняването на жлъчния мехур, най-вече защото извършва отсраняване, а не корекция, нещо което не беше възможно преди. Сравнено също с отворения метод при отсраняване на жлъчния мехур има значително увеличаване на комфорта на пациента и намаляване на следоперационното възстановяване. Въпреки, че този тип операции са

ограничени до сравнително прости интервенции, те все пак показват огромният потенциал скрит зад хирургията през малки разрези. Това довежда до търсенето на алтернативни методи, които да спомогнат този метод да бъде приложен в далеч по комплексни операции, и да направи операциите по-свойствени за хирурзите. Инженерите отправят поглед към по-комплексни хирургически системи използвайки опита придобит от телероботиката.

3.2. Схематично представяне на телехирургична операционна маса



Използвайки телехирургия, хирургът сяда комфортно пред конзолата и наблюдава операцията през най-комплексната 3D визуализираща система достъпна в днешни дни. По време на процедурата, камерата (закрепена за роботизирана ръка) може да бъде направлявана от хирурга осигуряваики му по този начин най-добрата гледна точка. Акуратната визуализация е критично важна, тъй като хирургът взема решения в хода на операцията, следваики единствено образа на монитора. Повечето от важните хирургически интервенции се извършват под многократно увеличение на образа, чрез увеличението операцията може да изглежда сравнително лесна - изобразяваики коронарната артерия (само няколко милиметра в диаметър) голяма колкото градински маркуч. Въпреки това, фактът че хирургът оперира тънка коронарна артерия остава, което изисква при голямо

движение на хирургическата ръка, сравнително малък отклик от опериращата роботизирана ръка. Това е постигнато чрез настроиките на степента на откликване (големината на реагиращото движение) на интуитивната система.
По време на процедурата ръцете на хирурга са удобно разположени пред неговото тяло насочваики апаратурата. Системата улавя неговите движения и реагира със скорост 1300 часица от секундата. Всяко едно движение на хирурга е уловено и симултантно изпълнено, чрез електромагнитно направляваната апаратура. Използвайки сензорна система и кинематичен модел на "господар-подчинен", компютърната система подава командите нужни за управлението на роботизираните ръце и за подаване на нужната информация обратно.

Наблюдаваики раната през конзолата, хирургът може да види краищата на роботизираната ръка, която се движи в зададената от него посока.

Система за направление на хирургическия телеробот

EndoWrist – инструмент от системата da Vinci

Допълнителното потапяне на хирурга във задачата е постигнато чрез "система за реагиране при допир" "force feedback", или така наречената "haptic" технология. Когато роботизираната ръка в тялото усети допир, това веднага се "докладва" на системата така, че хирургът да може да го усети сензитивно, не визуално. Възползваики се от това допълнение, хирургът може лесно да извършва дисекции и разрези дори на много малки структури.
От клинична гледна точка, тънката механична връзка наречена *EndoWrist* е ключов компонент в интуитивната хирургия. Чрез настоящата MIS, не е необичаино да забележим хирург огъваики тялото си в нестандартни пози търсеики по този начин оптимален ъгъл за зашиване на раната. *EndoWrist*, доставя тези ъгли на операционната маса, предоставяки толкова прецизно

управление, че хирургът може да промуши игла през точката на края на това изречение.
Върховете на апаратурата са направени по подобие на стандартните хирургически инструменти използвани вече в продължение на повече от 150 години в извършването на разнообразен тип хирургически задачи. Остатъкът от апаратурата е със съвършенно нов дизаин. Инструментите са стерилизиращи се, лесно подменящи се по време на операция и ценово съобразени. Възловия подход при разработката е хирургът да се чуства комфортно и в позната среда. [12]

3.3. Видове системи

3.3.1. **da Vinci**

da Vinci роботизирана ръка

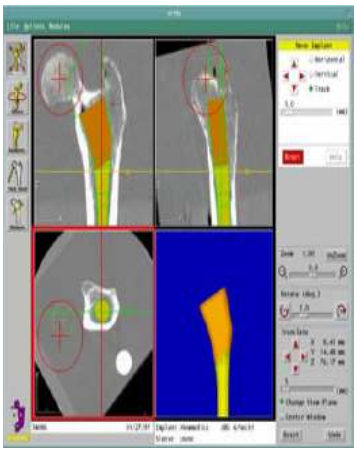
Използваики концепцията на виртуалната реалност, фирмата *Intuitive Systems* разработва системата *da Vinci*, потапя хирурзите в визуалната и осезателна среда, позволяваики им да следват опита придобит от отворената хирургия. Трите ключови елемента са - изключителна визуализация, система за навременно реагиране и акуратна дистанционно управлявана апаратура.

3.3.2. **ZEUS**

ZEUS операционна система

Системата *ZEUS* представлява ергономична хирургична конзола и три робтизирани ръце, прикрепени на хирургическа маса. Те изпълняват хирургически задачи и осигуряват визуализация по време на ендоскопската хирургия. Седнал на ергономичната конзола хирургът управлява левите и десни роботизирани ръце на системата, която в реално време препредава неговите действия на хирургическите инструменти.

3.3.3. **ROBODOC**



ROBODOC визуализираща система

Компютърно управляван хирургически робот, екипиран със специални пробивни инсгрументи и друг хардуер с цел предопрерационна подготовка на костите за протезни имплантанти. Роботът пробива отвори за

имплантантите, отстранява костния мозък и подготвя повърхностите на бедрена кост и пищял за коленни имплантанти.
Движението на работизираната ръка, екипирана с високо скоростна машина за пробиване на кости е обикновенно по-малко травматизираща отколкото съответните ръчни техники.

http://www.cs.jhu.edu/~cis/cista/445/Lectures/Robodoc903.pdf

3.3.4. **Aesop**

Aesop 3000

Aesop гласово управляван ендоскопичен позициониращ робот. Функционалността му е сравнително проста – да задвижва миниатюрна камера в тялото на пациента, в следствие на гласовите команди подавани от хирурга. По този начин системата елиминира нуждата от допълнителен човек за операциращия екип, който да насочва камерата. Това е предимство при операциите на затворен гръден кош или байпас техниките. В днешно време около една трета от всички *MIS* използват *Aseop* за управление на ендоскопите. Всяка *Aseop* машина може да извършва около 240 операции годишно.

4. Анализ на системи ZEUS и da Vinci

4.1. Описание на ZEUS

Системата *ZEUS* е създадена от ергономична хирургична конзола и три робтизирани ръце прикрепени на хирургическа маса. Те изпълняват хирургически задачи и осигуряват визуализация по време на ендоскопичната хирургия. Седнал на ергономичната конзола хирургът управлява левите и десни роботизирани ръце на системата, която в реално време препредава неговите действия на хирургическите инструменти. Третата ръка служи за визуализация на операционното поле. Уникалната система за гласово управление позволява прецизно насочване на движенията на ендоскопа, чрез прости гласови команди от страна на хирурга. Така хирургът се съсредоточва повече върху управлението на роботизираните ръце.

A Zeus System in "Operation"

Системата ZEUS в операционната зала

През 1999 г. *ZEUS* осъществява първия в историята сърдечен байпас от хирурга Дъглас Бойд. Това събитие стартира нова ера в MIS и вдъхновява както пациенти, така и хирурзи. В днешно време системата *ZEUS* вече е интегрирана в широк диапазон от процедури, обща хирургия, урология и неврохирургия.

Елементите на системата са следните: видео конзола, главен видео монитор не по-голям от 23x23 инача, допълнителен монитор и др [15].

Конзолата за управление на хирурга се състои от: монитор управляван чрез допир (Touch screen), роботоризирани ръце и ръкохватки, табло включващо микрофони, достъп до компютър и център за управление *HERMES* и др.

Zeus Console

Контролната конзола на хирурга

4.1.1. **Общи характеристики на системата**

- Индустриален стандартен механизъм за стерилизация - включва механизъм за почистване, направен по стандартите на ендоскопичното оборудване.
- Годност за неколкократна употреба - използва здрави преизползваеми инструменти изработени за да отстояват на корозия и счупване
- Размер на отвора - предлага уникална прецизност от 3,5 до 5 мм иструменти и ендоскопични приспособления.
- Широк набор от инструменти - прилага широка гама от над 40 вида *ZEUS* съвместими инструменти.
- Бърза смяна на инструментите - включва механизъм за бърза подмяна на инструментите.
- Бързо стартиране на системата - отнема не повече от 15 минути
- Визуализация - конструиран за лесна и бърза адаптация към индивидуалните особености на хирурга при визуализацията, позволява дву и триизмерна графика и включва широк диапазон от ендоскопични и

  мониторни настройки.
- Допълнителен монитор - той е вграден като допълнение към главния монитор и служи за предаване на допълнителна информация от пациента. Включва подсистемата *SOCRATES*™.
- Профил - изградена от изключително леки материали за по-лесно инсталиране и подмяна на частите съобразно предстоящата процедура.
- Задвижването на системата става интуитивно чрез педал за управление - при активиране на педала се включва системата за звуково управление и монитора за допир (Touch screen)
- Подсистема за ръчено управление „Microwrist" - служи за препредаване на движението на ръцете на хирурга към роботизираните ръце
- 6 степени на свобода
  - 4 моторизирани
    - нагоре и надолу
    - навътре и навън
    - движения от рамото - напред и назад

- движение от лакъта - напред и назад

- 2 плавнии движения
  - частта от лакъта до китката - назад и напред
  - китка
- Една фиксирана промяна на ъгъла на работа
  - лакътът може да се движи на +/- 3 градуса

Роботизирани ръце на системата (ZEUS Arms)

- Мащабиране - предлага големи възможности за управление върху мащабирането без лимит към произволно избрани нараствания. Мащабирането може да бъде контолирано чрез звукови команди или чрез Touch screen.

- Комфорт - ергономичната конзола и мястото на хирурга, предлагат максимален комфорт по време на процедурите.
- По време на операцията роботизираните ръце се позиционират спрямо операционната маса. Специалният дизайн елиминира нуждата от допълнителни настройки и калибриране на роботизираните ръце по време на операцията.
- Запазване на позицията - предоставя средство за запазване на три различни едноскопични позиции, чрез x, y, z координатите им към които можем да се върнем по всяко време.
- Звуков контрол - компонентите за звуковия контрол са съставени от комплексна комуникационна програма:

- Индивидуално разпознаване на гласа на хирурга.
- Разпознаване на подадените команди и йерархична система за степенуване им по важност.
- Ограничен речник с цел избягване на „фалшиви команди”, звуков и визуален „отговор” при успешно изпълнение на командите.
- Компенсаторен механизъм за избягване на звукови смущения.

- Огледално повторение - използва технологията за „огледано повторение” в такт до 1000 пъти в секунда, за по-голяма сигурност на пациента

4.2. Описание на da Vinci

Intuitive Surgical е водеща компания в областта на дигиталната хирургия с тяхната *da Vinci Surgical System*. През 2000 г. е представена напреднала хирургическа техника за рязане и зашиване. Тази система е първата оперативна хирургическа роботизирана система, чиято подвижност е приемущество над конкуренцията. Повече от 210 уреда се използват в САЩ, Европа и Япония. Въпреки многото пречки, сега е мултимилионен бизнес, който продължава да расте.

Има четири главни компонента в системата: хирургична конзола, операционна маса, *EndoWrist* инструмент, *Insite Vision System* с висока резолюция и 3D ендоскоп и *Image Processing Equipment*

- Хирургична конзола - Разположена е на няколко крачки от операционанта маса с пациента. Хирургът седи и гледа през увеличената три измерна картина на хирургическото поле в реално време, движенията на инструментите с който оперира. Управлението над тях е възможен за движения на 30 см от работното място.
- Област на пациента (*Patient-side Cart)* -Този компонент от системата съдържа роботизирани ръце, които са в директен контакт с пациента. Състои се от две или три роботизирани ръце и една ендоскопична ръка. Обратната връзка днес е ограничена, така че хирургът се нуждае наистина от единство на визуалната.

В края на осемдесетте години, мотивирани от *MIS* и нарастващия набор от хирургически инструменти, изследователи от *SRI International* започват да търсят

начини за разширяване на хирургическите умения в *MIS* и микрохирургията. Финансирани от Националния здравен институт през деведесетте години, в *SRI* се разработва успешен прототип, който скоро става известен като *SRI system*. Той комбинира предимствата на отдалечената манипулация със стереоскопично визуализиране, мултимодална сензорна реагираща система и ергономичен дизайн. Това позволи разширяването на набора от операции чрез *MIS*, както и отдалечена хирургическа намеса.

Ранният успех на тази система привлича вниманието на "Администрацията на Института за отбрана и напреднали разработки" *DARPA, Arlington*, същата агенция която стои зад разработката на Интернет в първоначалният му вариант.

DAPRA вижда зад телехирургията метод да бъдат оперирани воиниците на бойното поле и да ги поддържа живи до евакуацията им в „спешните отделения" на болниците. Използвайки телехирургия чрез сателитна връзка, армейските хирурзи могат да оперират ранените войници отдалечени на хиляди километри и същевременно оставайки далече от боиното поле.

Intuitive sergical е основана през 1995 г. с цел създаването на цивилни медицински приложения и въвеждане телероботиката в *MIS*. Използвайки технологии на *SRI*, *IBM*, и *Massachusetts Institute of Technology*, е разработена ръка-робот способна на движения, извършвани под голям градус, изискван за комплексна реконструкивна хирургия през сантиметров отвор. В същото време се появява апарат за триизмерена картина и звук, показвайки безпрецедентна визуализация на човешката вътрешност. Създаден е уред "отворен" към хирурга, който му позволява да използва пълия си набор от умения в натурална и интуитивна среда. Прогресът е ускорен от експлодиращата мощ на компютърните процесори. Настоящата версия на *Intuitive Systems* използва четири процесорна система

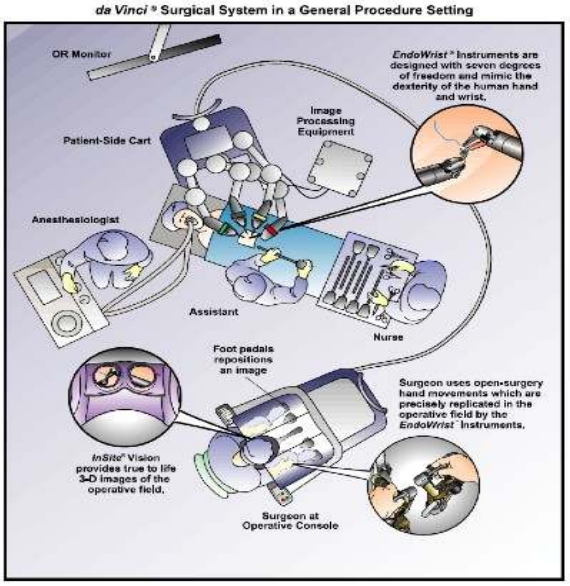
способна да достави 250 мегафлопа компютърна мощ.
След повече от десетилетие от началото на телехирургията, кулминацията се достиг при първите телехирургични операции извършени върху човек в Белгия през 2005 г.

Роботизираните хирургически системи са високо мобилни и използват 4 градусов ъгъл на свобода (3 за ориентация и 1 за захващане). *EndoWrist* и 3 ”градусов ъгъл на свобода” робот за осигуряването на широк диапазон на движение, осигуряваики на хирурга ориентация и достъп до оперираното място. Прецизни механични движения осигуряват достъп до екстремно малки структури в човешкото тяло. Вече се оперират структури с размер 0.05 до 0.002 инча.

3D визуализираща система с 800 линиина резолюция и 62 децибела “signal-to-noise” съотношение доставят на хирурга брилянтна картина на работа му.
Оптичната система ограничава също геометричните смущения в прожектираната картина, позволяваики стерео картина дори до ъгълчетата на екрана. За избягване на цветни смущения, системата използва високо акуратно рендиране на цветовете. Използването на 250 мегафлопа компютърна мощ прави възможно настройки в скоростa на реагиране, намаляване на трептенето, комуникацията между “господар-подчинен”, плавно управление както и компенсиране на гравитацията. Безупречната работа на станцията се осигурява от няколко степенно ниво на защита.

Телероботиката също така предоставя на хирурга така наречения метод “динамична задача” (dynamic assignment). По време на телехирургия, хирургът може да постави едната роботизирана ръка в статично положение и да се концентрира изцяло върху втората. Алтернативно двете ръце могат да бъдат поети от двама хирурзи.

Интуитивните системи и други като нея са само началото на широка вълна от хирургически апликации за телероботиката. Най-новата система *da Vinci* представлява четирирък хирургически робот с цена около 1,5 милиона долара. От началото до сега системата е спечелила много конкурси и е въведена в значителен брой болници по целия свят.[16]



4.2.1. **Общи характеристики на системата**

- Триизмерна визуализираща система (3D HD):
- Панорамна картина в формат 16 / 9
- Дигитално увеличение намаляващо смущенията между ендоскопът и инструментите
- 0 и 30 градусов стерео ендоскоп
- Прецизно управление дори чрез върха на пръстите с *EndoWrist* технологията.
- Контролиране степента на движението на робота спрямо движението на ръцете на хирурга.

- Широк диапазон на движение на роботизираните ръце и удължени инструменти позволяващи по добра достъпност.
- Тънки, телескопични инструменти
- Широка гама от 5 и 8 милиметрови *EndoWrist* инструменти.
- Изключителен комфорт за хирурга по време на операция.
- Механизирано легло на пациента.
- Интегрирана четвърта ръка за бързо разгръщане.
- Touch screen
- Високоскоростна вибро-оптична връзка между системите
- Стерилни адаптори за еднократна употреба с монтирани извивки.
- Възможност за отдалечена оперативна намеса
- Минимален разрез по време на операция, което води до бързо следоперационно възстановяване и минимална болка за пациената.
- Възможност за едновременна работа на двама или повече хирурзи върху един пациент

4.3. Сравнителен анализ

Функционалност и характеристики	ZEUS	da Vinci
Система за визуализация - Двумерна *(2D)* - Триизмерна *(3D)*	✓ ✓	 ✓
Системи за управление - Мултихирургическа намеса - Touch screen	 ✓	✓ ✓

- Звукова навигация - Отдалечена оперативна намеса - Мащабиране - Подсистема за ръчено управление - Запазване на позицията	✓ ✓ ✓ ✓	 ✓ ✓ ✓
Операционна система - Роботизирани ръце - Степени на свобода	3 6	4 7
Ергономичност	✓	✓
Цена на системата	$975,000	$1 милион

4.4. Заключение

Въпреки сходните задачи на двете системи (*ZEUS*, *da Vinci*), фактите съпътстващи тяхното създаване обуславят и голяма част от техните различия.

Системата *da Vinci* е създадена чрез прякото финансиране на военните и нейната цел е основно телехирургичната намеса. Това е и основното предимство на *da Vinci*: оперативните намеси през огромни разстояния и възможноста няколко хирурга да оперират едновременно един пациент. Бидейки един от пионерите в областа на телехирургията (пряк наследник на *SRI цялото име*), системата се прехвърля в цивилната област и използвайки технологичния напредък придобит през годините успява да се утвърди като най-комплексната и развита до момента технология в областа на отдалечените хирургични намеси.

ZEUS от друга страна е създаден с чисто комерсиална насоченост и това обуславя предимствата и предимно в областа на управление на робота (звуково управление и др.) и ергономичността на опериращия хирург.

5. Тест

Възможни са повече от 1 верни отговори на въпрос.

1. **Кой за първи път използва думата робот:**
   a) Карел Чапек
   b) Агата Криси
   c) Чарлс Дарвин

2. **С каква цел са замислени телероботите:**
   a) по висока ефективност
   b) по ниски разходи

c) развлекателна цел
d) работа от разстояние

3. Кой от изброените е език за управление на робот:

a) VRML
b) XRCL
c) MatLab

4. Какво значи абривиатурата MIS:

a) Minimally Invasive Surgery
b) Management Information System
c) Master of Information Science

5. Защо XML се използва за основа на езиците за телероботи:

a) лесно надграждане
b) проста структура
c) широка поддръжка

6. Кой от езиците се използва за визуализация:

a) Action Script
b) SVG
c) Adobe Photoshop

7. Кое от изброените не е част на телеробота:

a) камери
b) сензори
c) изкуствен интелект

8. Колко роботизирани ръце има системата ZEUS:

a) 2
b) 3
c) 4

9. Кой от езиците е "базиран" на размита логика:

a) XRCL
b) BioML
c) XML

Заключение

На база на изготвената дипломна работа, дипломантът счита, че е изпълнил поставените цели, а именно:

- Запознаване с робтите, видовете и приложенията им
- Описание на тяхна функционалнист и начин на поведение
- Описание на системите за управление и за визуализация, както и езиците за тяхното управление
- Приложението на телероботите в медицината
- Сравнителен анализ на двете най-разпространени телехирургини системи

Подобряване на дипломната работа с цел бъдещо развитие:

- Отделяне по голямо внимание на ИИ и приложението му в телероботиката, с цел повишване на ефективността и минимизиране участие на човека в процеса на управление

Речник

- **AI** (Artificial intelligence)
  Изкуствен интелект
- **CSS** (Cascading Style Sheets)
- **DOM** (Document Object Model)
- **GIS** (Goegraphic Information System)
  Географска информационна система

- **GUI Interface** (Graphical user interface)
  Графичен интерфейс
- **HRI** (Human Robot Interface)
  Интерфейс човек-робот
- **IRL** (Industrial Robot language)
  Език за програмиране от високо ниво за индустрялни роботи.
- **MIS** (Minimally Invasive Surgery)
  Минимално проникваща хирургия
- **RoboML** (Robotic Markup Language)
- **RobotScript** - Robotic Workspace Technologies (RWT)
- **SVG** (Scalable Vector Graphics)
  Графичен формат за векторни графики
- **XHTML** (Extensible HyperText Markup Language)
- **XRCL** (The Extensible Robot Control Language)
- **XSL** (EXtensible Stylesheet Language)

Използвана литература

[1] **Айзък Азимов**, *Трите закона на роботиката;* http://bg.wikipedia.org/

[2] *Industrial Robotics And Telerobotics In Manufacturing And Service Industries, March 1996;* http://www.wtec.org/loyola/hci/c6_s3.htm

[3] Extensible Markup Language*;* http://www.w3.org, http://www.w3.org/XML/, http://www.xml.com/axml/testaxml.htm

[4] **Maxim Makatchev, S.K.To, 2000,** *Human-Robot Interface Using Agents Communicating in an XML Based Markup Language*

[5] **T. Arai, T. Itoko, H. Yago**, *A Graphical Robot Language Developed in Japan, Robotica,* vol. 15, pp. 99-103, 1997

[6] **T. Finin, Y. Labrou, J. Mayfield,** *KQML as an Agent Communication Language, in Software Agents*, ed. J. M. Bradshaw, AAAI Press/The MIT Press, pp. 291-316, 1997

[7] **Sami Salmi, 2000,** *Potential Use of XML in Robotics*

[8] **Douglas S. Blank, Jared H.Hudson, Braian C. Mashburn and Eric A. Roberts**, *The XRCL Project: The University of Askansas' Entry into the AAAI 1999 Mobile RObot Competition*

[9] Scalable Vector Graphics;
http://w3.org/Graphics/SVG/; http://www.w3schools.com/svg/svg_grad_linear.asp

[10] The Virtual Reality Modeling Language;
http://www.web3d.org/x3d/specifications/vrml/vrml97/index.htm

[11] Interactive Robot Manipulation using VRML 2.0;
http://www.geocities.com/ResearchTriangle/Lab/8585/robot/robot.html

[12] **By J. Kenneth Salisbury, Jr.,** *The heart of microsurgery* **;**
http://www.memagazine.org/backissues/december98/features/microheart/microheart.html

[13] Peter Kazanzides, Development of the ROBODOC System for Image-Directed Surgery; http://www.cs.jhu.edu/~cis/cista/445/Lectures/Robodoc903.pdf

[14] Robotics: the Future of Minimally Invasive Heart Surgery;
http://biomed.brown.edu/Courses/BI108/BI108_2000_Groups/Heart_Surgery/Robotics.html

[15] The Zeus System;
http://library.thinkquest.org/03oct/00760/Zeus%20System.htm

[16] da Vinci Surgical System;
http://www.intuitivesurgical.com/index.aspx