

Амбиент-ориентирано моделиране на контекстно- чувствителни системи

Публична академична лекция

доц. д-р Тодорка Глушкова

Пловдивски университет „Паисий Хилендарски“

Факултет по математика и информатика

Въведение

- През последните години стават все по-актуално създаването на контекстно-зависимите системи, които се самонастройват и адаптират според промените на околната среда.
- Улавянето на контекста става с помощта на сензори, които могат да са физически, виртуални или логически, което предопределя тези среди като IoT екосистеми.

Контекст

- Съществуват различни дефиниции на понятието „контекст“.
 - място, идентичности от хора и заобикалящите ги обекти;
 - идентичност, местоположение, заобикаляща среда и времето на потребителя;
 - знанията на компютъра за елементите на околната среда;
 - абстрактно ниво на текущото състояние, което се констатира от физическите и логически сензори и пр.

Контекст. Дефиниция

Ще премем дефиницията на Deu, според който *контекст е всяка информация, която може да се използва за категоризиране на състоянието на една идентичност (entity)- човек, място или обект, които се смятат за свързани с взаимодействието между потребител и приложение, включително и самите потребител и приложение.*

IoT системи

- IoT системите включват разнотипни хардуерни конфигурации (от мощни сървъри до елементарни сензори) и комуникационни протоколи.
- Съществено за тези системи е отчитане на заобикалящия физически свят – състоянието му и промените в него, случващи се събития, време и местоположение.
- Една софтуерна IoT архитектура включва различни видове активни и пасивни компоненти – основно агенти, но също услуги, модули, онтологии.
- Удачно е използването на интелигентни, автономни, контекстно-информирани или контекстно-зависими компоненти.

Моделиране

- Поради хардуерната и софтуерната сложност и хетерогенност, изграждането на IoT контекстно-зависими приложения е свързано със сериозни рискове.
- Нецелесъобразно, неефективно и финансово неизгодно е директна разработка на такъв тип приложения.
- Предшестващото моделиране би било от огромна помощ, като моделът на една IoT система може да се изследва и анализира предварително за идентифициране на слаби места и грешки.

Амбиент- ориентирано моделиране

- Изборът на подходящ подход за моделиране е важен проблем.
- Един подходящ и адекватен подход за моделиране на IoT системи е амбиент-ориентираното моделиране (ambient-oriented modeling, съкратено AOM).
- Този тип моделиране се основава на понятието **амбиент**.

Амбиенти, амбиентна интелигентност и IoT

Един амбиент е идентичност, която притежава следните характеристики:

- **Ограниченост** – ограничена локация, където се случва интересуващо ни изчисление.
- **Вложеност** – един амбиент може да бъде вложен в други амбиенти. Могат да се създават йерархии от амбиенти.
- **Мобилност** – един амбиент може да бъде преместван от едно местоположение в друго като едно цяло.

Амбиенти

За целите на моделирането един амбиент може да бъде представен като структура със следните елементи:

- **Идентификатор**– задължителен елемент, който освен за идентификация служи и за контрол на достъпа.
- **Кореспондиращо множество от локални изчислителни процеси** – те работят директно в амбинета и в известен смисъл го контролират.

Ambient Intelligence-AmI

- Основната идея зад AmI е, че опериращите в средата приложения, използвайки събраната информация в реално време и натрупаните във времето фонові данни, могат да вземат решения в полза на потребителите в нея.
- Типични примери са персоналните помощници, които, в зависимост от ситуацията, са способни да оказват проактивна помощ или да упражняват съдържаност.

Основни характеристики на AmI

- **Чувствителност** – отчитат обекти и субекти в околната среда и реагират на поведението им.
- **Адаптивност** – коригират поведението си в зависимост от конкретната ситуация. В идеалния случай предвиждат събития.
- **Прозрачност** – от потребителите не се очаква да са наясно с използваните технологии.
- **Универсалност** – една от идеите на AmI е да присъства на възможно най-много места, като наличната информация да позволява на системата да реагира по-адекватно на възникващи събития.
- **Интелигентност** – използват се методи от изкуствения интелект.

AmI и IoT

- Конвергенцията на AmI и IoT цели изграждане на среди, в които интелигентни обекти, свързани към Интернет, си взаимодействат и предоставят комплексни услуги на потребителите.
- Според визията на AmI IoT - устройствата са широко разпространени в околната среда под формата на сензори, задвижващи устройства, интелигентни уреди, активни обекти или мобилни роботи.
- Тези устройства са взаимосвързани помежду си и могат да бъдат винаги и навсякъде достъпни посредством малки преносими мобилни устройства.

AmI и IoT

- В сравнение с класическите уеб- услуги, услугите за AmI и IoT доставят не само изчислителни и софтуерни ресурси, но и различни възможности за контрол на физически устройства и събития.
- Конвергенцията може да бъде особено ефективна за хора в неравностойно положение, възрастни хора, деца или пациенти, на които повсеместната и околна среда може да предостави множество реактивни или проактивни услуги за подобряване на качеството на живот.

Формализми за амбиент-ориентирано моделиране

- Основополагащ формализъм, за АОМ е π -**калкюла**, който представя вид процесно смятане. Основно понятие е **канал**, който може да се "движи" по други канали.
- **Join-калкюльът** е преформулиране на π -калкюла с по-ясна представа за локациите на взаимодействие
- **Ambient Calculus (AC)** - възприета е парадигмата за мобилност. В AC амбиентите са йерархично структурирани, агентите са ограничени до амбиенти и амбиентите се движат под контрола на агенти.
- Наследник на AC е **Calculus of Boxed Ambients(CBA)**

ССА моделиране

- ***Calculus of Context-aware Ambients (CCA)***, надгражда CBA с нови конструкции и се осигурява възможност на мобилните амбиенти да реагират в зависимост от промените на околната среда.
- За представяне на свойствата на CCA- процесите се използва логически език, в който основна роля играе контекстния израз.
- Контекстните изрази се използват в CCA, за да се гарантира изпълнението на определена възможност само при наличието на определени условия на средата, т.е. в определен контекст.

Амбиентът в ССА

- ССА се базира на нотацията на амбиентите.
- Понятието "ambient" означава околна, обкръжаваща. Това е идентичност, която се използва за описание на обект или компонент – личност, процес, устройство, местоположение и пр.
- Амбиентът може да бъде мобилен и да комуникира с други амбиенти.
- Амбиентът има име и може да съдържа други амбиенти в себе си, като позволява набор от амбиенти да се представи йерархично.
- Съществуват 3 възможни взаимоотношения между два амбиента: родител (parent), дете (child) и роднина (sibling).

Комуникация между амбиенти

- Всеки амбиент може да комуникира с амбиентите около себе си.
- Амбиентите могат да обменят (изпращат и получават) съобщения помежду си.
- Процесът по обмяна на съобщения се извършва с използването на процеса ръкостискане (handshaking).
- В ССА-нотацията „::“ е символ за роднини амбиенти; „↑“ и „↓“ са символи за родител и дете; „<>“ означава изпращане, а „()“ – получаване на съобщение.

ССА синтактични категории

- В ССА могат да се разграничат 4 синтактични категории:
 - **процеси** (processes) P ,
 - **възможности** (capabilities) M ,
 - **местоположения** (locations) a ,
 - **контекстни изрази** (context expressions) k .

ССА-Процеси

- Процесът 0 не извършва нищо и приключва незабавно.
- Процесът $P|Q$ индикира, че процесът P се извършва паралелно с процеса Q .
- Знакът за удвояване е $!$, така процесът $!P$ означава процес, който може да създаде ново копие на процеса P , т.е $!P = P|!P$.
- Процесът $n[P]$ обозначава амбиент с име n , а P е процесът, който описва поведението на амбиента.
- Двойката скоби " $[$ " и $]$ " предопределят границите на амбиента.

ССА - възможности

- Възможностите за движение позволяват на един амбиент да се движи в заобикаляща го среда навътре (in) и навън (out).
- Възможността за извикване на процес $\alpha < \tilde{y} >$, свързан с x на място α , се реализира, като всеки фактически параметър от списъка параметри $\tilde{y}$ се замести със съответстващия му формален параметър.
- Амбиентите могат да си разменят съобщения и чрез възможността $\alpha < \tilde{y} >$ един амбиент може да изпрати списък с имена $\tilde{y}$ на локация α , а чрез възможността $\alpha(\tilde{y})$ друг амбиент ще получи списък с променливи $\tilde{y}$ на локация α .

ССА-Местоположение

- Местоположението а може да бъде:
 - " $\uparrow$ ", което означава някой родител амбиент;
 - " $n \uparrow$ " се използва за конкретен родител с име n ;
 - " $\downarrow$ " се отнася за дете амбиент, а
 - " $n \downarrow$ " обозначава конкретно дете амбиент с име n ;
 - знакът " $::$ " се използва, за да покаже роднина амбиент, а
 - " $n::$ " индикира конкретен роднина амбиент с име n .
- Всеки амбиент може да използва и празен стринг ϵ , за да се обърне към себе си.

ССА-контекстни изрази

- Контекстните изрази определят поведението на системата.
- Контекстният израз $\bullet$ е изпълнен само за празен контекст.
- Контекстният израз $n=m$ е приложим само, ако имената n и m са лексикално идентични.
- Логическите оператори от първи ред разширяват стандартното си значение при контекстните изрази.
- Контекстният израз $k_1|k_2$ е валиден за съчетание на два контекста.
- Контекстния израз $n[k]$ е валиден за контекст с име n , такъв че k е изпълнен, когато е в самия амбиент.

ССА-контекстни изрази

- Контекстният израз **new(n,k)** е валиден за контекст, който е ограничение на името n на друг контекст, за който k е изпълнен.
- Контекстният израз $\oplus k$ е валиден за контекст, който има дете контекст, за което k е изпълнен.
- Контекстният израз $\langle \rangle k$ е валиден за контекст, ако в него съществува подконтекст, за който k е изпълнен.

ССА-контекстни изрази

Унарни предикати:

- ✓ $\text{has}(n) = \oplus(\bullet | n[\text{True}] | \text{True})$, амбиент λ съдържа амбиент с име n ;
- ✓ $\text{at}(n) = n[\oplus(\bullet | \text{True})] | \text{True}$, амбиент λ се намира в амбиент с име n .
- ✓ $\text{with}(n) = n[\text{True}] | \oplus(\bullet | \text{True})$, λ и n имат общо местоположение

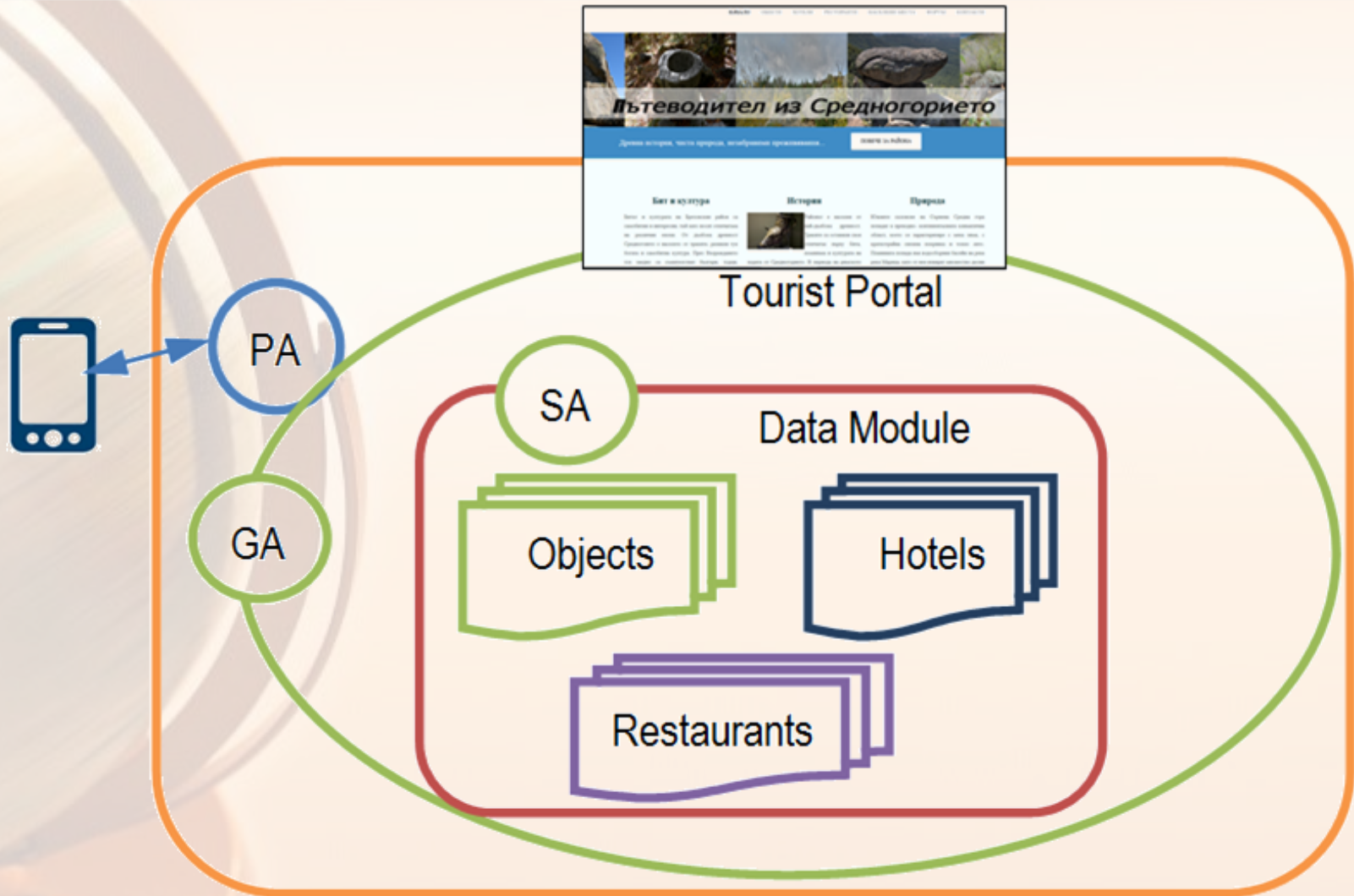
Могат да се дефинират и бинарни предикати като:

- ✓ $\text{at2}(n, m) = n[m[\text{True}] | \text{True}] | \text{True}$, ако m е разположен в n .
- ✓ $\text{with2}(n, m) = n[\text{True}] | m[\text{True}] | \text{True}$, когато амбиент m е с амбиент n .

Пример 1: Туристически пътеводител

- Да разгледаме моделирането и прототипирането на система, която ще отчита:
 - **времевите характеристики,**
 - **промяната в местоположението на потребителя,**
 - **надморската височина,**
 - **температурата,**
 - времето като част от денонощието и др.
- Ще използваме сензорите на мобилното устройство на туриста за получаването на актуалната информация за текущия контекст.

Архитектура на системата



Видове асистенти

- **Персонални асистенти (РА)**-подпомагат работата на потребителите в средата
- **Асистенти специалисти (SA)** - осигуряват специализирани услуги на потребителите при изпълнението на техните заявки.
- **Гардове (GA)**-имат специфични функционалности, свързана със сигурността по време на изпълнението на услугите. В модела те ще следят промяната на температурата, надморската височина, вероятността от буря и при възникване на определени извънредни ситуации, ще предупреждават туристите, а ако се налага и спасителните служби.

Услуга „Туристически пътеводител“

- Според местоположението на туриста се реализира търсене на най-близкия туристически обект и информация за него в графичен, текстов или аудио формат.
- При движението на туриста се осигурява информация за маршрута, местата за отдих, забележителни точки по маршрута или такива, които могат да се видят от разстояние.
- При получаване на информация за рязка промяна в температурата и вероятността за буря, ГА активира търсене на най-близки хотели, заслони или места за отдих и предлага навигация на туриста до тях.
- В случай на извънредна ситуация (ниска температура, буря, висока надморска височина и преустановяване на движението на туриста) ГА може да сигнализира спасителните служби.

ССА нотация на разработвания модел.

CCA constants		CCA variables		
Notation	Description	Notation	Description	Values
TP	Tourist Portal	PA _i	Personal Assistant	PA ₁
DM	Data Module	dtype	Device type	PDA, GSM...
MD	Mobile Device	aname	Ambient name	PDA201,...
SLIST	Services List	<u>obj_id</u>	Object ID	Obj001,...
OR	Objects Repository	replay	Replay	<u>OK,NO,content</u>
HR	Hotels Repository	content	Info	Content
Null	NULL	<u>ack</u>	Acknowledgment	ACK

ССА моделиране

$$P_{SA_Obj} \triangleq \left(\begin{array}{l} (! :: (PA_i, location, time, altitude).TP \uparrow \langle PA_i, location, time, altitude \rangle . 0) | \\ (!TP \uparrow (reply, PA_i).PA_i :: \langle reply \rangle) \end{array} \right)$$

$$P_{SA_hot} \triangleq \left(\begin{array}{l} (! :: (PA_i, location).TP \uparrow \langle PA_i, location \rangle . 0) | \\ (!TP \uparrow (list1, PA_i).PA_i :: \langle list1 \rangle) \end{array} \right)$$

$$P_{SA_rest} \triangleq \left(\begin{array}{l} (! :: (PA_i, location).TP \uparrow \langle PA_i, location \rangle . 0) | \\ (!TP \uparrow (list2, PA_i).PA_i :: \langle list2 \rangle) \end{array} \right)$$

$$P_{DM} \triangleq \left(\begin{array}{l} (!TP :: (location, PA_i).(HR \downarrow \langle location, PA_i \rangle . 0 | RR \downarrow \langle location, PA_i \rangle) | \\ (!HR \downarrow (list1, PA_i) | (!RR \downarrow (list2, PA_i)).TR :: \langle list, PA_i \rangle . 0) \end{array} \right)$$

$$P_{TP} \triangleq \left(\begin{array}{l} !SA_Obj \downarrow (location, altitude, time, PA_i). \\ GA :: \langle location, altitude, time, temperature, procent, PA_i \rangle . 0 | \\ !GA :: (location, PA_i).DM :: \langle location, PA_i \rangle . 0 | \\ !DM :: (list, PA_i).PA_i \downarrow \langle list \rangle . 0 \end{array} \right)$$

$$P_{GA} \triangleq \left(\begin{array}{l} !TP :: (location, altitude, time, temperature, procent, PA_i). \\ ((time > 19) or ((temperature < 2) and (procent > 50) and (altitude > 800))) \\ ?TP :: \langle list, PA_i \rangle . 0 \end{array} \right)$$

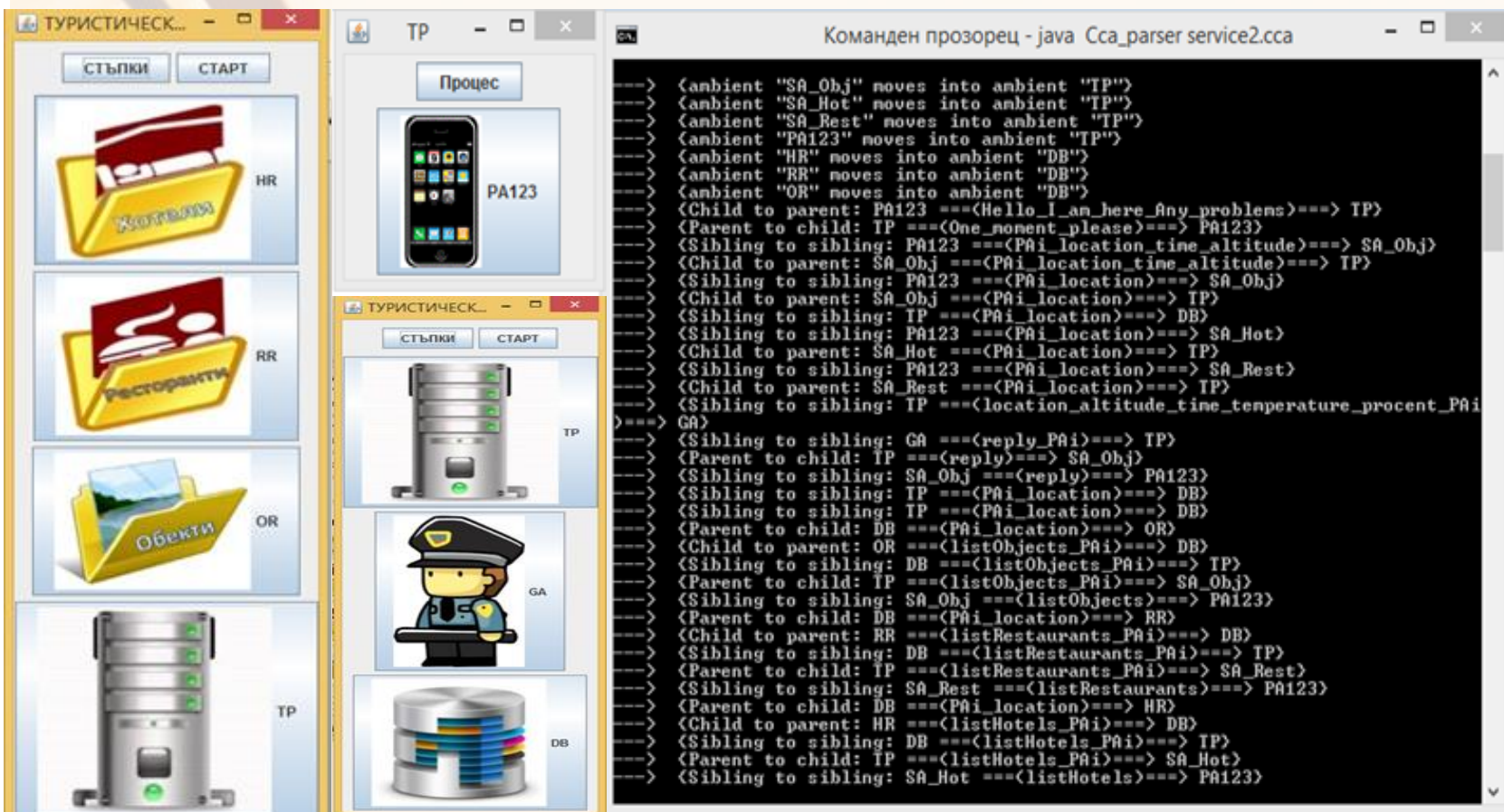
ССА симулатор

- Трудно е да се възпроизведе синтаксисът на ССА, поради специфичните символи като например „↑“ и „↓“.
- Езикът сsaPL е четима за компютъра версия на синтаксиса на ССА.
- Въз основа на базовата версия на интерпретатора на сsaPL като java- приложение, разработихме симулатор на представения модел.
- Нотацията „ $A \Rightarrow (X) \Rightarrow B$ “ означава, че амбиент „A“ изпраща съобщение „X“ до амбиент „B“. Нотациите „Child to parent“, „Parent to child“, и „Sibling to sibling“ дават информация за взаимовръзката между *амбиентите*.

ссаPL програмна реализация на сценария

```
SA_Obj[in TP.0]|SA_Hot[in TP.0]|SA_Rest[in TP.0]|PA123[in TP.0]|
TP[PA123[TP@send>Hello_I_am_here_Any_problems).TP@recv(x1) .0]|
PA123#recv(y1).<y1=Hello_I_am_here_Any_problems>PA123#send(One_moment_please).0|
SA_Obj[PA123 :: recv(PAi_location_time_altitude). TP @ send(PAi_location_time_altitude). 0|
TP@recv(reply).PA123::send(reply).0|PA123::recv(PAi_location).TP@send(PAi_location).0|
TP@recv(listObjects_PAi).PA123::send(listObjects).0]|SA_Hot[PA123::recv(PAi_location).TP@se
nd(PAi_location).0|TP@recv(listHotels_PAi).PA123::send(listHotels).0]|
SA_Rest[PA123::recv(PAi_location).TP@send(PAi_location).0|TP@recv(listRestaurants_PAi).PA1
23::send(listRestaurants).0]|PA123[SA_Obj::send(PAi_location_time_altitude).SA_Obj::recv(reply
).0|SA_Obj::send(PAi_location).SA_Obj::recv(listObjects).0|SA_Hot::send(PAi_location).SA_Hot::
recv(listHotels).0|SA_Rest::send(PAi_location).SA_Rest::recv(listRestaurants).0]|
SA_Obj#recv(PAi_location_time_altitude).GA::send(location_altitude_time_temperature_procent_
PAi).0|GA::recv(reply_PAi).SA_Obj#send(reply).0|SA_Obj#recv(PAi_location).DB::send(PAi_locat
ion).0|DB::recv(y10).<y10=listObjects_PAi>SA_Obj#send(listObjects_PAi).0|
SA_Rest#recv(z3).<z3=PAi_location>DB::send(z3).0|DB::recv(z4).<z4=listRestaurants_PAi>SA_
Rest#send(listRestaurants_PAi).0|SA_Hot#recv(x11).<x11=PAi_location>DB::send(x11).0|
DB::recv(x12).<x12=listHotels_PAi>SA_Hot#send(listHotels_PAi).0 ]|
GA[!::recv(x7).<x7=location_altitude_time_temperature_procent_PAi>TP::send(reply_PAi).0]|
DB[OR[!@recv(PAi_location).DB@send(listObjects_PAi).0]|TP::recv(y6).<y6=PAi_location>OR
#send(y6).OR#recv(y7).<y7=listObjects_PAi>TP::send(listObjects_PAi).0|RR[!@recv(PAi_locatio
n).DB@send(listRestaurants_PAi).0]|!::recv(y8).<y8=PAi_location>RR#send(y8).RR#recv(y9).<y9
=listRestaurants_PAi>TP::send(listRestaurants_PAi).0|HR[!@recv(PAi_location).DB@send(listHo
tels_PAi).0]
```

сcaPL симулятор на модела

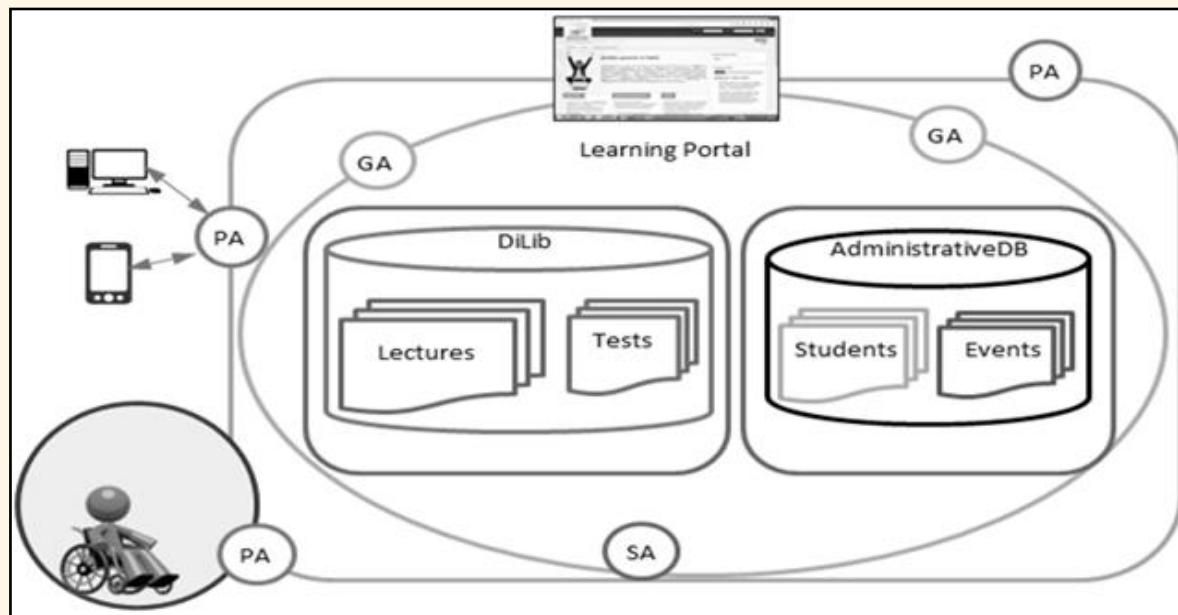


Пример 2: Инвалидна количка

- Студент с двигателни затруднения участва в учебния процес с интелигентната си инвалидната количка
- Количката разполага с комплект от сензори и комуникира с останалите обекти в образователната среда.
- След разпознаването на количката (и студента в нея), се активира РА, който доставя всички образователни услуги и учебни ресурси на неговото мобилно устройство.
- Щом студентът получи списъка с услуги sList, той разбира къде и кога ще се проведе съответната лекция или изпит.

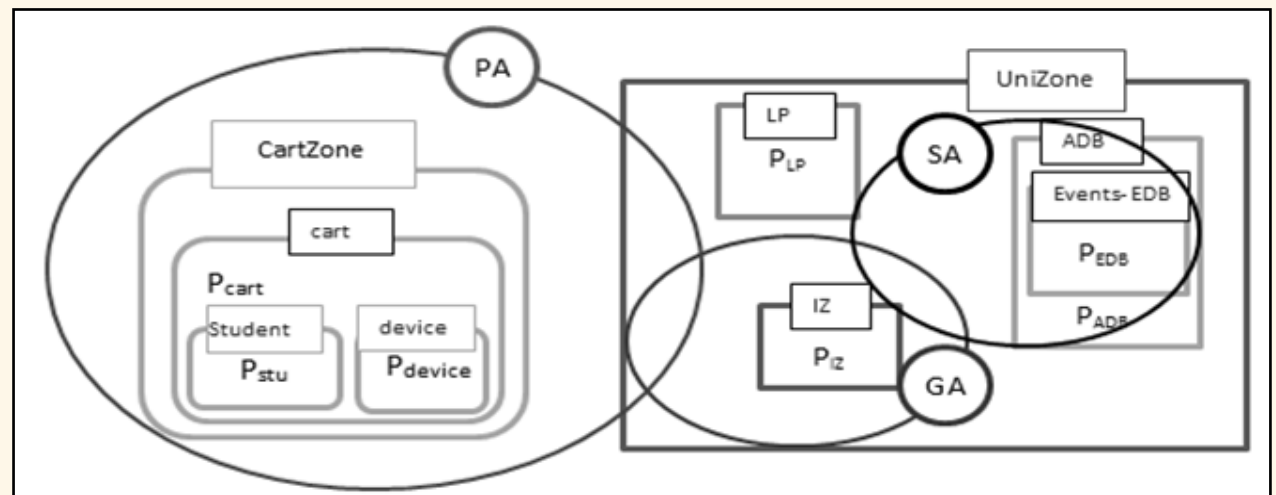
Примерен сценарий

- GA също получава тази информация и проверява работоспособността на всички важни зони, през които трябва да премине инвалидната количка и изпраща списъка на РА.
- От получената информация студентът избира маршрут и го изпраща на количката.
- В хода на придвижването, GA следи местоположението и при доближаване на някоя от зоните активира актуатори, свързани с отваряне на врати, спускане рампи, доставка на асансьор и пр.
- Ако в реално време някоя от важните зони стане неактивна, GA информира количката, която очаква от студента нов маршрут.
- Динамичното генериране на маршрути се реализира чрез модифициран A^* алгоритъм.



Архитектура на системата

Амбиентна структура



ССА- модел на сценария

$$P_{\text{cart}} \triangleq \left(\begin{array}{l} (\neg \rightarrow \text{at2}(\text{UniZone}, \text{cart})) ? \log \text{in} < \text{cart} > .0 | \\ \neg (\neg \rightarrow \neg \text{at2}(\text{UniZone}, \text{cart})) \\ ? \log \text{out} \diamond . (\neg \rightarrow \text{at2}(\text{UniZone}, \text{cart})) \\ ? \log \text{in} < \text{cart} > .0 \end{array} \right) | \left(\begin{array}{l} !GA :: (\text{ListActiveIZ}).\text{Device} :: < \text{ListActiveIZ} \\ !\text{Device} :: (\text{route}(\text{iz1}, \text{iz2}, \text{iz5}...)). \\ GA :: < \text{location}, \text{route} > .0 \end{array} \right)$$

$$P_{\text{SA}_E} \Leftrightarrow \left(\begin{array}{l} !PA :: (\text{stu_id}, \text{act_time}).\text{ADB} :: < \text{stu_id}, \text{act_time} > .0 | \\ !\text{ADB} :: (\text{stu_id}, \text{Event}(\text{event_id}, \text{event_type}, \text{attribute_list})). \\ PA :: < \text{Event}(\text{event_id}, \text{event_type}, \text{attribute_list}) > .0 \end{array} \right)$$

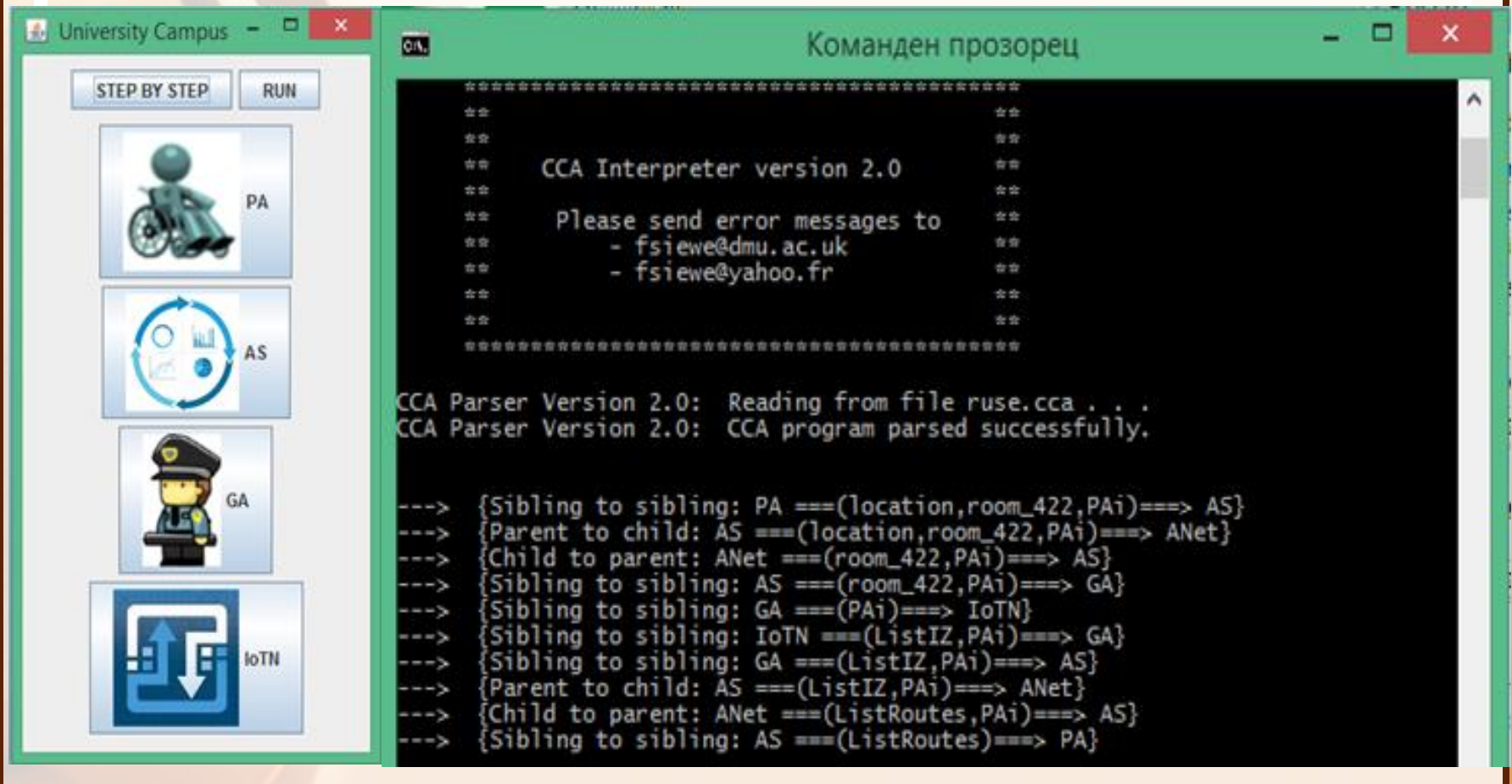
$$P_{\text{ADB}} \Leftrightarrow \left(\begin{array}{l} !SA_E :: (\text{stu_id}, \text{act_time}).\text{EDB} \downarrow < \text{stu_id}, \text{act_time} > .0 | \\ !\text{EDB} \downarrow (\text{stu_id}, \text{Event}(\text{event_id}, \text{event_type}, \text{attribute_list})). \\ SA :: < \text{stu_id}, \text{Event}(\text{event_id}, \text{event_type}, \text{attribute_list}) > .0 \end{array} \right)$$

$$P_{\text{GA}} \Leftrightarrow \left(\begin{array}{l} !PA :: (\text{act_location}, \text{event_location}).\text{IZ} :: < \text{act_location}, \text{event_location} > .0 | \\ !\text{IZ} :: (\text{act_location}, \text{event_location}, \text{ListActiveIZ}).PA :: < \text{ListActiveIZ} > \end{array} \right)$$

$$P_{\text{IZ}} \Leftrightarrow \left(\begin{array}{l} !GA :: (\text{act_location}, \text{event_location}). \\ GA :: < \text{act_location}, \text{event_location}, \text{ListActiveIZ} > .0 \end{array} \right)$$

$$P_{\text{PA}} \Leftrightarrow \left(\begin{array}{l} !SA_E :: < \text{stu_id}, \text{act_time} > .0 | \\ SA_E :: (\text{Event}(\text{event_id}, \text{event_type}, \text{attribute_event})). \\ GA :: < \text{act_location}, \text{event_location} > .0 | \\ !GA :: (\text{ListActiveIZ}).\text{cart} :: < \text{route} > .0 \end{array} \right)$$

сcaPL-симулатор на част от сценария



The screenshot displays the 'University Campus' simulation interface. On the left is a control panel with a 'STEP BY STEP' button and a 'RUN' button. Below these are four icons representing different entities: a person in a wheelchair labeled 'PA', a circular flow diagram labeled 'AS', a police officer labeled 'GA', and a network diagram labeled 'IoTN'. On the right is a 'Команден прозорец' (Command Window) with a black background and white text. The text in the command window shows the CCA Interpreter version 2.0, instructions to send error messages to fsiewe@dmu.ac.uk or fsiewe@yahoo.fr, and the CCA Parser Version 2.0 reading from file ruse.cca and parsing successfully. Below this, a series of commands and responses are shown, including sibling, parent, child, and parent-to-child interactions between PA, AS, ANet, GA, and IoTN.

University Campus - [X]

STEP BY STEP RUN

PA

AS

GA

IoTN

Команден прозорец

```
*****
**                                     **
** CCA Interpreter version 2.0        **
**                                     **
** Please send error messages to     **
**   - fsiewe@dmu.ac.uk              **
**   - fsiewe@yahoo.fr               **
**                                     **
*****

CCA Parser Version 2.0: Reading from file ruse.cca . . .
CCA Parser Version 2.0: CCA program parsed successfully.

---> {Sibling to sibling: PA ===(location,room_422,PAi)====> AS}
---> {Parent to child: AS ===(location,room_422,PAi)====> ANet}
---> {Child to parent: ANet ===(room_422,PAi)====> AS}
---> {Sibling to sibling: AS ===(room_422,PAi)====> GA}
---> {Sibling to sibling: GA ===(PAi)====> IoTN}
---> {Sibling to sibling: IoTN ===(ListIZ,PAi)====> GA}
---> {Sibling to sibling: GA ===(ListIZ,PAi)====> AS}
---> {Parent to child: AS ===(ListIZ,PAi)====> ANet}
---> {Child to parent: ANet ===(ListRoutes,PAi)====> AS}
---> {Sibling to sibling: AS ===(ListRoutes)====> PA}
```

Заклучение и бъдещи планове

- В момента се работи по създаването на специфично подпространство AmbiNet като част от архитектурата на контекстно-зависими IoT-системи.
- Структурата на AmbiNet ще дава възможност за автоматично генериране на маршрути, оптимизиране на маршрути, визуален симулатор на сценарии, сsaPL редактор и др.
- Приложение на АОМ при разработката на системи, свързани с интелигентно земеделие, умни градове, интелигентна медицина и т.н.

**Благодаря за
вниманието!**