**Universitatea Politehnica Bucureşti**

Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

Temă de casă

**UML Profile for Schedulability, Performance and Time**

**Studenți: Matei Ioan-Tiberiu**

**Petre Mihai-Cătălin**

**Grupa: 443A**

**Prof. coordonator:**

**Conf. Dr. Ing. Ștefan Stăncescu**

2016

Cuprins

[*1.* General time modeling 3](#_Toc441740713)

[1.1 Modelul timp 4](#_Toc441740714)

[1.2 Mecanisme de cronometrare 5](#_Toc441740715)

[*2.* Schedulability modeling 7](#_Toc441740716)

3. Performance modeling...............................................................................................................11

3.1. Conceptul Domain Viewpoint.............................................................................................11

3.2. Modelul de tip Domain........................................................................................................12

3.3. Contextul de performanta....................................................................................................14

3.4. Reprezentarea in UML........................................................................................................16

4. Bibliografie................................................................................................................................18

Notiuni generale

Un profil UML este un subset specializat al UML care:

* Este în concordanță cu specificațiile UML
* Poate fi constrâns să omită porțiuni ale UML sau constrânge modul de utilizare
* Poate include notații specializate sau particularizate
* Se aplică pe un domeniu de aplicații vast
* Extinde sau specializează UML folosind „mecanisme de extensie ușoară”: Stereotipuri, valori etichetate, constrângeri

# General time modeling

De-a lungul timpului, mai multe tehnici și concepte de modelare au apărut în comunitatea software real-time. Fiecare tehnică a favorizat propriul set divers de terminologii și notații. Una dintre intențiile acestei specificații este să ofere o arhitectură comună în cadrul UML care cuprinde în întregime diversitatea tehnicilor dar lasă destulă flexibilitate pentru specializări diferite. Atenția a fost acordată îndeosebi proprietăților referitoare la modelarea aspectelor de timp și referitoare la timp precum caracteristicile cheie de actualitate, performanță și planificare.

Modelul domeniului timp identifica un set de concepte referitoare la timp si semantica ce sunt suportate, direct sau indirect, de acest profil. Prin urmare, orice concepte sau semantica ce nu sunt acoperite de acest model de domeniu sunt considerate a fi in afara ariei de acoperire a acestui profil si trebuie modelate prin alte mijloace. Modelul este destul de general, dar o aplicatie data trebuie doar sa utilizeze subsetul de concepte si semantica necesare.

Modelul de domeniu timp este partitionat in urmatoarele grupuri de concepte separate dar inrudite:

* Concepte de modelare de timp si valori de timp
* Concepte pentru modelare de evenimente in timp
* Concepte pentru modelare a mecanismelor de timp (ceasuri, cronometre)
* Concepte pentru modelarea serviciilor de datare, precum cele gasite in sistemele de operare realtime

Conceptele sunt grupate intr-un set de pachete precum urmatorul:

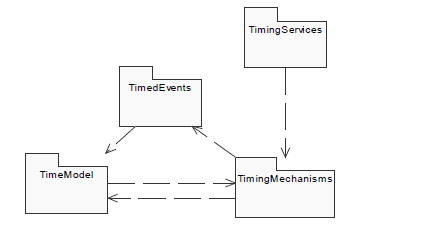


Figure 1 - Modulele modelarii in domeniul timp

Se poate observa dependenta reciproca intre pachetul de modelare in timp si pachetul de mecanisme de timp. Acest lucru este datorat imposibilitatii de a oferi exactitate asupra timpului fara a se folosi ceasuri.

Toate aceste pachete sunt bazate pe modelul de resurse general. In urmatoare sectiune, se va face o descriere a conceptelor din cadrul acestor pachete.

## Modelul timp

Modelul conceptual pentru reprezentare a timpului si valorilor de timp care este suportat de acest pachet este aratat in diagrama UML de mai jos.

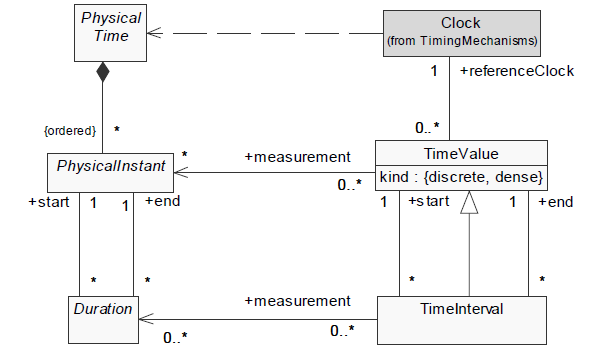


Figure 2 - Concepte uzuale de modelare de timp

In sens abstract, timpul fizic poate fi gandit ca o relatie ce impune o ordine partiala a evenimentelor. Se considera timpul fizic ca o progresie continua si nelimitata a *momentelor fizice*, asa cum sunt percepute de un observator, astfel incat acesta este:

* Set ordonat complet, ceea ce inseamna ca pentru oricare doua elemente distincte ale setului, p si q, fie p il preceda pe q fie q il preceda pe p.
* Set dens, ce presupune ca exista intotdeauna un moment de timp intre orice pereche de momente.

Cea din urma proprietate implica faptul ca modelul de timp fizic este continuu. In orice caz, intrucat computerele lucreaza doar cu numere de precizie finita, nu este intotdeauna posibil sa se reprezinte corect timpul fizic. Din acest motiv se poate face diferenta intre timpul dens, ce corespunde modelului continuu al timpului fizic, si timpul discret, care reprezinta timpul ce este impartit in cuante. Timpul dense poate fi reprezentat de un set de numere reale intrucat timpul discret corespunde numerelor intregi.

## Mecanisme de cronometrare

In modelarea infrastructurii de suport a sistemelor in timp real, este adesea necesar sa se identifice explicit acele aspecte ale modelului care reprezinta mecanismele de cronometrare, adica mecanismele care masoara timpul. Sunt doua tipuri de baza de mecanisme de cronometrare in acest model: cronometre si ceasuri.

Cronometrele sunt mecanisme care pot genera un timeout atunci cand se ajunge la un moment de tip specificat. Acesta poate fi momentul in care unele ceasuri ajung la valoarea predefinita sau cand un interval de timp specificat a expirat in raport cu un moment de timp dat (de obicei momentul in care a inceput cronometrul).

Aceste mecanisme de masurare a timpului sunt genul de resurse definite in modelul de resurse generic. Fragmentul de model prezentat in figura urmatoare ilustreaza mecanismele de cronometrare si conceptele legate de acestea.

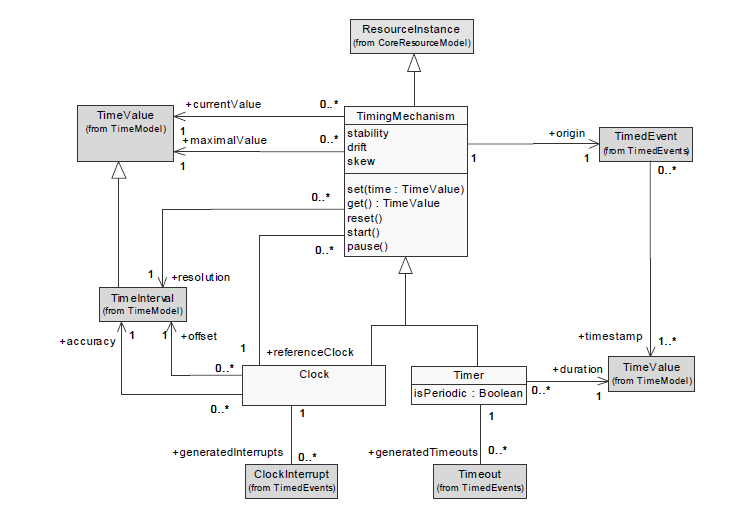


Figure 3 - Conceptele mecanismelor de cronometrare

Indiferent de tipul de mecanism, anumite proprietati caracterizeaza dispozitivele de cronometrare:

* Valoarea curenta care identifica progresul in timp
* Ceasul de referinta cu care este in legatura
* Originea, care este un eveniment clar identificat in timp de la care se incepa masurarea timpului
* Valoarea maxima de timp pe care valoare curenta nu o poate depasi
* Rezolutia, care este intervalul minim de timp ce poate fi recunoscut de mecanism
* Stabilitatea, reprezinta abilitatea mecanismului de a masura progresul timpului la o rata consistenta

In plus, in modelul prezentat se presupune ca un mecanism de cronometrare poate oferi urmatoarele servicii standard:

* Serviciul de *potrivire a timpului*, pentru setarea valorii de timp potrivita pentru mecanism
* Reset – serviciu pentru setarea mecanismului la valoarea initiala
* Pauza – serviciu folosit pentru a suspenda masurarea timpului, ceea ce inseamna ca valoarea curenta a mecanismului se opreste din progres
* Start – folosit pentru a reporni un mecanism aflat in pauza

# Schedulability modeling

In acest capitol se va descrie o componenta a profilului destinata analizei planificarii. Instrumente tipice pentru aceasta clasa de analiza a modelului ofera doua functii importante. Prima este sa calculeze capacitatea de planificare a sistemului; aceasta este abilitatea sistemului de a indeplini toate termenele limita definite pentru fiecare sarcina individuala. Asemenea instrumente identifica de obicei care entitati sunt planificabile si care nu. A doua funcie este de asistenta in determinarea modului in care sistemul poate fi imbunatatit. Acest lucru poate insemna sugestii pentru a face o entitate planificabila sau poate insemna optimizarea utilizarii sistemului pentru un sistem mai balansat. Un proiectant de sistem va analiza de obicei sistemul in cazul diferitelor scenarii folosind parametrii diferiti pentru fiecare scenariu in parte, mentinand aceeasi structura a sistemului per total.

Planificarea operatiilor reprezinta o parte importanta a oricarei aplicatii care lupta pentru utilizarea resurselor de executie ce executa planul – procesorul (numit mai general motor de executie).

Un mecanism de planificare defineste o tehnica de implementare folosita de un planificator pentru a lua decizii in legatura cu ordinea firelor de executie. Cateva exemple de mecanisme de planificare ar fi planificatoarele cu prioritate fixa, si planificatoarele cu cel mai devreme deadline. Algoritmii alesi au uneori acelasi nume cu mecanismul de planificare.

Pentru a analiza capacitatea de planificare a sistemului, trebuie sa se tina cont de:

* Politica de planificare
* Schema de arbitrare a resurselor
* Metoda de analiza a modelului

GRM (General Resource Modeling) reprezinta scheletul de baza pe care se bazeaza modelele UML. GRM este privit ca fundatia necesara pentru orice analiza cantitativa a modelelor UML si cuprinde doua puncte de vedere in stransa legatura. In primul rand se analizeaza din punct de vedere al domeniului, analiza ce cuprinde, intr-un mod generic, conceptele si schitele structurale si comportamentale uzuale ce caracterizeaza sistemele real-time si metodele de analiza a sistemelor real-time. Al doilea punct de vedere este cel al UML, in care se analizeaza elementele modelului UML.

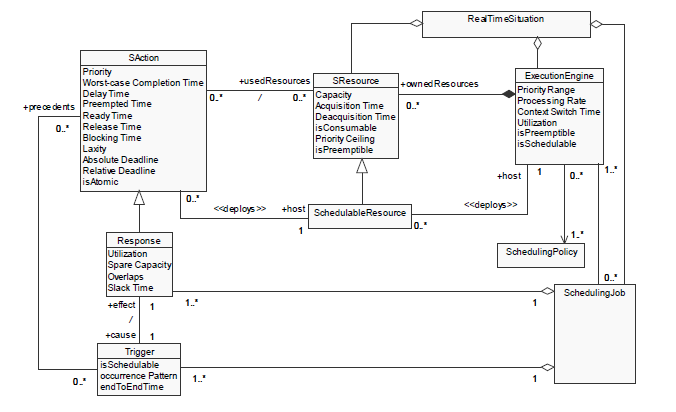


Figure 4 - Model de planificare de baza

Figura 4 reprezinta un model general de planificabilitate care identifica relatiile si abstractiile de baza folosite in analiza planificabilitatii. Acest model este definit ca o specializare a modelelor de baza, asa cum este aratat in Figura 5.

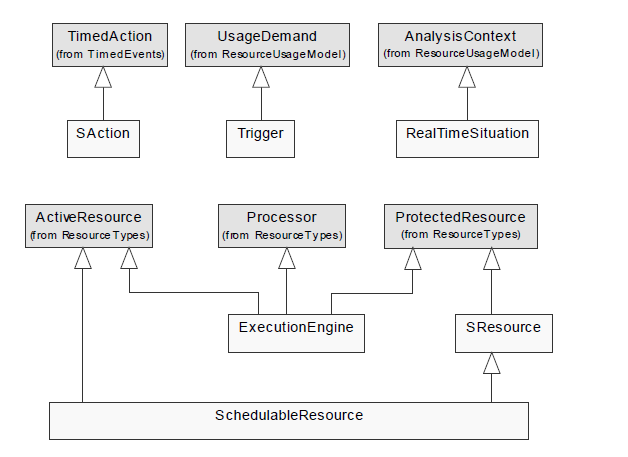


Figure 5 - Derivare de la GRM

ExecutionEngine a fost creat in scopul de a fi un procesor, excluzand posibilitatea de a fi un Device (dispozitiv) sau CommunicationResource (resursa de comunicatii), ceea ce intr-o analiza holistica ar permite actiunilor din SchedulingJob sa reprezinte codul de executie precum si mesajele din retea.

Asocierea intre SchedulingJob si Trigger-ul sau a fost de tipul one-to-one. Acest lucru inseamna ca SchedulingJob putea fi stimulat doar de un singur Trigger. Sunt cazuri in care cateva Triggere pot fi implicate in SchedulingJob si invers, un Trigger porneste anumite SchedulingJobs. De aceea s-a adoptat o noua specificatie, prin care se permite ca mai multe Triggere sa fie asociate unei SchedulingJob, fiecare cu propriul raspuns, considerand raspunsurile diferite ca parte din SchedulingJob.

Asocierea intre SchedulingJobs si Triggerele sale poate fi observata in Figura 6.

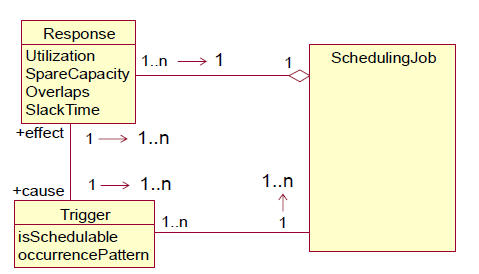
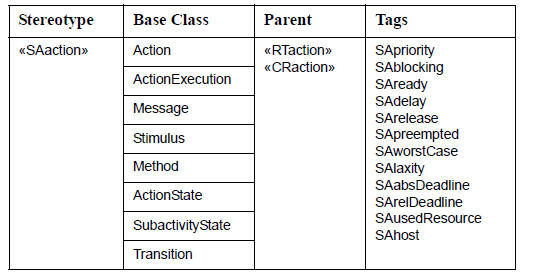


Figure 6 - Relatia intre SchedulingJob si Trigger

Saction reprezinta conceptul domeniului de actiuni, cu urmatoarele clase si taguri.



**3. Performance modeling (Matei Ioan - Tiberiu)**

In aceasta ramura a profilului UML SPT vom descrie componenta responsabila de analiza generala de performanta a modelelor UML. Acest profil ofera facilitati pentru: captarea sau intregistrarea cerintelor de performanta in cadrul contextului de design a modelului, asocierea caracteristicilor de performanta QoS (Quality of Service) cu elemente specifice modelului UML, specificarea parametriilor de executie ce pot fi folositi de unelte de modelare pentru a calcula si pentru a prezice caracteristicile de performanta si redarea rezultatelor de performanta de catre uneltele aferente sau descoperite in cadrul testarii.

Cele mai specifice unelte pentru acest tip de model ofera doua functii importante:

- estimarea performantei unei instante a sistemului prin folosirea unui tip de model

- asistarea cu determinarea felului in care un sistem poate fi imbunatatit prin identificarea problemelor si a resurselor critice

De regula, un proiectant al sistemului va dori efectuarea analizei produsului prin testarea anumitor scenarii folosind parametri diferiti pentru fiecare scenariu in timp ce va pastra aceeasi structura a sistemului.

3.1. Conceptul Domain Viewpoint

Pentru a intelege mai bine analiza de performanta, vom descrie ceea ce numeste '*Domain Viewpoint*', notiune ce identifica structurile abstracte de baza prezente in cadrul analizei. Aceste structuri abstracte, precum si legaturile gasite sunt, de cele mai multe ori, explicate cu ajutorul unui model UML. In cadrul Domain Viewpoint gasim urmatoarele concepte esentiale precum si legaturile aferente:

- scenariu (cerintele QoS sunt plasate la nivel de scenariu ce defineste calea de raspuns a carei finalitate este observabila din exterior)

- fiecare scenariu este executat de asa numitele clase '*job´* sau '*user*' (a nu se confunda confunda cu partea software a claselor, de aceea clasele job si user se mai numesc *workload* - *open workload* si *closed workload*)

- pasii scenariului sau activitati se regasesc sub forma elementelor scenariilor, elemente constituente unei secvente cu legaturi de tipul succesor - predecesor ce includ fork, join sau loop

- fiecare pas are un numar mediu de executie precum si un *host execution demand*

- cerintele de resurse

- resurse

- timpul de funtionare a unei resurse active se defineste ca cererea de executie a host-ului a pasilor indepliniti de resursa

- masurile de performanta ale unui sistem includ utilizarea resurselor iar fiecare masura poate fi definita in versiuni diferite, unele pot fi specificate chiar in cadrul aceluiasi model cum ar fi: a valoare ceruta, o valoare aproximata, o valoare estimata sau chiar o masurare a valorii.

Tipuri de metode ale analizei de performanta

Orice sub-profil constituent analizei generale de performanta ar trebui sa poata suporta unelte si limbaje de modelare pentru contruirea diverselor modele de performanta. Astfel, putem identifica mai multe tipuri de modele de performanta:

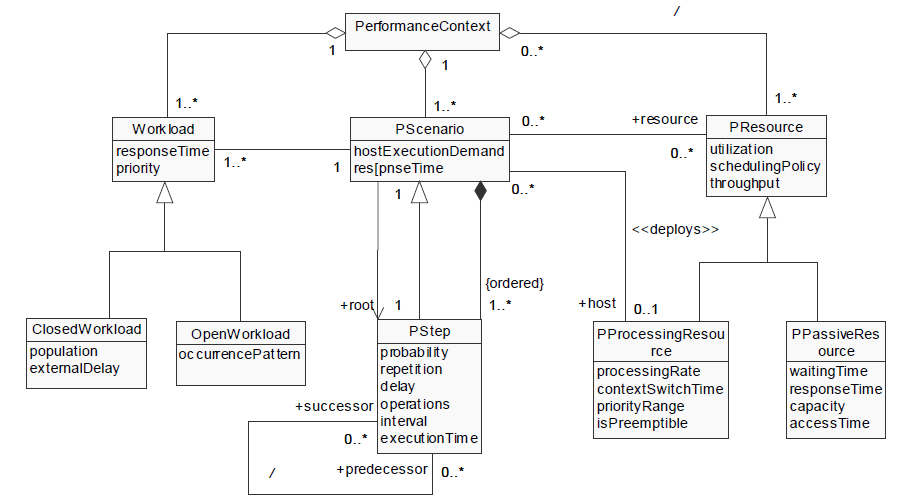
1. *Queueing model* ce defineste clasele de tip *customer* ce executa anumite aspecte ale produsului software, aspecte captate in cadrul diverselor scenarii. De remarcat faptul ca modelele de tip queueing calculeaza throughput-ul mediu, utilizarea si timpul de raspuns al claselor.

2. *Simulation model* ce defineste multipli pasi logici pentru executia software-ului

3. Modele de stare discreta de felul *Petri Nets*

3.2. Modelul de tip *Domain*

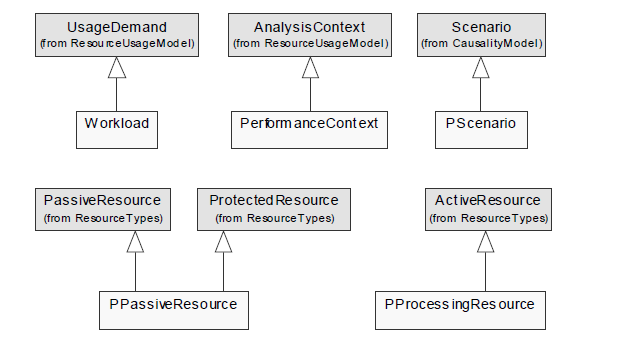
Analiza de performanta este, de cele mai multe ori, bazata pe instante. Se aplica modelelor ce pot inregistra actuale sau presupuse cicluri de executie ale sistemului software ce se bazeaza pe anumite instante. De aceea, conceptele de tip *domain* se refera mai degraba la instante decat la descriptori, cum ar fi clasele (Fig. 7).



*Fig. 7. Modelul de tip domain al analizei de performanta*

*(*[*http://www.omg.org/spec/SPTP*](http://www.omg.org/spec/SPTP)*)*

De asemenea, putem reprezenta si legaturile prezente in cadrul modelului de tip *domain* (Fig. 8). [1]



*Fig. 8. Legaturile in cadrul ceonceptelor de performanta*

*(*[*http://www.omg.org/spec/SPTP*](http://www.omg.org/spec/SPTP)*)*

Analiza de performanta are propria terminologie si nu este perfect standardizata. Putem aplica anumite definitii folosind profilul specific QoS (Quality of Service). Folosind profilul Qos, fiecarui obiect al proiectarii i se poate asocia un obiect QoSValue ce contine atribute aferente QosSlotDimension. Atributele specifice masurii performantei sunt in permanenta deschise schimbarii pentru a se modela fiecarei probleme intalnite. Acest lucru poate fi asemanat cu partea introductiva a mecanismelor standard de extensie in cadrul UML. In anumite probleme este necesara definirea masurarii specifice a problemelor. Acest lucru se poate face, in esenta, prin definirea intarzierilor intre anumite evenimente. De cele mai multe ori, o masura a performantei este ori o intarziere intre doua evenimente predefinite, ori se poate calcula din masurari ale intarzierilor. Intre momentul de start si momentul de final, intarzierea este reprezentata de timpul de raspuns sau de o resursa ce se afla in stransa legatura cu timpul. Calculand valoarea inversa a acestor intarzieri vom obtine rata de throughput, iar rata de throughput multiplicata cu timpul asociat va determina obtinerea utilizarii de resurse. O masura definita asociata poate fi gasita in cadrul clasei *Duration* din UML 2, clasa ce identifica un eveniment de start, final si o valoare a intarzierii. [2]

3.3. Contextul de performanta

Contextul de performanta specifica unul sau mai multe scenarii ce se utilizeaza pentru explorarea diverselor situatii dinamice ce presupun un anumit set de resurse. Pentru anumite specificatii de sistem putem avea mai multe cazuri ale contextului de performanta ce contine suprapunerea anumitor resurse, dar cu scenarii aferente fiecarui tip de context de performanta.

Componentele contextului de performanta pot contine valori parametrizate ale QoS pentru a permite studiul in detaliu asupra QoS, dar structura unui context de performanta este fixa si contine asocieri (resurse, scenariu si workload).

1. Scenariul (scenario) este o secventa alcatuita din unul sau mai multi pasi. Acesti pasi sunt ordonati si se supun unei legaturi de tipul predecesor/succesor. De mentionat faptul ca un pas din cadrul scenariului poate avea mai multi succesori datorita operatiei de *forking*. Scenariul contine urmatoarele atribute:

- *hostExecutionDemand*

- *responseTime*

Sceariul se afla in stransa asociere cu:

- *host*

- *pas(step)*

- *root*

- *workload*

- *context de performanta (performanceContext)*

2. Pasul (*step*) se gaseste sub forma unui model de incrementare in executia unui sccenariu specific si foloseste anumite resurse pentru indeplinirea functiei sale. De regula, pasul dureaza un timp limitat pentru a se executa. Pasii se afla in stranse legaturi cu cei ce preced sau succed intr-o relatie. Pasul contine urmatoarele atribute:

- *hostExecutionDemand*

- *intarziere (delay)*

- *timpul de raspuns (responseTime)*

- *probabilitate (probability)*

- *interval*

- *repetitie (repetition)*

- *operatii (operations)*

Pasul contine urmatoarele entitati ca asocieri:

- *predecesor*

- *succesor*

- *scenariu*

- *host*

- *resursa*

3. Resursa, in sens abstract, participa in cadrul scenariilor contextului de performanta. Resursa prezinta urmatoarele atribute:

- *utilizare*

- *throughput*

- *schedulingPolicy*

Resursele presupun asocierea de tip *performanceContext*.

4. Resursa de procesare (*ProcessingResource*) poate fi un dispozitiv, precum un procesor, ce presupune pasi de procesare alocati in cadrul proiectarii sistemului. Aceasta resursa contine urmatoarele atribute:

- *schedulingPolicy*

- *rata de procesare (processingRate)*

- *contextSwitchTime*

- *domeniul de prioritizare (priorityRange)*

- *isPreemtable*

Resursele de procesare presupun asocieri cu scenariul (*scenario*).

5. Workload-ul specifica intensitatea cerintelor de executie a unui scenariu specific precum si timpul de raspuns estimat asociat workload-ului. Contine urmatoarele atribute:

- *timp de raspuns (responseTime)*

- *prioritate*

- *scenariu*

6. Valorile de performanta ajuta la realizarea completa a analizei de performanta (prin colaborare cu valorile numerice ale caracteristicilor de performanta de tip QoS). Valorile de performanta au, de regula, forma:

<performance-value> ::= <source-modifier> <type-modifier> <time-value>

3.4. Reprezentarea in UML

Considerand toate modelele conceptului *Performance Domain* putem realiza o echivalenta a acestora in cadrul UML. Din moment ce performanta reprezinta un teritoriu dinamic, scenariul joaca un rol important in determinarea performantei unui sistem in cadrul modelelor UML. De cele mai multe ori, scenariile, in cadrul UML, sunt modelate direct prin folosirea unor colaborari sau a unor grafuri de activitate. Prin prisma acestor doua modelari putem realiza echivalenta cu elementele conceptului *Performance Domain*.

Analiza pe baza colaborarii contine: o analiza bazata pe colaborari si o analiza bazata pe instante ale colaborarii. Semantica este asemanatoare in ambele cazuri, alegerea depinde de circumstante sau preferinte individuale. Analiza pe baza colaborarii contine urmatoarele atribute:

- *PerformanceContext* (un context de performanta este modelat sub forma unui stereotip «PAcontext» a unei colaborari de tip UML

- *Scenariu* ( scenariul se bazeaza pe o interactiune sau pe un set al instantelor de interactiune)

- *Pas (Step)* reprezinta executia unei actiuni oarecare si se poate identifica prin asocierea unei pas cu stereotipul «PAstep»

- *Workload*-ul este specificat ca stereotipul unui pas; acest stereotip poate fi aplicat doar primului pas al scenariului

- *Resursa de procesare (ProcessingResource)* se poate modela prin asocierea unui stereotip aferent de tipul «PAhost» cu orice rol de clasificare, rol ce executa pasii scenariului

Analiza pe baza grafului de activitate presupune captarea scenariului in cadrul unui graf de activitate. Aceasta forma prezinta anumite avantaje datorita naturii ierarhice mult mai explicita a grafurilor de activitate. Analiza pe baza grafului de activitate contine urmatoarele atribute:

- *PerformanceContext* ce poate fi modelat cu ajutorul unui graf de activitate ce se afla sub stereotipul «PAcontext». Din moment ce graful de activitate contine doar o singura activitate, aceasta analiza este restransa la un singur scenariu per context

- *Scenariul* este modelat de un set de stari sau activitati si tranzitii ale grafului de activitate

- *Pasul (Step)* ce se regaseste in cadrul stereotipului «PAstep» ce contine, de altfel, si fiecare actiune sau stare a unei subactivitati

- *Workload*-ul este modelat cu ajutorul unor valori asociate primului pas in cadrul contextului de performanta si poate fi gasit in cadrul stereotipurilor «PAopenLoad» sau «PAclosedLoad»

- *Resursa de performanta (ProcessingResource)* se poate modela prin asocierea stereotipului specific «PAhost» cu un graf de activitate ce se afla in legatura cu un obiect specific sau cu un clasficator [3]

Putem observa faptul ca analiza de performanta se bazeaza pe instante.

4. Bibliografie

1. UMLTM Profile for Schedulability, Performance, and Time Specification, January 2005 Version 1.1 formal/05-01-02

2. Capabilities of the UML Profile for Schedulability Performance and Time (SPT), Murray Woodside and Dorina Petriu

Dept of Systems and Computer Engineering, Carleton University, Ottawa Canada

{cmw | petriu} @ sce.carleton.ca

April 2004

3. UMLTM Profile for Schedulability, Performance, and Time Specification,

An Adopted Specification of the Object Management Group, Inc.