# Aspecte privind specificarea concurenţei utilizând diagramele comportamentale UML

Dumitru CIORBĂ
*Universitatea Tehnică a Moldovei*
*dumitru.ciorba@ati.utm.md*

***Abstract* — Două treimi din organizaţiile de dezvoltare software şi peste 10 milioane de specialişti IT (conform unor date ale *BZ Research* şi, respectiv, în baza analizelor *Gartner, Inc.* [1]) utilizează limbajul unificat de modelare UML (*Unified Modeling Language,* [2] ). Astfel acest limbaj acum este unul din cele mai populare limbaje de specificare şi modelare a sistemelor informaţionale. Totuşi nici versiunea 2.4 a specificaţiei infrastructurii limbajului unificat de modelare (UML) [2] nu poate fi considerată completă în perspectiva specificării concurenţei (fapt susţinut şi în [3 p. 487]). Lipsa unor funcţionalităţi caracteristice ei este compensată într-o măsură oarecare de posibilităţile de extindere ale limbajului UML. În ceea ce urmează se vor descrie aspecte pe care un dezvoltator trebuie să le ia în considerare la specificarea concurenţei în diagramele UML.**

***Cuvinte cheie* — concurenţă, specificare, UML.**

## I. INTRODUCERE

Concurenţa ţine, în special, de aspectele dinamice ale sistemului. Ea nu are cum, în mod implicit, să fie expusă în *diagramele structurale*, care prin natura lor nu sunt dinamice. Explicit în diagramele de clasă concurenţa se defineşte prin *clasele active*, care prevăd existenţa unui *fir de control* în fiecare obiect. În legătură cu această noţiune, care este determinată de atributul „*isActive*”, în perspectiva semantică atrag atenţia cel puţin două chestiuni: 1) limbajul UML defineşte *clasele* ca fiind active şi nu obiectele acestora, deşi tocmai ele definesc interacţiunile dinamice; 2) termenul abstract *fir de control* se înţelege separat de termenul *fir de execuţie* (din domeniul sistemelor de operare), deşi în contextul implementării relaţia este evidentă. Deci este rezonabil să fie impuse precizări prin mecanisme complementare oferite de limbaj.

Totuşi utilizarea multiplelor mecanisme poate crea confuzii. În lucrarea [4] se afirmă că pentru tratarea corectă a concurenţei, mecanismele limbajului de modelare trebuie utilizate în concordanţă reciprocă. De exemplu, stereotipul „*concurrent*” nu poate fi aplicat operaţiilor membre ale unei clase active, ci doar claselor pasive, care nu-şi au propriul fir de control [2 p. 447], iar într-o clasă activă operaţiile sunt implicit de tip „*sequential*” etc. Limbajul UML nu defineşte restricţii de combinare ale mecanismelor enunţate. Astfel dezvoltatorii trebuie să ia în calcul posibile „efecte laterale” rezultate din modelele diagramelor de clasă, care pot fi evitate sau, cel puţin, explicate specificând concurenţa şi în alte modele UML.

## II. DIAGRAMA CAZURILOR DE UTILIZARE

*Diagrama cazurilor de utilizare* este o abstracţiune generală ce permite exprimarea naturală a concurenţei. Totuşi în perspectiva implementărilor, semantica cazurilor de utilizare reduce din utilitatea acestora pentru specificarea concurenţei. Acest fapt se explică prin aceea că prin definiţie cazurile de utilizare exprimă interacţiuni ale sistemului cu actorii externi, deci scenariile nu sunt parte de sistem. Astfel deşi sunt definitorii pentru sistem, specificându-i cerinţele, cazurile de utilizare rămân încă a fi utilizate în mod neformal.

## III. DIAGRAMELE DE SECVENŢĂ

Având la bază logica de cauzalitate, *diagramele de secvenţă* pot prezenta interacţiuni concurente inter-obiect. Totuşi mecanismele existente par a fi inflexibile şi neîndemânatice, în special, în cazul specificării unor comportamente concurente complexe. De exemplu, mesajele *m4 (1a.2)* şi *m6 (1a.1.1)* din figura 1 sunt concurente. Deci aceste două mesaje ar trebui să fie diferite în finalul expresiei prin marcajul literal. Dar metoda de „numerotare” a mesajelor (ce include şi nivelele imbricate) nu evidenţiază acest fapt. Astfel marcarea mesajelor recomandată de standard nu este o soluţie „sigură” pentru recunoaşterea mesajelor concurente.

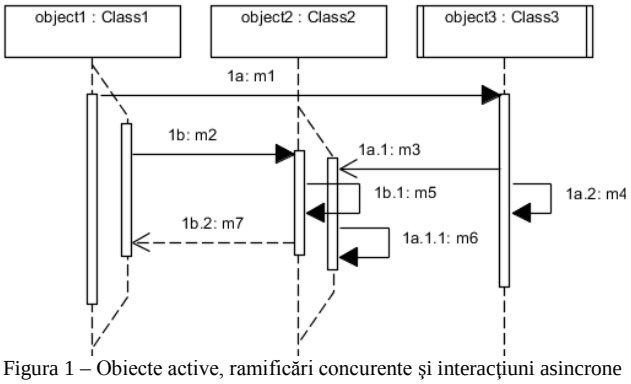


Figura 1 – Obiecte active, ramificări concurente şi interacţiuni asincrone în diagrama de secvenţă

Sincronizarea este un aspect important al concurenţei. Dar cu regret sincronizarea nu-şi are o notaţie specială în diagramele de interacţiuni. Astfel, în cazul când se doreşte de specificat o activitate de sincronizare se recurge la diverse tehnici, ceea ce poate genera interpretări inexacte. Desigur fragmentele şi operatorii de interacţiune ar putea diminua din aceste efecte, dar careva notaţii de sincronizare

ar fi mai potrivite pentru aceasta.

Obiecţii ce ţin de aspecte fundamentale ale *diagramelor de secvenţă* sunt expuse în [5], în care se menţionează că diagramele definesc implicit comportamente *posibile*, iar comportamentele ce *trebuie* să fie îndeplinite se impun în model într-o manieră ambiguă prin fragmente cu operatori *assert* şi invariante de stări (*state invariant* – eng.) asortate cu expresii OCL. Se poate completa observaţiile enunţate şi cu cele scrise în specificaţia limbajului UML care afirmă că uniunea secvenţelor de evenimente nu trebuie neapărat să acopere întregul univers al secvenţelor [2 p. 496]. Deşi din specificaţiile UML reiese că aceste diagrame nici nu trebuie să ofere mai mult, căci ele definesc un comportament emergent (*emergent behavior* – eng.), totuşi eventualitatea şi incompletitudinea modelelor definite de ele par a fi inconveniente majore.

## IV. DIAGRAMELE DE STARE ŞI DE ACTIVITĂŢI

Mecanismele şi semantica *intra-obiect* face mai potrivite pentru specificarea concurenţei anume *diagramele de stare* şi *de activităţi*. Comportamentele de executare (*executing behavior* – eng.), definite de aceste diagrame, *pot* oferi o descriere mai completă şi certă. Dar şi aici persistă semne de întrebare. De exemplu, ce ar însemna în perspectiva implementării pentru un obiect pasiv regiunile ortogonale (concurente) ale unei stări (figura 2)? Doar faptul că anumite operaţii sunt reciproc independente? Într-un obiect pasiv, ce nu are propriul fir de control, acest fapt poate genera interpretări indezirabile. Nici pentru obiectele active semantica stărilor compuse (cu regiuni ortogonale) nu este descrisă fără echivoc, căci limbajul UML nu face distincţie clară între modele nedeterministe întrepătrunse (*interleaving* – eng.) şi modele adevărat concurente (*true concurrent* – eng.)/paralele[6,7].

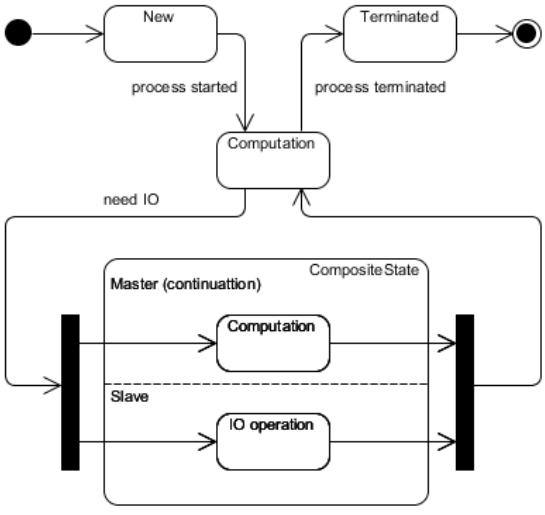


Figura 2 – Proces cu stări ortogonale (concurente)

Un alt aspect, menţionat în [4], pe care un dezvoltator trebuie să-l ia în considerare este faptul că în moment ce obiectele pasive nu-şi au un propriu fir de control, maşinile de stare ale acestora nu au un „domeniu” propriu de execuţie. În consecinţă maşina de stare al obiectului pasiv „rulează” pe un fir de control al altui obiect. În acest caz apare o întrebare: cum şi cine gestionează cererile noi parvenite la obiectul pasiv în timp ce el „deserveşte” deja un alt obiect? Desigur în particular o operaţie poate fi definită ca fiind concurentă (şi atunci lucrurile sunt clare), dar acest fapt nu este extins pe întreaga maşină de stare.

## V. CONCLUZII

Limbajul UML este un limbaj complex, care încearcă să cuprindă multiple aspecte ale modelării de sistem. Dar datorită complexităţii şi lipsei descrierilor precise, construcţiile de limbaj conduc la interpretări ambigue ale modelelor. Deci pentru precizarea elementelor UML de modelare este necesară îmbunătăţirea semanticii, care poate fi realizată prin două strategii: *de restrângere*, care are ca scop oferirea unei semantici neambigue a unei părţi restrânse ale UML prin simplificarea meta-modelului; şi *de extindere*, care se bazează pe ideea că UML nu este un limbaj de modelare, ci o familie de limbaje de modelare, astfel se operează la diverse nivele ale arhitecturii meta-model pentru specificarea unui nou membru al familiei [8].

Fiecare strategie are tehnici specifice, dar soluţia universală pentru perfecţionarea limbajului este *formalizarea* limbajului [5,9,8]. Rigurozitatea semanticii pe lângă faptul că va oferi o mai bună înţelegere, va asigura coerenţa şi corectitudinea modelelor. Având descrieri exacte ale acestora, implementările vor putea fi validate la corespunderea cu specificaţiile. Iar tehnicile ingineriei inverse vor evolua şi vor fi posibile nu doar pentru diagramele de clasă. Demonstrarea corectitudinii, datorită specificaţiilor formale, va fi mai facilă utilizând tehnicile automatizate ale instrumentelor de proiectare.

## BIBLIOGRAFIE

[1] **A. Watson.** UML® vs. DSLs: A false dichotomy. *Object Management Group.* [Online] Septembrie 03, 2008. [Cited: Mai 13, 2011.] http://www.omg.org/cgi-bin/doc?omg/08-09-03.pdf.

[2] **Object Management Group Inc.** UML Superstructure Specification, v2.4. *The Object Management Group.* [Online] 01 2011. [Cited: 05 21, 2011.] http://www.omg.org/spec/UML/2.4/Superstructure.

[3]**G. Hillar.** *Professional Parallel Programming with C#.* Indianapolis, Indiana : Wiley Publishing, Inc., 2011. ISBN: 978-0-470-49599-5.

[4] **S. Gérard, I. Ober.** Parallelism/Concurrency specification within UML. [ed.] C. Atkinson, et al. *Workshop on Concurrency Issues in UML*. 2001.

[5] **J. Küster-Filipe.** *Modelling concurrent interactions.* Theoretical Computer Science, Februarie 21, 2006, Vol. 351, 2, pp. 203-220.

[6] **D. Ciorbă, V. Beşliu.** *Architecting software concurrency.* Computer Science Journal of Moldova (ISSN 1561-4042), 2011, Vol. 19, 1 (55), pp. 92-108. http://www.math.md/files/csjm/v19-n1/v19-n1-(pp92-108).pdf.

[7] **V. Sassone, M. Nielsen, G. Winskel.** Models for Concurrency: Towards a Classification. *Theoretical Computer Science.* 1996. Vol. 170, 1-2, pp. 297-348.

[8] **M. Terrasse, M. Savonnet.** *Formalization of the UML Metamodel: An Approach Based Upon the Four-Layer Metamodeling Architecture.* ECOOP'00 Workshop 14 on Defining a Precise Semantics for UML, 2000. http://ufrsciencestech.u-bourgogne.fr/~terrasse/Ecoop00.pdf.

[9] **L. Peng.** *Formalization of uml using algebraic specifications.* Faculty of Science, Vrije Universiteit Brussel. 2001. Master Thesis.