Clasificări ale limbajelor de specificare

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

***Abstract* — Specificarea defineşte totalitatea cerinţelor faţă de sistemul în elaborare pe care un dezvoltator le va implementa în mod obligatoriu. Multiple limbaje de specificare sunt cunoscute şi acceptate (mai mult sau mai puţin) în ingineria software. Deşi descrierea cerinţelor este scopul primar al specificărilor, totuşi actuale şi utile sunt considerate doar acelea care permit verificarea/demonstrarea calităţilor enunţate. Clasificările permit înţelegerea şi utilizarea limbajelor de specificare, astfel în ceea ce urmează se prezintă succint diverse clasificări ale acestora.**

***Cuvinte cheie* —clasificare, formalism, specificare.**

I. SPECIFICĂRI ALE ETAPELOR DE DEZVOLTARE

Specificarea completă şi corectă este o sarcină dificilă, căci trebuie să cuprindă diverse informaţii: de la contextul în care va funcţiona sistemul la ce trebuie să facă sistemul. Există diferite tipuri de specificări, caracteristice diferitor etape de dezvoltare [1,2]:

- ***Specificarea cerinţelor funcţionale şi nefuncţionale ale utilizatorului*** (user requirements – eng.) se aplică la faza de analiză a cerinţelor şi defineşte comportamentul extern aşteptat de utilizatorii sistemului în elaborare; se exprimă, de regulă, informal în termeni de utilizator;
- ***Specificarea cerinţelor de sistem*** (software requirements – eng.) conţine descrieri ale funcţiilor în termeni de sistem; contribuie la completarea definiţiilor din faza anterioară, respectând un grad de detaliere nu prea mare; deseori documentarea este combinată cu diagrame ale fluxurilor de date sau entitate-relaţie, sau urmând standarde industriale precum UML;
- ***Specificarea arhitecturală*** urmăreşte obţinerea unei structuri de sistem, exprimată în termeni de componente şi a legăturilor logice dintre acestea; termenul „componente" este în mod deliberat vag şi este determinat de tipul problemei; se poate acoperi practic toate tipurile de entităţi software, cum ar fi procese, programe, clase, baze de date, etc.;
- ***Specificarea tehnică*** aplicată la proiectarea în detaliu defineşte cu un nivel de granularitate mic cum se vor implementa algoritmii de funcţionare a fiecărei componente, structurile de date asociate acestora şi interconexiunile din sistem; deşi specificaţia, în această fază, este influenţată de limbajul de implementare, totuşi aspectele sintactice nu primează în cazul proiectului.

II. DIMENSIUNI ALE UNEI SPECIFICĂRI BAZATE PE UN SISTEM LOGIC

Obţinerea unor specificaţii ce ar „produce" arhitecturi cu înalte caracteristici necesită metode, care se bazează pe un anumit sistem logic raportat la teoria dezvoltării software ce cuprinde, la rândul său, definiţii, reguli, principii, algoritmi, etc. Conform [3], orice sistem logic poate fi definit în opt dimensiuni:

- ***sintactică***, ce defineşte regulile de formare a enunţurilor sistemului utilizând simboluri primare (alfabetul);
- ***semantica***, ce defineşte regulile de formare a funcţiilor de interpretare prin care enunţurilor le sunt asociate valori de adevăr;
- ***algebrică***, ce defineşte modul în care se combină între ele valorile de adevăr, organizate într-o structură algebrică, căreia îi sunt puse în corespondenţă operaţii algebrice;
- ***topologică***, ce permite studiul de proprietăţi ale sintaxei şi semanticii ce pot avea echivalente topologice; dimensiunea poate adăuga subtilităţi de interpretare şi de demonstraţie;
- ***probabilistă***, ce defineşte axiome probabilistice care exprimă comportamentul funcţiilor de adevăr faţă de operaţiile logice;
- ***algoritmică***, ce defineşte aspectele computaţionale; şi este determinată de rezolvabilitatea prin algoritmi a problemei;
- ***categorială***, ce oferă generalitate şi flexibilitate în tratarea unor probleme existente în sisteme particulare;
- ***filozofică***, ce oferă explicaţii asupra originii şi a semnificaţiei sale, precum şi a mecanismelor sale interioare.

III. SPECIFICĂRI CU DIVERSE GRADE DE FORMALITATE

Exactitatea şi *claritatea* sunt cerinţe ce asigură precizia [4 p. 9], trăsătură importantă a unei specificaţii calitative. În acelaşi timp precizia nu este în mod obligatoriu asociată expresiilor matematice[5 p. 30]. Astfel deşi majoritatea limbajelor tind să specifice formal sistemul în termeni matematici, aşa încât să-i poată fi dovedite calităţile în timpul verificării, totuşi există metode pentru care *înţelegerea* este un criteriu mai relevant. În această perspectivă se pot distinge următoarele tipuri de specificări:

- ***informale***, care fiind exprimate în limbaj natural, sunt

înţelese şi de utilizatori ne-matematicieni/ne-specialişti, dar lipsite de reguli stricte de structurare/formare sunt dificile de analizat;
- ***semiformale*** implică notaţii grafice şi expresii definite riguros, care permit descrieri de modele compuse; pot fi complimentare cu diverse metode de specificare fără a denatura modelul;
- ***formale,*** bazate pe diverse aparate matematice (logică clasică şi temporală, teoria mulţimilor, etc.), sintaxa şi semantica acestora permit descrieri de structuri de date abstracte şi complexe.

IV. SPECIFICĂRI BAZATE PE FORMALISME MATEMATICE

Automatizarea verificării trăsăturilor esenţiale/cerute ale arhitecturilor prevede stricteţe în descrieri determinate de un context matematic. Specificul aparatului matematic cauzează deosebiri majore dintre limbaje. Astfel prin formalismul utilizat în lucrarea [6] se identifică patru clase de limbaje de specificare bazate pe:
- ***logica de ordinul întâi şi teoria mulţimilor***: deseori numite orientate pe model, căci specifică sistemele construind modele matematice pentru ele; clasa cuprinde limbaje ca Z/Z++/Object-Z, VDM (*Vienna Development Method*)/VDM++, etc.
- ***abordări algebrice***: utilizează ecuaţii algebrice pentru a stabili semantica operaţiilor; cuprinde limbaje ca Maude, Larch, etc.
- ***reţele Petri/de nivel înalt***[7]: definesc modele operaţionale prin stări, reprezentate de *poziţii* (*place* – eng.) ce pot conţine jetoane (*token* – eng.), şi tranziţii ce pot specifica pre- şi post-condiţii; cuprinde limbaje ca CPN (*Colored Petri Nets*), OPN (*Object Petri Nets*), etc;
- ***logica temporală***: reprezintă formalisme axiomatice utile în descrierea proprietăţilor sistemelor reactive şi concurente în termeni de timp; cuprinde limbaje ca LTL (*Liniar Temporal Logic*), OO-LTL, etc.

Puţin o altă perspectivă asupra limbajelor enunţate regăsim în [8 p. 223], care clasifică modelele comportamentale rezultante din specificaţii în:
- ***modele de procese***: comportament descris în termeni de mulţimi de trasee, diagrame pentru tranziţii de stări (*state-transition* – eng.), etc.;
- ***modele algebrice***: abordare funcţională;
- ***modele definite pe logici modale***: comportament descris în termeni de reguli, utilizând logici temporale, dinamice sau deontice.

Această perspectivă este apropiată altei clasificări, care oferă trei clase generice de limbaje în dependenţă de semantică [9]:
- ***limbaje operaţionale***, scopul cărora este de a descrie, cum este realizat calculul, deci specificaţiile definesc modele cărora le pot fi asociate *maşini abstracte* ce descriu tranziţii induse de stări;
- ***limbaje denotaţionale***, definesc specificaţii ce includ construcţii (inclusiv matematice), care oferă semnificaţii prin o mulţime finită de asociaţii între argumente şi valori, determinate de un domeniu abstract (dar bine determinat, precis); limbajele declarative corespund acestei categorii;
- ***limbaje axiomatice***, bazate pe logica matematică, definesc semnificaţii ale entităţilor specificate prin *aserţiuni*, ce sunt predicate logice în care variabilele pot descrie stări; un exemplu canonic al acestei categorii este *logica Hoare*.

Toate limbajele formale cunoscute pot fi atribuite uneia din aceste clase. Totuşi multe limbaje reprezintă variaţii ale acestora prin combinarea celor de bază (semantica acţiunilor, semantica algebrică, semantica concurentă, definită prin calculul proceselor sau modelul actor etc.).

V. SPECIFICĂRI MULTIDIMENSIONALE

Studiile comparative ale metodelor/tehnicilor de specificare definesc de obicei un cadru de comparaţie bazat pe un set întreg de criterii pentru a remarca trăsăturile necesare/specifice. Totuşi relevante pot fi considerate acele cadre, criteriile cărora se regăsesc în câteva dimensiuni conceptuale, astfel rămânând valabile zeci de ani. Un exemplu poate servi cadrul prezentat în [10], în care se compară două tehnici (IDEF0 – *Integration Definition for Function Modeling* şi DFD – *Data flow diagram*) în termeni *al procesului de modelare* şi *al rezultatului final*, deosebind patru dimensiuni:
- ***sintactică***, include criterii de evaluare a regulilor sintactice;
- ***semantică***, se referă la aspectele ce ţin de semnificaţii, asigurând consistenţa şi lipsa ambiguităţii în specificaţii;
- ***comunicabilităţii,*** se referă la lizibilitatea (*readability* – eng.) şi înţelegerea/inteligibilitatea (*understandability* - eng.) documentului rezultat din folosirea tehnicii/metodei.
- ***uzabilităţii***, se referă la uşurinţa de utilizare în mod eficient a tehnicii/specificaţiei obţinute.

În rezultatul analizei multiplelor studii comparative (inclusiv cel menţionat mai sus) în [11 p. 522] se oferă criterii relevante utilizabile într-un cadru de comparaţie a metodelor/tehnicilor de definire:
- *gradul de înţelegere şi dificultatea de utilizare;*
- *capacitatea de a servi ca bază pentru concepţie, construire şi testare;*
- *capacitatea de a servi ca bază pentru generarea automată de prototipuri;*
- *capacitatea de a servi ca bază pentru obţinerea automată a codului;*
- *asigurarea mijloacelor de verificare a automată a inexistenţei ambiguităţilor, inconsistenţelor şi incompletitudinilor;*
- *potenţialul oferit elaboratorilor cerinţelor de a gândi şi scrie în termenii comportării externe a sistemului;*
- *potenţialul sintactic şi semantic privind organizarea şi reprezentarea informaţiilor;*
- *capacitatea de a servi ca bază pentru generarea automată a testelor;*

– *capacitatea de a captura paradigmele fundamentale şi/sau specifice ale noului sistem.*

Criteriile definesc un spectru larg de evaluare. În rezultat, nu toate metodele pot acoperi toate aspectele vizate. Iar evaluarea înaltă a unui criteriu, poate implica evaluare joasă la un alt criteriu. De exemplu înţelegerea şi uşurinţa de utilizare sunt invers proporţionale cu nivelul de complexitate şi formalism.

VI. CONCLUZIE

În final se poate menţiona că alegerea setului de criterii ale unei evaluări/selecţii de metode în mare parte este determinată de multiple aspecte ce ţin atât de procesul de specificare, cât şi de rezultatul propriu-zis al specificării, care poate determina succesul sau insuccesul întregului proiect/proces de dezvoltare.

BIBLIOGRAFIE

[1] **J. Tilbury.** *Software specification.* s.l. : Tessella Support Services plc, September 1999. http://www.tessella.com/wp-content/uploads/2008/05/softwarespecification.pdf.

[2] **J. Chaar.** *Software design methodologies : a survey.* Department of Electrical Engineering and Computer Science, The University of Michigan. 1987. Technical report. http://hdl.handle.net/2027.42/4077.

[3] **G. Georgescu.** Dimensiuni ale unui sistem logic. *Revista de logică (publicaţie online).* [Interactiv] 01 Septembrie 2007. [Citat: 20 Iulie 2011.] http://egovbus.net/rdl/view_articol.php?vid=12. ISSN 2065-1449.

[4] **D. Bryan.** Exactness and clarity in a component-based specification language. [ed.] H. Kilov, W. Harvey. *Object-oriented behavioral specification.* Boston / Dordrecht / London : Kluwer Academic Publishers, 2006, pg. 1-15.

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

[6] **N. Guelfi, şi alţii.** Comparison of Object-Oriented Formal Methods. *DeVa ESPRIT Long Term Research Project No. 20072.* 1997. pg. 31-82. http://lglwww.epfl.ch/Team/NG/Publis/all_ps/97_coofm.ps.gz.

[7] **K. Hoffmann, H. Ehrig, T. Mossakowski.** *High-level nets with nets and rules as tokens.* [ed.] G. Ciardo, P. Darondeau. Applications and Theory of Petri Nets 2005, 26th International Conference, ICATPN 2005, Miami, USA, June 20-25, 2005, 2005, pg. 268-288. www.informatik.uni-bremen.de/~till/papers/high-level-nets.ps.

[8] **C. Paredes, J. Fiadeiro, J. Costa.** Architectural specifications: modeling and structuring behavior through rules. [ed.] H. Kilov, W. Harvey. *Object-oriented behavioral specification.* Boston / Dordrecht / London : Kluwer Academic Publishers, 2006, 14, pg. 221-240.

[9] **K. Slonneger, B. Kurtz.** *Formal syntax and semantics of programming languages: a laboratory based approach.* s.l. : Addison-Wesley Publishing Company, Inc., 1995. http://www.cs.uiowa.edu/~slonnegr/plf/Book/. ISBN 0-201-65697-3.

[10] **S. Yadav, şi alţii.** *Comparison of analysis techniques for information requirements.* Communications of the ACM, September 1988, Vol. 31, 9, pg. 1090-1097. http://www.mech.upatras.gr/~nikos/mis-ii/papers/yadav.pdf.

[11] **F. Păunescu, D. Goleşteanu.** *Sisteme cu prelucrare distribuită şi aplicaţiile lor.* Bucureşti : Editura Tehnică, 1993. p. 560. ISBN 973-31-0472-8.