STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LII**, Number 2, 2007

PROFESSOR FLORIAN MIRCEA BOIAN AT HIS SIXTIES

L. ŢÂMBULEA AND M. FRENŢIU

Professor Florian Mircea Boian graduated the Faculty of Mathematics-Mechanics 32 years ago, i.e. in 1975. He had the opportunity to be a graduate of the first promotion of the newly born section of Computer Science in our Faculty, in that year 1971.

The first year after graduation he worked in a computing center of the Carbochim-Cluj factory, and the following three years as an analyst in our University Computer Center, where he gained huge real experience in programming, and especially in operating systems. Since 1979 he became a member of department, were we are being colleagues for almost thirty years. During these years he stepped through all the didactic positions: assistant, lecturer, associate proffesor and, since 1997, full professor in Informatics Systems. As a teacher he realised various didactical and extradidactical tasks: admission commisions, license examination commisions, doctoral coordinations, the main promoter of the organisation of our computers network. He is a good colleague and colaborator, has published many scientifical papers and books together with some of his colleagues, and has participated to many scientifical grants and contracts of our department. He is an active participant at the huge evolution of Computer Science during the last thirty years.

He obtained his PhD degree in 1986, and since 2001 he is a PhD supervisor. He has an important contribution in many fields of Computer Science, and in the education of many generations of Computer scientists. Shortly, we mention:

- didactical activities at 28 disciplines (lectures, seminaries, laboratories);
- 12 published books, and 38 university manuals or culegeri de probleme;
- 78 published scientifical papers and 82 scientific conference presentations;
- director or member of many research grants, real industrial contracts;
- director of endowment grants, having as main beneficiaries our students and the young faculty members he cooperates with.

2000 *Mathematics Subject Classification.* 6803.

1998 *CR Categories and Descriptors.* A.0 [**General Literature**]: GENERAL – *Biographies/autobiographies*.

We must underline the quality of his work in our Department of Computer Science and University:

- His lectures are very appreciated by his students for the their clarity and contents. Consequently, he was elected by many generations as their graduation dean.
- He has introduced some new born disciplines in the Curriculum of the Computer Science Section at our University.
- He is coordinating our well organised network of computers from our laboratories.
- He has participated in many committees and juries of sesions of the students communications or competitions in Computer Science and Computer Engineering, or of school children in olympiads of Computer Science. He was a member of the scientific board of the international Computer Science contest for high school pupils in Central and Eastern Europe, May 1994.
- He is a member of various scientifical organisations: SSM (Romanian Society of Mathematical Sciences) since 1975; ACM (Association of Computing Machinery) since 1995; EUNIS (European Universities Informating Systems) since November 1995; founding member of ANIRO (Romanian National Association of Computer Scientists), since 1999; founding member of RoEduNet Romanian Education Network; member in the Editorial Advisory Board of: "International Journal of Intelligent Computing & Information Science", Ain Shams University, Cairo, Egypt; member in Editorial Board la "Carpatian Journal of Mathematics", Univ. Baia Mare; Executive Editor of the Journal "Studia Universitatis Babes-Bolyai", series Mathematica-Informatica, until 1996, co-president of the "Teaching Center Cluj", since 1998; member in the National Comission RoEduNet of the Romanian Ministry of National Education since 1997; evaluation expert for grants of the Romanian National Research Council of Higer Education, and of the World Bank, since 1996; member in the Editorial Board of "Studia Universitatis Babes-Bolyai", series Informatica.
- During the years 1994-2002 he has been the director of the newly born Communication Center of our University.
- Starting from these years he succeded to gather many young enthusiatic and talented students, who have succeded in their future carrier. Many of them received a PhD degree diploma and today are well appreciated teachers, or are working in software companies in Romania or abroad.

We, and all his colleagues do appreciate his hard work and effort to teach our students and for the development of our didactical and scientifical activity in Computer Science. We wish him a happy long life, full of achivements.

1. Scientific activity

1.1. **Printed books.**

(1) Informatica pentru elevi, ISBN 973-96096-2-7, Editura Microinformatica, Cluj, editiile 1-2 1992, editia 3 1993, (coautor), 212 pagini.
(2) Programare Pascal; programe ilustrative, probleme propuse pentru elevi si studenti, ISBN 973-96862-1-4, Editura Promedia-Plus, Cluj, 1995, (coautor), 228 pagini.
(3) Manualul începatorului în programare Pascal, ISBN 973-9215-04-1, Editura Microinformatica, Cluj, 1995, (coautor), 252 pagini.
(4) Sisteme de operare interactive, ISBN 973-96494-1-6, monografie, Editura LIBRIS, Cluj, 1994, 416 pagini.
(5) De la aritmetica la calculatoare, ISBN 973-97535-5-8, Editura Presa Universitara Clujeana, Cluj, 1996, 160 pagini.
(6) Programare distribuita în Internet: metode si aplicatii, ISBN 973-9215-60-2, monografie, Editura Albastra (grupul Microinformatica), editia 1 1998, editia 2 1999, editia 3 2000, 450 pagini.
(7) Bazele matematice ale calculatoarelor ISBN 973-8095-39-5, Editura Presa Universitara Clujeana, Cluj, 2000, (coordonator, autor în colab.cu Liana Bozga), 154 pagini.
(8) Programare concurenta pe platforme Unix, Windows, Java ISBN 973-650-072-1, monografie, Editura Albastra - grupul Microinformatica, Cluj, 2002. (coordonator, autor în colab.cu C Ferdean, R. Boian, R. Dragos), 420 pagini.
(9) Tehnologii fundamentale Java pentru aplicatii Web, ISBN 973-650-131-0, monografie, Editura Albastra - grupul Microinformatica, Cluj, 2004. (în colaborare cu R. Boian), 469 pagini.
(10) Informatica de baza, ISBN 973-610-340-4, Editura Presa Universitara Clujeana, Cluj, 2004. (în colab.cu M. Frentiu, L. Tâmbulea, I. Lazar), 226 pagini.
(11) Arhitectura calculatoarelor. Limbajul de asamblare 80x86, ISBN 973-651-037-2, Editura RISOPRINT, Cluj, 2005. (în colab.cu A.Vancea, D. Bufnea, A. Gog, A. Darabant, A. Sabau), 400 pagini.
(12) Sisteme de operare, ISBN 973-751-220-0 978-973-751-220-8, Editura RISOPRINT, Cluj, 2006. (în colab.cu A. Vancea, R. Boian, D. Bufnea, A. Sterca, C. Cobrzan, D. Cojocar), 350 pagini.

1.2. **Scientific papers.**

(1) Metoda reducerii si marcajelor pentru rezolvarea problemei transporturilor, în Studii si cercetari de calcul economic si cibernetica economica, Bucuresti, 1977, nr. 3, pp. 95-105.
(2) O metoda de rezolvare a problemei de transport dupa criteriul timp, în Studii si cercetari de calcul economic si cibernetica economica, Bucuresti, 1980, nr. 4, pp. 35-42.
(3) Pastrarea bibliotecilor de subprograme stiintifice sub sistemul de operare SIRIS, în Lucrarile primului simpozion national de teoria sistemelor, Craiova, 1980, vol II, pp. 260-264.
(4) Intocmirea orarelor în învatamânt cu ajutorul calculatorului, (în colab.cu I. Boieriu, A. Diaconu s.a.), în Informatica pentru conducere, progrese în informatica romaneasca, 1980, p. 127.
(5) Determinarea functiilor FIRST1, FOLLOW1 si EFF1 folosind metode booleene, în Lucrarile celui de-al IV-lea colocviu national de informatica, INFO-IASI, 1983, pp. 183-192.
(6) Automatizarea procesului de luare a deciziilor, (în colab.cu Z. Kasa, D. Oprean, L. Tâmbulea), în Supliment la revista economica, nr. 50, 1984, pp. 4-6.
(7) Syntactic Equivalence between Marked Graph Schemes and Loop-Exit Schemes, în "Babes-Bolyai" University, Faculty of Mathematics, Seminar on Computer Science, Preprint no. 4/1984, pp. 34-56. (MR87h.68008, p. 4489).
(8) INTADA/k: Subsets of Ada language for Teaching Computer Programming, în Proc. of the First Conference of Program Designers, Eotvos Lorand University, Budapest, 1985 pp. 25-30.
(9) Programarea calculatoarelor asistata de calculator, în Lucrarile seminarului de cercetari interdisciplinare, Universitatea din Cluj-Napoca, 1985, pp. 17-18.
(10) Sistemul pentru instruire INTADA; cerinte, definire, implementare, în Lucrarile celui de-al V-lea colocviu national de informatica, INFO-IASI, 1985, vol II, pp. 550-557.
(11) O metoda eficienta de implementare a deexecutiei, (în colab.cu A. Szekely), în Lucrarile celui de-al V-lea colocviu national de informatica, INFO-IASI, 1985, vol III, pp. 688-692.
(12) Reducing the Loop-Exit Schemes, în Mathematica, Cluj-Napoca, 28 (51), no. 1, 1986, pp. 1-7. (Zbl. Math. 614.68003, p.363).
(13) Towards a New Standard FORTRAN, (în colab.cu M. Frentiu, Z. Kasa, L. Tâmbulea), în "Babes-Bolyai" University, Faculty of Mathematics, Seminar on Computer Science, Preprint no. 6/1985, pp. 1-20.

(14) Simularea automatelor programabile, (în colab.cu M. Frentiu, Z. Kasa, L. Tâmbulea, L. Erdo, A. Szen), în Lucrarile simpozionului "Informatica si aplicatiile sale", Zilele academice Clujene, Cluj-Napoca, 1985, pp. 44-51.
(15) Compilare conversationala bidirectionala cu aplicatii la sistemul INTADA, în Lucrarile simpozionului "Informatica si aplicatiile sale", Zilele academice Clujene, Cluj-Napoca, 1985, pp. 52-58.
(16) Folosirea corecta a matricelor în programarea modulara, (în colab.cu M. Frentiu), în Lucrarile Seminarului "Didactica matematica", 1986, pp. 86-101.
(17) Loop-Exit Schemes and Grammars; Properties, Flowchartablies, în Studia Univ. "Babes-Bolyai", Mathematica, XXXI, 3, 1986, pp. 52-57. (Zbl. Math. 639.68017, p.353)
(18) A System for Program Writing and Debuging, (în colab.cu M. Frentiu, Z. Kasa), în "Babes-Bolyai" University, Faculty of Mathematics, Seminar on Computer Science, Preprint no. 5/1987, pp. 1-21.
(19) Analiza semantica bidirectionala, în Lucrarile celui de-al VI-lea colocviu national de informatica, INFO-IASI, 1987, vol I, pp. 83-88.
(20) Analiza sintactica bidirectionala, în Lucrarile sesiunii stiintifice a CCUB, Bucuresti, 1987, pp. 145-150.
(21) Sistem de programe pentru elaborarea statelor de functii, (în colab.cu M. Frentiu, Z. Kasa, L. Tâmbulea), în Lucrarile sesiunii stiintifice a CCUB, Bucuresti, 1987, pp. 438-443.
(22) FORTRAN can be Improved, (în colab.cu M. Frentiu, Z. Kasa, L. Tâmbulea), în Studia Univ. "Babes-Bolyai", Mathematica, XXXII, 3, 1987, pp. 15-16.
(23) Reversibile Execution with Loop-Exit Schemes, în Studia Univ. "Babes-Bolyai", Mathematica, XXXII, 3, 1987, pp. 29-36. (Zbl. Math. 638.68012, p. 321 si MR 1989, p. 119).
(24) Parallel Execution in Loop-Exit Schemes, (în colab.cu M. Frentiu, Z. Kasa), în "Babes-Bolyai" University, Faculty of Mathematics, Seminar on Computer Science, Preprint no. 9/1988, pp. 3-16. (Zbl. Math. 668.68023).
(25) Translatare bidirectionala între doua limbaje, în Lucrarile celui de-al doilea colocviu national de limbaje, logica, lingvistica matematica, Brasov, 1988, pp. 25-33. (Zbl. Math. 667.68022, p. 321).
(26) Folosirea calculatorului personal în predarea geometriei, (în colab.cu M. Frentiu, Z. Kasa), în Lucrarile Seminarului "Didactica matematica", 1987 1988, pp. 39-50.

(27) Elemente de programare în limbajul BASIC, (în colab.cu M. Frentiu, Z. Kasa), în Lucrarile Seminarului "Didactica matematica", 1987 1988, pp. 51-64.
(28) Sistem de fisiere bazat pe B-arbori, în Lucrarile celui de-al VII-lea colocviu national de informatica, INFO-IASI, 1989, pp. 33-40.
(29) Cautare rapida în B-arbori, în Lucrarile simpozionului "Informatica si aplicatiile sale", Zilele academice Clujene, Cluj-Napoca, 1989.
(30) Parallel Executable Sequences în Serial Programs, (în colab.cu M. Frentiu, Z. Kasa), în Studia Univ. "Babes-Bolyai", Mathematica, XXXIV, 3, 1989, pp. 3-16.
(31) Computer aided Geometry, (în colab.cu M. Frentiu, Z. Kasa), în "Babes-Bolyai" University, Faculty of Mathematics, Seminar on Computer Science, Preprint no. 9/1989 pp. 11-20.
(32) Efficiency în parallel evaluation of Arithmetic Expressions, (în colab.cu M. Frentiu, Z. Kasa), în "Babes-Bolyai" University, Faculty of Mathematics, Seminar on Complexity, Preprint no. 10/1989, pp. 1-14. (MR92m:65009, p. 7036 si Zbl. Math. 796.68110 p. 470).
(33) A Translator for Syntactic Bidirectional Analysis, în Analele Universitatii Bucuresti, Matematica-Informatica, XXXVIII, no. 2, 1989, pp. 14-20. (MR 1991 p. 125 si Zbl. Math. 739.68012 p. 371).
(34) Extended B-tree, în Studia Univ. "Babes-Bolyai", Mathematica, XXXVII, 3, 1992 pp. 13-20.
(35) Program testing for LOOP - EXIT Schemes, (în colab.cu M.Frentiu), în Studia Univ. "Babes-Bolyai", Mathematica, XXXVII, 3, 1992 pp. 21-30.
(36) Doua decenii de informatica universitara Clujeana, în Gazeta de Informatica 11/1992, pp. 7-8.
(37) An Implementation Scheme for PARBEGIN - PAREND Construction, (în colab.cu A. Vancea), în Studia Univ. "Babes-Bolyai", Mathematica, XXXVIII, 3, 1993, pp. 7-10.
(38) Bounds in very big arithmetic with very big bases, în "Babes-Bolyai" University, Faculty of Mathematics, Seminar on Computer Science, Preprint no. 4/1993 pp. 79-82 (Zbl. Math. 883.68043 pp. 483-484).
(39) The UNIX Environment at the "Babes-Bolyai" University, (în colab.cu A. Vancea, H.F. Pop, S. Iurian, M. Iurian), în Proc. of 2-nd International Conference ROSE, Bucuresti, 1994, pp. 145-149.
(40) Distributed Processing in Extended B-tree, (în colab.cu A. Vancea), în Studia Univ. "Babes-Bolyai", Mathematica, XXXIX, 3, 1994, pp. 25-34 (Zbl. Math. 857.68058 p.474).

(41) Teaching Parallel and Distributed Computing, (în colab.cu A. Vancea si H.F. Pop), în ROSE'95 Proceedings, Politechnical University Bucuresti, 1995, pp. 66-74.
(42) Folosirea calculatorului personal în predarea geometriei, (în colab.cu M.Frentiu si Z.Kasa) în lucrarile conferintei: "Informatizarea învatamântului", Balti, Republica Moldova, oct. 1995, pp. 66-71
(43) Comunicarea prin e-mail si iesirea în Internet, în lucrarile conferintei: Informatizarea învatamântului, Balti, Republica Moldova, oct. 1995, pp. 74-77
(44) An Efficient Topology for the "Babes-Bolyai" computer network: UBBNET, (în colab.cu C. Ciplea, G. Ciplea, L. Lazar, A. Moldovan), în Studia Univ. Babes-Bolyai, Mathematica, XXXXI, 1996.
(45) UBBNET: Computer Network at Babes-Bolyai University, (în colab.cu C. Ciplea, G. Ciplea, L. Lazar, A. Moldovan), în Proceedings of EUNIS97: European Cooperation în Higher Education Information Systems, Grenoble, France, September 1997.
http://www.lmcp.jussieu.fr/eunis/congres/second_en.html
(46) A Surveillance Authentication Protocol for UBBNET, to appear in INFORMATICA, Vilnius, Lithuania, (în colab.cu A. Vancea si M. Vancea).
(47) An Intranet Information System over UBBNET, (în colab.cu L. Lazar). Proceedings EUNIS'98, Prague, Czech Republic, ISBN 80-213-0420-0, 1998, pp. 159-162.
(48) Internet: a science, an information technology tool, a fashion, or all of these? în "Babes-Bolyai" University, Faculty of Mathematics, Seminar on Computer Science, Preprint no. 2/1997 pp. 31-34.
(49) Distributed Application based on Java, HTML and CGI at "Babes-Bolyai" University: Case Studies, în Advanced Educational Technology Conference Proceedings, ISBN 973-98726-8-9, Tg. Mures, 1998. Ed. Petru Maior University, 1999, pp 31-53
(50) On the Exactness of a Data Dependence Analysis Method, (în colab.cu A. Vancea), în Studia Univ. "Babes-Bolyai", Informatica, XLIII, 1, 1998, pp. 13-24.
(51) Half synchronized transition systems, (în colab.cu C. Ferdean), în Studia Univ. "Babes-Bolyai", Informatica, XLIV, 2, 1999, pp. 77-86.
(52) Nonsequential Program paradigms with Java Platform Applications, în Advanced Educational Technology Conference Proceedings, ISBN 973-8083-33-4, Cluj-Napoca, 2000. Ed. Petru Maior University, 2000, pp 5-43.

(53) Improving Distance Education in Computer Science at the Babes-Bolyai University (în colab.cu Z.Kasa si C.Ferdean) în Internet as a Vehicle for Teaching, Romanian Internet Learning Workshop, Miercurea-Ciuc, 11-20 Aug. 2001, pp. 125-129.
(54) Properties and implementation of the half-synchronized transition systems (în colab.cu C. Ferdean), în "Babes-Bolyai" University, Faculty of Mathematics, Seminar on Computer Science, Preprint, 2000, pp. 65-74.
(55) New Interaction Mechanisms between Java Distributed Objects, (în colab.cu C. Ferdean), în Studia Univ. "Babes-Bolyai", Informatica, XLV, 1, 2000, pp. 89-100.
(56) Properties and application framework for half-synchronized transition model, (în colab.cu C. Ferdean), in "International Journal of Computer & Information Sciences", Ain Shams University, Cairo, Egypt,.1, 1, Jul 2001 pp. 58-67.
(57) Advanced collaboration techniques between Java objects distributed on clusters (în colab.cu C. Ferdean), în "Advanced Environments, Tools, and Applications for Cluster Computing. Lecture Notes in Computer Science 2326, Springer Verlag, (International Workshop on Cluster Computing proceed., NATO Advanced Research Workshop, Mangalia, Romania, 2001), pp. 259-270.
(58) An Efficiency Comparison of Different Java Technologies, în Studia Univ. "Babes-Bolyai", Informatica, XLVI, 2, 2001, pp. 29-39.
(59) Enterprise JavaBeans, în Advanced Educational Technology Conference Proceedings, ISBN 973-8084-60-1, Târgu Mures, 2001. Ed. Petru Maior 2001, pp 59-128.
(60) Life In The JavaSpace (în colaborare cu C. Duda) Proceedings of the symposium "Zilele Academice Clujene" Cluj, 2002, pp. 25-34.
(61) Using EJB to develop enterprise applications (în colaborare. cu A. Sterca) Proceedings of the symposium "Zilele Academice Clujene" Cluj, 2002, pp. 41-51.
(62) Advanced Features in JDBC (Java Database Connectivity) Technology (în colaborare cu A. Câmpan) Proceedings of the symposium "Zilele Academice Clujene" Cluj, 2002, pp. 34-41.
(63) Filters, a new powerful feature of Java Servlets (în colaborare cu D. Bufnea) Proceedings of the symposium "Zilele Academice Clujene" Cluj, 2002, pp. 19-25.
(64) Challenges in today network routing (în colaborare cu D. Bufnea), Proceedings of the symposium "Zilele Academice Clujene" Cluj, 2003, pp. 19-25.

(65) Using the RMI with activation technology in distributed applications Proceedings of the symposium "Zilele Academice Clujene" Cluj, 2003, pp. 13-19.
(66) Active Queue Management for Multimedia Streams (în colab.cu A. Sterca, D. Bufnea, C. Cobârzan) In Proceedings of the Symposium Colocviul Academic Clujean de Informatica, 2004, pp. 117-122.
(67) Some Analisys of the Bbehavior for TCP Connections sharring a common subpath (în colab.cu D. Bufnea, A. Sterca, C. Cobârzan) în the 4-th International Conference on Applied Mathematics, Baia Mare, 2004, vol 20, no 2, pp. 149-154.
(68) RMI versus CORBA: A Message Transfer Speed Comparison (în colab.cu R. Boian în Studia Univ. "Babes-Bolyai", Informatica, XLIX, 1, 2004, pp. 83-91.
(69) Half-synchronized Mechanisms in Distributed Business Applications (în colab.cu R. Boian) "4-th International ROEDUNET Conference, Sovata, aprilie 2005.
(70) Continuations for remote objects control (în colab.cu E. Todoran, C. Melenti, N. Papaspyrou), în Studia Univ. "Babes-Bolyai", Informatica, L, 1, 2005, pp. 21-37.
(71) AMS: An Assignment Management System for Professors and Students (în colab.cu R. Boian, A. Vancea) Proceedings of the Symposium "Colocviul Academic Clujean de Informatica" Cluj, 2006, pp. 137-142.
(72) A Model for Efficient Session Object Management in Web Applications (în colab.cu D. Bufnea, A. Vancea, A. Sterca, D. Cojocar, R. Boian) Proceedings of the Symposium "Colocviul Academic Clujean de Informatica" Cluj, 2006, pp. 131-136.
(73) Shared Bottleck Detection from Receiver Point of Viev (în colab.cu D. Bufnea, A. Vancea) Proceedings of the Symposium "Colocviul Academic Clujean de Informatica" Cluj, 2006, pp. 125-130.
(74) Supporting multimedia streaming applications inside the network (în colab.cu A. Sterca, D. Bufnea, C. Cobârzan) în Studia Univ. "Babes-Bolyai", Informatica, LI, 2006, pp. 37-48.
(75) Evaluating Dynamic Client-Driven Adaptation Decision Support in Multimedia Proxy-Caches(în colab.cu A. Sterca, D. Bufnea, C. Cobârzan), KEPT 2007 Knowledge Engineering Principles and Techniques, Cluj University Press 2007 ISBN978-973-610-556-2, pp. 298-306.
(76) Some Formal Approaches for Dynamic Life Session Management (în colab.cu D. Bufnea, A. Vancea, A. Sterca, D. Cojocar, R. Boian), KEPT 2007 Knowledge Engineering Principles and Techniques, Cluj University Press 2007 ISBN978-973-610-556-2, pp. 227-235.

(77) Distance Learning and Supporting Tools at Babes-Bolyai University (în colab.cu R. Boian, A. Vancea, H.F.Pop), IEEII - Informatics Education inEurope, 29-30 November, 2007, Thesaloniki, Greece (accepted - to appear).

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Department of Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

E-mail address: leon,mfrentiu@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LII**, Number 2, 2007

WEB SOURCE CODE POST-PROCESSING: A NEW APPROACH BASED ON CLASSIC MODELS AND METHODS

FLORIAN BOIAN

Abstract. The majority of today's technologies for distributed and web application development adopted programming languages from the C family. To increase the application security, popular such languages like Java, C#, and PHP have been designed without features for direct memory manipulation such as pointers, pointer arithmetic, or memory buffer casting to primitive types similar to the C union construct. The lack of these features makes much easier the implementation of pre-processing and post-processing models for source code simplification, verification and testing. In this paper, a formal approach post-processing mechanism for the web languages is describe. At the moment, the web developing languages has not use the post-processing techniques.

Keywords: source to source transformation, abstract programming schemes, formal methods of source code representation, web applications.

1. Introduction

Source to source code transformation is a wide-spread research direction [1, 11] that studies methods of source code improvement and optimization through automatic manipulation. Source code optimization through refactoring [1, 6] and elimination of redundant control structures are such source to source methods. For instance, sequences like the one in the left side of the table below, can be automatically transformed in the equivalent form on the right, provided that the a and t code segments are independently.

IF t THEN
  a;
  b;
ELSE
  a;
  c;
ENDIF
	a;
	
	IF t THEN
	  b;
	ELSE
	  c;
	ENDIF

This optimization method can also be applied to sequences of assignments as shown in the table below. In our example the transformation is conditioned by the independence between e and b*c [15].

`a = b * c + d;`
`x = b * c / e;`
	`temp = b * c;`
	`a = temp + d;`
	`x = temp / e;`

More recent research directions are focused on automatic parallelization of source code [8]. For example, the code sequence in the left side of the table below can be transformed in the sequence on the right, provided that a(i) and b(i) are independent except for the presence of variable i. In the new form, the two FOR loops in the sequence on the right can be parallelized.

`FOR i ... DO`
`  a(i)`
`ENDFOR;`
 	
`FOR i ... DO`
`  b(i)`
`ENDFOR`
	`FOR i1 ... DO`
	`  a(i1)`
	`ENDFOR;`
	
	`FOR i2 ... DO`
	`  b(i2)`
	`ENDFOR`

The LOOP-EXIT and LOOP-EXIT-CYCLE [13, 2, 5] schemes revolutionized the automatic source to source transformation techniques. Source code structured using these schemes can be represented easier with formal abstract constructs and thus it is easier to process automatically. These schemes will be addressed in more detail in the following sections.

The transformations presented above are implemented in most of today's compilers.

Today's technologies for development of distributed and web application are based on C-like programming languages without the C features that usually make the code vulnerable. Thus, popular programming languages such as Java [7, 12], C# [14], and PHP [10] have been designed without features for direct memory manipulation such as pointers, pointer arithmetic, or memory buffer casting to primitive types similar to the C union construct. Under these circumstances, many of the problems faced by the source to source transformation methods [1, 11] are no longer possible. Consequently, it becomes much simpler to implement and apply source to source translation for code improvement, optimization, validation, and testing.

In the following sections we will present and discuss source to source transformations using as abstract concept the LOOP-EXIT schemes. The examples will be written in one of the C-like programming languages presented above. The scope of these transformations is to process server side code and detect at

an abstract level incoherent code and suggest improvements to the developer. These transformations can be also used to provide information to support server side code execution and logging.

2. LOOP-EXIT Schemes, Branches, and Sections

For the purpose of this paper, we consider the LOOP-EXIT schemes as they are defined in [5]. We denote by A the set of the assignment symbols, and with T the set of the test symbols.

To each LOOP-EXIT scheme S, a language L(S) can be associated. The context-free grammar of scheme S is:

$$G(S) = (N, \Sigma, P, \Delta)$$

Here N is the set of non-terminals, Δ is the axiom of grammar G(S). For each IF_k from S there is a nonterminal I_k in N. For each $LOOP_k$ there are two nonterminals L_k and B_k. Σ is the terminal symbol set. Each symbol from A appears in Σ. For each symbol t from T, the symbols t+ (for true) and t- (for false), appear in Σ. The product rules are detailed in [5,6,8]. In this section we will avoid presenting how the products are formal constructed, but rather will give a practical example in the next section.

In a LOOP-EXIT scheme S, we can define special complete execution paths. A section from S is a maximal sequence in Σ^* where the order of the symbols is the same as in the static text of S.

More exactly, a word z from Σ^* is a section iff exists a word w from L(S) so that :

(1) w = z, or
(2) w = xz and the last symbol from x appears in the text of S after the first symbol from z, or
(3) w = zy and the last symbol from z appears in the text of S after the first symbol from y, or
(4) w = xzy and conditions 2 and 3 are true.

We denote by SECT(S) the set of the sections. A branch in S is a word in SEC(S) so that only conditions 1 or 2 above are true.

We denote by BRAN(S) the set of the branches from S. The sets SECT(S) and BRAN(S) can be generated as regular and finite language with products constructed starting with G(S). The complete construction algorithm is presented in [8].

3. A post-processing example of branch calculation

The following PHP function replaces in the input string special HTML characters and non-ASCII characters with their corresponding HTML codes.

The function returns the ASCII string resulted from processing. The arrays of special character and their corresponding HTML codes are:

$$\$sa = array(" "," & "," \"" ," < " ," > " ," | " , ...);$$

$$\$sc = array("\ ","\&","\"","\<","\>","\&brvar;", ...);$$

Translating function receives as arguments the input string, and the two arrays above. The PHP code of the function is:

```
function replace($s, $si, $so) {
  if (!isset($s))
    return "";
  for($i = strlen($s)-1; $i>=0; $i--) {
    for($j=0; $j<count($si); $j++) {
      $p = strrpos($s, $si[$j]);
      if ($p === false)
        continue;
      if ($p == $i) {
        $s = substr($s,0,$i).$so[$j].substr($s, $i+strlen($si[$j]));
        break;
      }
    }
  }
  return $s;
}
```

The transformation is executed using the list of variables and constants that appear in the source code.

$$V = \{\$s, \$si, \$so, "", \$i, \$j, \$p, false\}$$

The assignments statements will be grouped in set A and will be denoted by a1, a2, ..., an. The test statements will form the set T and will be referred to as t1, t2, ..., tn.

The PHP code above will be first translated in abstract code using LOOP-EXIT-CYCLE constructs [13,5]. The result of the transformation is in the following table, the left part.

In the abstract code above, we use EXIT for exiting the inner-most cycle, and CYCLE for jumping to the beginning of the inner-most cycle.

According to theory, a LOOP-EXIT-CYCLE scheme can be automatically transformed in LOOP-EXIT scheme by adding additional LOOP-ENDLOOP statements. The code resulting after these transformations is presented in the following table, the right part. To simplify the code, we replaced the real statements with abstract ones.

```
LOOP1
  IF t1($s) THEN
    a1("")
    EXIT
  ENDIF
  a2($i, $s)
  LOOP2
    IF t2($i) THEN
      EXIT
    ENDIF
    a3($j)
    LOOP3

      IF t3($j, $si) THEN
        EXIT
      ENDIF
      a4($p,$s,$si,$j)
      IF t4($p,false) THEN
        CYCLE
      ENDIF
      IF t5($p,$i) THEN
        a5($s,$i,$so,$si,$j)
        EXIT
      ENDIF
      a6($j)

    ENDLOOP3
    a7($i)
  ENDLOOP2
  a8($s)
  EXIT
ENDLOOP1
```

```
LOOP1
  IF1 t1 THEN1
    a1
    EXIT1
  ENDIF1
  a2
  LOOP2
    IF2 t2 THEN2
      EXIT2
    ENDIF2
    a3
    LOOP3
      LOOP4
        IF3 t3 THEN3
          EXIT3
        ENDIF3
        a4
        IF4 t4 THEN4
          EXIT4
        ENDIF4
        IF5 t5 THEN5
          a5
          EXIT3
        ENDIF5
        a6
      ENDLOOP4
    ENDLOOP3
    a7
  ENDLOOP2
  a8
  EXIT1
ENDLOOP1
```

According to [5], the productions of the grammar associated to scheme S above are:

Δ $->$ L1
L1 -> I1 a2 L2 a8 L1 | t1+ | a2 B2 a8
B1 -> I1 a2 L2 a8 B1 | ε
I1 -> t1-
L2 -> I2 a3 L3 a7 L2 | t2+
B2 -> I2 a3 L3 a7 B2 | ε
I2 -> t2-
L3 -> L4 L3 | B4 t3+ | I3a4I4 t5+ a5
B3 -> L4 B3 | ε

L4 -> I3 a4 I4 I5 a6 L4 | I3 a4 t4+
I3 -> t3-
I4 -> t4-
I5 -> t5-

The language generated by this grammar is:

L(S) = t1+a1 | t1-a2(t2+ | t2-a3(t3+ |(a4t4+ | (a4t4-t5+a5 | a4t4-t5-)* a7)*)a8)*

The branch set of scheme S is:

BRAN(S) = {t1+a1, t1-a2t2+a8, t1-a2t2-a3t3+a7, t1-a2t2-a3t3-a4t4+, t1-a2t2-a3t3-a4t4-t5+a5, t1-a2t2-a3t3-a4t4-t5-a6}

The section set of the scheme S is:

SECT(S) = BRAN(S) ∪ {a2t2+a8, a2t2-a3t3+a7, a2t2-a3t3-a4t4+, a2t2-a3t3-a4t4-t5+a5, a2t2-a3t3-a4t4-t5-a6, t3+a7, t3-a4t4+, t3-a4t4-t5+a5, t3-a4t4-t5-a6}

4. Conclusions

The example presented in the section above, shows that post-processing is easily to implement in programming languages lacking direct memory manipulation features such as PHP. Server side applications are usually written in Java, C#, or PHP all of which make post-processing simple. Further optimizations of post-processing can be done on syntactically correct source code. For instance, the post-processing can skip verifying matching parentheses or brackets, or checking for correct statement closing with ";", thus reducing processing work. Post-processing is also simplified by the existence of reserved keywords and variable declaration.

What are the benefits of post-processing? In our view, post-processing in the sense presented above, can assist the developer, in the following ways:

- A tool that can provide the code flow branches can aid the developer visualize the code and place proper logging messages in relevant places.
- According to [9], it is possible to perform an automatic analysis of branches to detect un-initialized variables. Java and C# are able to signal such cases at compile time, while PHP signals such cases only at runtime.
- The branch analysis can help the developer take a series of code refactoring decisions, that remove redundant operations and simplify the flow.
- The representation of the real code into an equivalent abstract code can give the developer a different perspective of the code, which can lead to positive changes in the code.

- Server applications are inherently difficult to debug. Post-processing can help restructure the code in manners making it clearer and easier to analyze, leading to less runtime errors.
- Large scale web applications (such as the one presented in [3]) raise problems more complex than smaller applications. Branch analysis is mandatory to reduce the problems experienced by the large number of users accessing the application, and avoid high maintenance costs.
- Web applications working under high load must be optimize the usage of the resources. The section extraction done by post-processing assist the developer to organize the information to be saved in the HTTP session objects [4].

As a future work, we will implement a post-processing mechanism, for the web languages as PHP, C#, and Java.

References

[1] **Arsac J. J.,** *Syntactic Source to Source Transformation and Program Manipulation.* Comm. ACM, 22, no 1, 1979, pp. 43-53.

[2] **Baker B.S., Kosaraju S.R**., *A Comparision of Multilevel Break and Next Statements.* Journal ACM, 26, no 3, 1979, pp 555-566.

[3] **Boian F.M. et.al.,** *Distance Learning and Supporting Tools at Babes-Bolyai University* IEEII - Informatics Education inEurope, 29-30 November 2007, Thesaloniki, Greece (accepted - to appear).

[4] **Boian F.M. et.al.,** *Some Formal Approaches for Dynamic Life Session Management* KEPT 2007 Knowledge Engineering Principles and Techniques, Cluj University Press 2007 ISBN978-973-610-556-2, pp. 227-235.

[5] **Boian F.M.,** *Loop - Exit Schemes and Grammars; Properties, Flowchartablies.* Studia UBB, Mathematica, XXXI, 3, 1986, pp. 52-57.

[6] **Boian F.M.,** *Reducing the Loop - Exit Schemes.* Mathematica (Cluj) 28(51), no 1, 1986, pp. 1-7.

[7] **Boian F.M., Boian R.F.,** *Tehnologii fundamentale Java pentru aplicaii Web*, Editura Albastr - grupul Microinformatica, Cluj, 2004.

[8] **Boian F.M., Frentiu M**. **Kasa Z.,** *Parallel Execution in Loop - Exit Schemes.* UBB, Faculty of Mathematics and Physics, Research Seminaries, Seminar on Computer Science, Preprint no. 9, 1988, pp. 3-16.

[9] **Boian F.M., Frentiu M**., *Program Testing in Loop - Exit Schemes.* Studia UBB, Mathematica, XXXVII, 3, 1992, pp. 21-30.

[10] **Converse T. et. al.,** *PHP5 and MySQL Bible.* Wiley, 2004.

[11] **Greibach S.** *The Theory of Program Structures: Scheemes, Semantics, Verification.* Springer Verlag, LNCS 1975, 36,1.

[12] **Jendrock E. et.al.,** *The Java^{TM} EE 5 Tutorial, Third Edition: For Sun Java System Application Server Platform Edition* Addison Wesley, 2006.

[13] **Moss C.D.S.,** *Structured Programming With LOOP Statements.* SIGPLAN Not. 15, no 1, 1980, pp. 86-94.

[14] **Turtschi et.al.,** *C# .NET Web Developer's Guide.* Syngress, 2002.

[15] **Vancea A., Boian F.M.,** *On the Exactness of a Data Dependence Analysis Method.* Studia UBB, Mathematica, XLIII, 1, 1998, pp. 13-24.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Department of Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

E-mail address: florin@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LII**, Number 2, 2007

COMPONENT-BASED ANT SYSTEM FOR A BIOBJECTIVE ASSIGNMENT PROBLEM

CAMELIA-M. PINTEA AND ANDREEA VESCAN

Abstract. The paper proposes a component-based approach for a particular biobjective assignment problem: the Airport Gate Assignment Problem (AGAP). ACS-QAP [2] is the starting point for the proposed ACS model for solving an over-constrained version of AGAP, seeking feasible flight-to-gate assignments so that total passenger connection time, as proxied by walking distances, is minimized.

1. Introduction

Since the late 90's Component Based Development (CBD) is a very active area of research and development. CBSE covers both component development and system development with components [6]. There is a slight difference in the requirements and business ideas in the two cases and different approaches are necessary. Of course, when developing components, other components can be (and often must be) incorporated and the main emphasis is on reusability. Development using components is focused on the identification of reusable entities and relations between them, starting from the system requirements.

The complex biobjective problem modeled for an ant system algorithm using components is the over-constrained *Airport Gate Assignment Problem (AGAP)*. The problem has two objectives. The first one to minimize the number of flights assigned to the apron, when the number of flights exceeds the number of gates. The second objective is to minimize the total distance walk of the passengers in the airport.

The preliminary sections of the paper show the specifications of $AGAP$ and the techniques used to solve the specified problem. The main sections of the paper show the component-based solution of the ant system algorithm for solving $AGAP$ including the control flow and data flow.

Received by the editors: 30.08.2007.

2000 *Mathematics Subject Classification.* 68N30, 68T20.

1998 *CR Categories and Descriptors.* code I.6.5 [**Simulation and modeling**]: Subtopic – *Model Development*; code I.2.8 [**Artificial intelligence**]: Subtopic – *Problem Solving, Control Methods, and Search* .

2. The Biobjective Assignment Problem

A particular case of a *Quadratic Assignment Problem (QAP)* it is considered: *the Airport Gate Assignment Problem (AGAP).*

There were several attempts to solve *AGAP*. We are mentioning a Tabu Search metaheuristic by Xu and Bailey [17]. Another algorithm for solving *AGAP* was proposed by Ding et al. [7] with a greedy algorithm minimizing ungated flights while providing initial feasible solutions followed by a new neighborhood search technique.

The gate assignment problem has the objective of minimizing distance costs of the over constrained gate assignment problem, minimizing the number of ungated aircrafts and the total walking distances.

We consider the notations as in [7]:
N: set of flights arriving at the airport and/or departing from the airport;
M: set of gates available at the airport;
n: total number of flights, i.e., $|N|$, where $|N|$ denotes the cardinality of N;
m: total number of gates, i.e., $|M|$;
a_i: arrival time of flight i;
d_i: departure time of flight i;
w_{kl}: walking distance for passengers between the gates k and l;
f_{ij}: the number of passengers transferring between two flights i and j;

Two dummy gates are used: gate 0 the entrance or exit of the airport and gate $m+1$ the apron where flights arrive at when no gates are available. $y_{i,k}$ denotes that flight i is assigned to gate k if $y_{i,k}=1$ and otherwise $y_{i,k}=0$, where $(0<k<m+1)$.

$w_{k,0}$ is the walking distance between gate k and the airport entrance or exit. $f_{0,i}$ is the number of originating departure passengers of flight i. $f_{i,0}$ is the number of the disembarking arrival passengers of flight i. The walking distance between the apron and gate k is $w_{m+1,k}$.

The mathematical model of the biobjective problem: *The Airport Gate Assignment Problem* is following.

1. Minimize the number of flights assigned to the apron:

$$\sum_{i=1}^{n} y_{i,m+1} \rightarrow min,$$

2. Minimize the total walking distance:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m+1}\sum_{l=1}^{m+1} f_{i,j}w_{k,l}y_{i,k}y_{j,l}+$$

$$+\sum_{i=1}^{n}\sum_{l=1}^{m+1} f_{0,i}w_{0,l} + \sum_{i=1}^{n}\sum_{l=1}^{m+1} f_{i,0}w_{l,0} \to min,$$

The constraints of $AGAP$ are following.

$$\sum_{k=1}^{m+1} y_{i,k} = 1(\forall i, 1 \le i \le n) \tag{1}$$

These constraints ensure that every flight must be assigned to one and only one gate or assigned to the apron.

$$a_i < d_i(\forall i, 1 \le i \le n) \tag{2}$$

Constraint (2) specifies that each flight's departure time is later than its arrival time.

$$y_{i,k}y_{j,k}(d_j - a_i)(d_i - a_j) \le 0(\forall i, j, 1 \le i, j \le n, k \ne m+1) \tag{3}$$

Constraint (3) says that two flights schedule cannot overlap if they are assigned to the same gate.

$$y_{i,k} \in \{0,1\}(\forall i, 1 \le i \le n, \forall k, 1 \le k \le m+1) \tag{4}$$

The condition (4) disallows any two flights to be scheduled to the same gate simultaneously except if they are scheduled to the apron.

2.1. **The over-constrained approach.** For the over-constrained $AGAP$, the first step is to minimize the number of flights that need be assigned to the apron. The minimal number of flights can be computed by a greedy algorithm described in [7].

First of all the flights are sorted by the departure time and after that flights are assigned one by one to the gates. A flight is assigned to an available gate with latest departure time. If there are no gates available, the flight will be assigned to the apron.

The solution of the greedy algorithm is the optimal number of flights that can be scheduled in gates and it is used to provide initial feasible solutions for the *ACS*-based algorithm.

3. Ant System for Airport Gate Assignment Problem

The algorithm proposed is an improved version of *Ant Colony System (ACS)* for *Quadratic Assignment Problem (QAP)* [2]. The new algorithm is called *Reinforcing Ant System-QAP (RAS-QAP)* where the trail intensity is locally updated using the inner rule [14] (*local_update_pheromone_trails()*).

The problem of the *Airport Gate Assignment* is about finding the feasible flight-to-gate assignments so that total passenger connection time is minimized. The function we have to minimize is using the distances from check-in

to gates in the case of embarking or originating passengers, from gates to check-out in the case of disembarking or destination passengers and from gate to gate in the case of transfer or connecting passengers.

When the number of aircraft exceeds the number of available gates, in the over-constrained case, the distance from the apron to the terminal for aircraft assigned to these areas is also considered.

First are computed the distance potentials and flow potentials as in [17, 7]. Each edge (i, j), at moment t, is labeled by a trail intensity $\tau_{ij}(t)$.

Algorithm 1 RAS-QAP

```
assign_initial_pheromone_levels();
compute_distance_potentials();
compute_flow_potentials();
place_ants_on_locations();
for (t = 1) to t_max do
  for (k = 0) to num_ants − 1 do
    build_solution_for_ant(k);
    local_update_pheromone_trails();
    compute_cost_solution_for_ant(k) based on constraints (1)-(4);
    if ants[k].cost_solution < Best_solution.cost_solution then
      Best_solution=ants[k].cost_solution;
      Best_solution_t=t;
    end if
  end for
  update_pheromone_trails();
  write_experimental_results(t);
end for
```

Initially the ants are randomly placed in the graph nodes. At each iteration an ant moves to a new node. When an ant decides which node is the next move it does so with a probability based on the distance to that node and the amount of trail intensity on the connecting edge. Evaporation takes place, at each step, to stop the intensity trails increasing unbounded.

Two tabu lists are used. The first tabu list stores the ants visited locations, so that they never visit them again. The second tabu list stores the activities that have been mapped to the visited locations.

To favour the selection of an edge that has a high pheromone value and high visibility value, a probability is considered. Compute cost solution for ant k and update the best solution. The global update rule is applied to the edges belonging to the *best tour-solution*.

The solution of the algorithm is the tour with the minimal cost. A solution is a vector with the potentially best distances, called distance potentials, and potentially best flows, called flow potentials. The algorithm runs for a given number of iteration t_{max}.

4. Component elements

A system can be designed and implemented by assembling components, customizing or extending them as needed; and publishing components in a form that can be applied to design and construct others, based purely on interface specifications.

One of the most popular definitions of a component was offered by a working group at ECOOP (European Conference on Object-Oriented Programming).

Definition 1. *A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and it is subject to composition by third parties.* [15]

Clemens Szyperski and David Messerschmitt [15] give the following five criteria for what a software component shall be to fulfill the definition: multiple-use; non-context-specific; composable with other components; encapsulated i.e., non - investigable through its interfaces and a unit of independent deployment and versioning.

We must first establish our entities involved in the component system definition before describing our component-based approach for modeling AGAP.

Considering X a component over the set A of attributes, we will use the following notations: $inports(X) \in A$ - represents the set of input ports (attributes) of the component X; $outports(X) \in A$ - represents the set of output ports (attributes) of the component X; $attributes(X) \in A$ - represents the set of attributes of the component X.

We can view components from a different perspective [10] as simple components and compound components with the following characteristics:

- **Simple Component** - over A is a 5-tuple **SC** of the form

  $(inports, outports, attributes, function, \prec_{SC}), where$:

  - $inports$ is a n-tuple $(in_1, ..., in_n)$ of attributes;
  - $outports$ is a m-tuple $(out_1, ..., out_m)$ of attributes and $(out_1, ..., out_m) \notin inports(SC)$;
  - $function$ is an n-ary function $Type(in_1) \times Type(in_2) \times ... \times Type(in_n) \rightarrow Type(out_1) \times Type(out_2) \times ... \times Type(out_m)$.

– *attributes* is defined to be the set of attributes consisting of the inports and the outports;
– the binary relation
$\prec_{SC} \subseteq (inports(SC) \times outports(SC)) \times outports(SC)$.

- **Compound component** - over A is a group of connected components, in which the output of a component is used as input by another component from this group.

A graphical representation of our view of components is given in Figure 1.

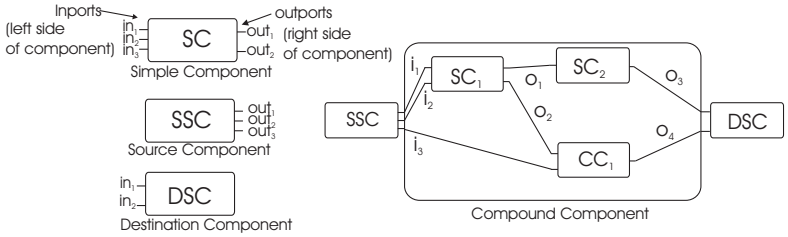


FIGURE 1. Components graphical representation. The compound component contains two simple components SC_1 and SC_2 and one compound component CC_1

Two particular components are the source component [1] and the destination component [2]. The source component represents the "read" component and the destination component represents the "write" component, components that should exists in any software system. In our approach every assembly system (and subsystem) has only one source component and only one destination component.

4.1. **Component construction and execution elements.** The wiring of components in order to construct a component-based system is made using a connection between the output of a component and the input of another component.

A **connection K** is made of an origin - output of a component, and a destination - input of another component:

$$K = (origin, destination),$$

[1]source component has no inports and generates data provided as outports in order to be processed by other components

[2]destination component has no outports and receives data from its inports and usually displays it, but it does not produce any output

where *origin* is an *outport* of a component; *destination* is an *inport* of a component.

The composition result is also a component, a compose component using [10] notation. The resulted system is called **BlackBox** and is specified as follows:

$$BlackBox = (\{in\}, \{out\}, \{component\}, \{connection\}),$$

where *in* represent the inputs for the blackbox; *out* represent the output of the blackbox; *component* represent the components involved in the composition and *connection* represent all the connections between the involved components.

The execution of the BlackBox component is composed of sequences of the form:

$$(Op_0, C_0), (Op_1, C_1), (Op_2, C_2), ...$$

where for each $i \geq 0$, Op_i is a subset of possible operations and C_i is a subset of components ready for execution.

The possible operations are:

- propagation - this rule moves values that have been generated by a component along connections from the component's outport to other components;
- evaluation - the component function is evaluated and the result is passed to the output of the component.

State of execution. At a given time of execution, the state is presented as follows:

$$State = (\{operation\}, \{componentForEval\}),$$

where $operation = \{C->, C=\}$;

- $C->$ - propagation operation from component C;
- $C=$ - evaluation operation of component C.

componentForEval - a component ready for evaluation.

If at a given time, both types of operation can be performed, the propagation operation is chosen. Between many evaluation operations, one component is chosen randomly.

4.2. **Component assembly construction process.** A top-down approach is used when reasoning about the way to solve a problem (from the system requirements develop the needed modules to accomplish the requirements of the system under development) and a bottom-up approach when assembling the pieces in order to build the desired final system [16].

The components composition is accomplished using a bottom-up approach: starting from a given set of components (stored in a repository) there are two main steps to obtain the final system:

(1) newly obtained components (if necessary) by assembling given components (simple components and/or compound components);
(2) compose the final system from the new set of available components.

Reasoning in a top-down approach we refer to the second step (from the bottom-up approach) as the first hierarchical level of the final system(s) and to the first step (which may contain many inside steps to develop new components) as intermediary hierarchical levels of the system.

5. Component-based Ant System for Airport Gate Assignment Problem

This section presents the architecture of the component-based approach for the AGAP, the control flow and the data flow model. At the end of this section we show how the computation steps are successively executed.

5.1. **AGAP architecture.** In section 3 we have presented the ant system solution for the Airport Gate Assignment Problem. Based on the pseudocode algorithm we can describe the solution using components as in Figure 2 as a first level of design: initialization, computation and printing the obtained results. The next two levels (decompositions) are also presented.

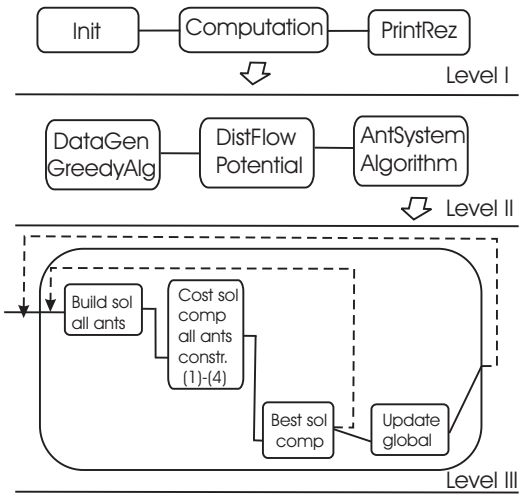


Figure 2. Architectural levels

The second level contains the data generation for the greedy algorithm, the computation of the distance flow potentials and the ant system algorithm.

The third level contains a more detail view of the ant system algorithm computation: build solution (for all ants), then cost solution computation (for

all ants) based on the constraints, best solution computation (for all ants) and the update global rule computation.

All these computations are performed for t_{max} steps/times. The build solution computation, cost solution and best solution computation are all computed for all ants, thus the loop is executed until the number ants *num_ants* (see algorithm 1 for details) is reached.

5.2. **Control flow AGAP model.** Figure 3 shows the overview of the system and the control flow.

The

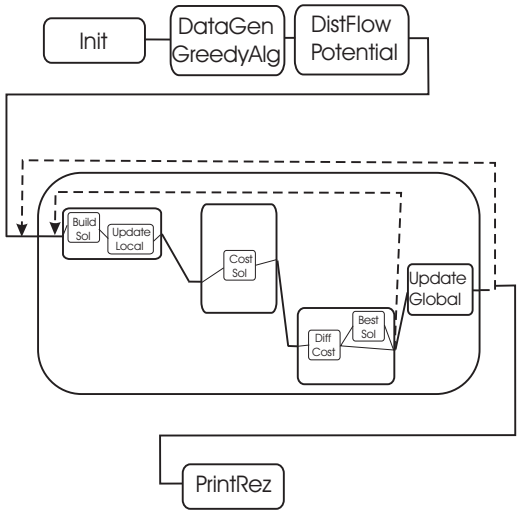


FIGURE 3. Control flow component-based RAS-QAP system

5.3. **Data flow AGAP model.** Figure 4 shows the overview of the system with all the components from all the architecture levels - the data flow. Component *DataGenGreedyAlg* corresponds to the first statement in algorithm 1, component *DistFlowPotential* to the second, third and fourth statements and component *AntSystemAlgorithm* to the rest of the statements.

FIGURE 4. Data flow component-based RAS-QAP system

The data transmitted from the init to the greedy algorithm is that from the problem description: N - set of flights arriving at (and/or departing from) the airport, M - set of gates available at the airport, n - total number of flights and m - total number of gates.

The greedy algorithm gives the optimal number of flights that can be scheduled in gates.

The $DistFlowPotential$ computes the sums based on the number of passengers f_{ij} and the walking distance from gate k to gate l, w_{kl}.

The $AntSystem$ computes L^+, thus minimizing the total walking distance.

5.4. **General and internal computation steps.** The steps of the computation of the ant colony component model for $AGAP$ are described successively.

We denote each component from the architecture (see figure 2 and figure 3 for details) with the following acronyms in order to be more easier to read: $Init$ component with I; $DataGenGreedyAlg$ with $DGGA$; $DistFlowPotential$ with DFP; $BuildSol$ and $UpdateLocal$ with $BSUL$; $CostSol$ with CS; $DiffCost$ and $BestSol$ with $DCBS$; $UpdateGlobal$ with UG and $PrintRez$ with PR.

General computation steps for the representation from figure 4:

- $state_0 = (\{I \equiv\}, \{I\})$;

-N, M, n, m receives the initial values.

- $state_1 = (\{I \rightarrow\}, \{\})$;

-data is propagated through the connections to the $DGGA$ component: N, M, n, m.

- $state_2 = (\{DGGA \equiv\}, \{DGGA\})$;

-the data are generated in the corresponding intervals for a_i, d_i, w_{kl} and f_{ij}. -the greedy algorithm minimizes the number of flights assigned to the apron and returns the $flights$ array.

- $state_3 = (\{DGGA \rightarrow\}, \{\})$;

-$DGGA \rightarrow$: data is propagated through the connections to the DFP component.

- $state_4 = (\{DFP \equiv\}, \{DFP\})$;

-$DFP \equiv$: computes the sums based on the number of passengers f_{ij} and the walking distance w_{kl} from the gate k to gate l.

- $state_5 = (\{DFP \rightarrow\}, \{\})$;

-$DFP \rightarrow$ data is propagated through the connection to the input of the ASA component.

- $state_6 = (\{ASA \equiv\}, \{ASA\})$;

-$ASA \equiv$ minimizes the total walking distance.

- $state_7 = (\{ASA \rightarrow\}, \{\})$;

-$ASA \rightarrow$ data is propagated through the connection to input of the PR component.

- $state_8 = (\{PR \equiv\}, \{PR\})$;

-$PR \equiv$ prints the obtained result.

- $state_9 = (\{\}, \{\})$.

-There are no more possibilities of applying either propagation or evaluation.

-The execution of the components involved in the system is finished.

The execution states for the representation from figure 3 (only Level III) are described in the following. The execution of the components starts from $state_6$ from the general computation from above.

- $state_{6_1} = (\{BSUL \equiv\}, \{BSUL\})$; $state_{6_{1'}} = (\{BSUL \rightarrow\}, \{\})$;
- $state_{6_2} = (\{CS \equiv\}, \{CS\})$; $state_{6_{2'}} = (\{CS \rightarrow\}, \{\})$;
- $state_{6_3} = (\{DCBS \equiv\}, \{DCBS\})$; $state_{6_{3'}} = (\{DCBS \rightarrow\}, \{\})$;

-if the previous steps were not executed for each ant then the state 6_1 is following, else state 6_4.

- $state_{6_4} = (\{UG \equiv\}, \{UG\})$; $state_{6_{4'}} = (\{UG \rightarrow\}, \{\})$.

-if the maximum t_{max} number of iterations are not reached then the state 6 is following, else state 7.

6. Conclusions

Ant algorithms are based on the real world phenomena that ants are able to find their way to a food source and back to their nest, using the shortest route. A component based Ant System for the Airport Gate Assignment Problem is introduced. The way of using the components is shown: the control flow and the data flow. The model execution steps are also illustrated.

References

[1] O. Babic, D. Teodorovic, V. Tosic, *Aircraft stand assignment to minimize walking*, Journal of Transportation Engineering, 110, pp. 55-66, 1984.

[2] A. Barton, *A simplified Ant Colony System applied to the Quadratic Assignment Problem*, Technical Report National Research Council of Canada no.47446, 2005.

[3] J. Braaksma, J. Shortreed, *Improving airport gate usage with critical path method*, Transportation Engineering Journal of ASCE 97, pp. 187-203, 1971.

[4] Y. Cheng, *Network-based simulation of aircraft at gates in airport terminals*, Journal of Transportation Engineering, pp. 188-196, 1998.

[5] Y. Cheng, *A rule-based reactive model for the simulation of aircraft on airport gates*, Knowledge-based Systems, 10, pp. 225-236, 1998.

[6] I. Crnkovic, *Component-based Software Engineering - New Challenges in Software Development*, Software Focus, 2001.

[7] H. Ding, A. Lim, B. Rodrigues, Y. Zhu, *The airport gate assignment problem*, hicss, p.30074b, Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS'04), pp. 74-81, Track 3, 2004.

[8] M. Dorigo and L. M. Gambardella, *Ant Colony System: A cooperative learning approach to the Traveling Salesman Problem*, IEEE Trans. Evol. Comp., 1:5366, 1997.

[9] M. Dorigo, *Optimization, Learning and Natural Algorithms (in Italian)*. Ph.D thesis, Dipartamento di Elettronica, Politecnico di Milano, Italy, pp.140, 1992.
[10] A. Fanea, S. Motogna, *A Formal Model for Component Composition*, Proceedings of the Symposium Zilele Academice Clujene, pp. 160-167, 2004.
[11] A. Haghani, M. Ching Chen, *Optimizing gate assignments at airport terminals*, Transportation Research, 32(6), pp. 437-454, 1998.
[12] T. Obata, *The quadratic assignment problem: Evaluation of exact and heuristic algorithms*, Tech. Report TRS-7901, Rensselaer Polytechnic Institute, Troy, New York, 1979.
[13] V. Maniezzo, A. Colorni, M. Dorigo, *The Ant System applied to the Quadratic Assignment Problem*, Technical report 94/28, IRIDIA, Université de Bruxelles, Belgium, 1994.
[14] C-M. Pintea, D.Dumitrescu, *Improving ant systems using a local updating rule*, Proc. 7-th Int. Symposium on Symbolic and Numeric Algorithms for Scientific Computing, IEEE C.S.Press, pp. 295-299, 2005.
[15] C. Szypersky, *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley, 1998.
[16] A. Vescan, S. Motogna, *Syntactic automata-based component composition*, The 32nd EUROMICRO Software Engineering and Advanced Applications (SEAA), Proceeding of the Work in Progress session, ISBN 3-902457-11-2, 2006.
[17] J. Xu, G. Baile, The Airport Gate Assignment Problem: Mathematical Model and a Tabu Search Algorithm, *HICSS*, 34th Annual Hawaii International Conference on System Sciences, Vol.3, 2001.
[18] S. Yan, C.-M. Chang, *A network model for gate assignment*, Journal of Advanced Transportation, 32(2), pp. 176-189, 1998.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: {cmpintea, avescan}@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LII**, Number 2, 2007

TOWARD A SIMPLE PHONEME BASED SPEECH RECOGNITION SYSTEM

MARGIT ANTAL

Abstract. This paper presents a simple speech recognition system using Gaussian mixtures as phoneme models. The proposed architecture does not follow the integrated search strategy. Instead we use a modular design. We propose two modifications to the Viterbi decoding algorithm in order to be applicable to our phoneme models. Both strategies have been implemented and tested on two corpora. Experiments have proved our phoneme recognition system reliability and its good recognition performance.

1. Introduction

The purpose of this paper is to present our findings during the construction and evaluation of our phoneme based speech recognition system. Despite the fact that good software packages already exist for solving this problem, we decided to develop our own software. The main objective was to use state of the art techniques, but only those which do not contradict human speech recognition. A secondary objective was to simplify the architecture of such a system to the extent of not decreasing the system performance and its usability. Nowadays it is important to create constrained speech recognition systems, which work reasonably in a low resource environment.

State of the art automatic speech recognition (ASR) is based on modelling the phonemes with hidden Markov models (HMM), using the well known three state left to right topology for each phoneme. This model incorporates an inherent phoneme duration, modeled by the state transition probabilities. Several papers [5, 8, 10, 11] noticed the negligible effect of these state transition probabilities on the recognition rate in HMM based ASR, and hence it is usual

Received by the editors: June 2007.

2000 *Mathematics Subject Classification.* 68T10, 62H30.

1998 *CR Categories and Descriptors.* I.5.2 [**Pattern Recognition**]: Design Methodology– *Classifier Design and Evaluation*; I.5.4 [**Computer Systems Organization**]: Special-purpose and Application-based Systems – *Signal Processing Systems* .

to ignore them or use the same value for each transition probability. Due to this observation we modeled every phoneme with a one state HMM, which can be considered as a Gaussian mixture (GMM).

Phoneme duration is an important problem for speech comprehension, especially in languages like Hungarian, in which most of the phonemes has both a short and long form. These durations are so important that even the written language uses different letters for each vowel, one for the short and one for the long form of the same phoneme. In the case of consonants there are no different written forms, but the letter is doubled. Good duration modelling can therefore be a major issue in these languages, not only for speech recognition but for speech synthesis too.

As a first step we performed some phoneme classification experiments in order to evaluate our phoneme models. These GMM phoneme models performed so well that we could go further to the problem of phoneme recognition. In this step we had to modify the classic Viterbi decoding algorithm (given by formula (13)) in order to make it suitable for GMM phoneme models. We should mention that the classic Viterbi algorithm without state transition probabilities (omitting the state transition matrix a_{ij} from formula (13)) made a huge number of insertion errors. In order to overcome this, our first attempt was to introduce explicit phoneme durations computed from speech corpora into the decoding process. The idea was taken from [9], but we simplified it. Levinson used statistical models for phoneme duration modelling (Gamma distributions), we used only the minimum and maximum duration for each phoneme. We went even further in simplifying this duration modelling by using the same fixed durational minimum and maximum for each phoneme. As this increased the decoding algorithm complexity by a factor D, which is the maximum duration of phonemes, we tried to find a cheaper solution for decoding.

In the second attempt we made another adaptation of the Viterbi algorithm for monophone one-state models, which introduces an empirical constant in order to be able to control the insertion errors. As we prove experimentally, this constant incorporates the average phoneme duration implicitly. This was our conclusion after we have tuned this parameter for two corpora: TIMIT the well-known English corpus and OASIS, a small Hungarian corpus for isolated word recognition.

Several measurements were carried out in order to demonstrate the viability of this simplified strategy. Whenever it was possible, we compared

our results with other results obtained on the same corpora, eventually using different phoneme modelling techniques. However, the purpose was not to outperform state of the art technologies, but the construction of such a system that is simple and efficient for some constrained speech recognition tasks.

This paper is organized as follows. Section 2 presents the architecture of our system, the feature extraction module, the acoustic-phonetic module, which is a standard GMM and the decoding module in which we propose modifications to the Viterbi algorithm. In section 3, we present the corpora used for experiments and the evaluations of the proposed decoding methods. We end with discussion and conclusions.

2. The recognition system

Our speech recognition system has a very simple modular architecture. The first module of the system is the feature extraction module. The extraction of Mel-frequency cepstral coefficients (MFCC) is presented in subsection 2.1. In this module only standard methods were used as recommended in [3, 5, 12].

The second module is the acoustic-phonetic module. We used Gaussian mixture models for training the phonemes based on phonetically segmented and annotated corpora. Once this stage is completed we can evaluate the phoneme models. As we have stated already, we used context independent phonemes modeled by Gaussian mixtures.

The third module is the phonetic decoding module containing two modified versions of the Viterbi decoding algorithm.

2.1. **Feature extraction module.** The extraction of reliable features is a very important issue in speech recognition. There are a large number of features we can use. Among others we can use is the speech waveform itself. However this has two main shortcomings. The first one is the dimension of this feature, and the second one is that time domain features are much less accurate than frequency-domain features. In the following we present the extraction of Mel-frequency cepstrum coefficients used in our system. This was implemented based on [5].

In our system the acoustic analysis of the speech signal was done by short-time spectrum analysis with 20 ms frames and 10 ms overlap between consecutive frames. For a frame length of 20 ms it can be assumed that the speech signal is stationary, allowing the computation of short-time Fourier spectrum.

Let us denote by $x[n],\ n = 0, 1, \ldots, N-1$ the samples from a frame. For this input signal we compute the Discrete Fourier Transform (DFT):

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi/N)kn},\ k = 0, 1, \ldots, N-1 \tag{1}$$

In order to reduce the dimension of the feature vector a filterbank composed of M triangular filters was used. The equation of the mth triangular filter is the following.

$$H_m[k] = \begin{cases} 0 & k < f[m-1] \\ \frac{k - f[m-1]}{f[m] - f[m-1]} & f[m-1] \leq k \leq f[m] \\ \frac{f[m+1] - k}{f[m+1] - f[m]} & f[m] \leq k \leq f[m+1] \\ 0 & k > f[m+1] \end{cases} \tag{2}$$

Such filters compute the average spectrum around each center frequency with increasing bandwidths.

Let us denote by f_l and f_h the lowest and the highest frequencies of the filterbank in Hz, F_s the sampling frequency in Hz, M the number of filters, and N the size of DFT.

The filterbank's boundary points $f[m]$ are uniformly spaced in the mel-scale:

$$f[m] = \frac{N}{F_s} B^{-1}(B(f_l) + m \frac{B(f_h) - B(f_l)}{M+1}) \tag{3}$$

where the mel-scale B is given by

$$B(f) = 1125 ln(1 + f/700) \tag{4}$$

and B^{-1} is its inverse

$$B^{-1}(b) = 700(e^{\frac{1}{1125}} - 1) \tag{5}$$

The next step is the computation of log-energy at the output of each filter

$$S[m] = ln(\sum_{k=0}^{N-1} |X[k]|^2 H_m[k]), \quad 1 \leq m \leq M \tag{6}$$

The Mel-frequency cepstrum is then the discrete cosine transform of the M filter outputs:

$$c[n] = \sum_{m=0}^{M-1} S[m] cos(\frac{\pi n}{M}(m - \frac{1}{2})) \quad 0 \leq n < M \tag{7}$$

For speech recognition applications it is typical to use a number of filters M between 24 and 40 and to evaluate only the first 13 coefficients given by equation (7). In our experiments we used $M = 28$ filters.

Temporal changes in spectra play an important role in human perception. One way to capture this information is to use delta coefficients that measure the change in coefficients over time. Delta features were obtained by evaluating the first and the second order delta cepstral coefficients given by the following equations

$$\Delta c_k = \frac{2(c_{k+2} - c_{k-2}) + (c_{k+1} - c_{k-1})}{10} \tag{8}$$

$$\Delta\Delta c_k = \frac{2(\Delta c_{k+2} - \Delta c_{k-2}) + (\Delta c_{k+1} - \Delta c_{k-1})}{10} \tag{9}$$

where c_k represents the feature vector containing the first 13 MFCC coefficients obtained using formula (7) for the kth time frame.

The combined cepstral, first and second order delta cepstral vectors form a set of 39-parameter feature vector (observation vector) $o_k = \begin{pmatrix} c_k \\ \Delta c_k \\ \Delta\Delta c_k \end{pmatrix}$, which were used in all the experiments described in this paper.

2.2. **The Acoustic-Phonetic module.** Observation densities in phonemes are modeled by mixtures of multivariate Gaussians. The proper number of Gaussians can be estimated separately for every phoneme or can be fixed the same value for every phoneme. We used the latter approach. Let us denote by M the number of Gaussian densities. In this case the observation density function for phoneme i, $b_i(\overrightarrow{o_t})$ has the form

$$b_i(\overrightarrow{o_t}) = \sum_{j=1}^{M} w_{ij} . \frac{1}{(2\pi)^{D/2}|\Sigma_{ij}|^{1/2}} \cdot e^{-\frac{1}{2}(\overrightarrow{o_t} - \overrightarrow{\mu_{ij}})^T \Sigma_{ij}^{-1} (\overrightarrow{o_t} - \overrightarrow{\mu_{ij}}))} \tag{10}$$

where D represents the dimensionality of the $\overrightarrow{o_t}$ observation (feature vector), $\overrightarrow{\mu_{ij}}$ and Σ_{ij} are the mean vector and the covariance matrix for the jth mixture

component. For every phoneme the mixture weights sum to unity ($\sum_{j=1}^{M} w_{ij} = 1$) in order to have a true probability function.

The complete model thus consists of the set of n phonemes, the ith phoneme being modeled by a GMM with the parameters $(w_{ij}, \overrightarrow{\mu_{ij}}, \Sigma_{ij}), 1 \leq j \leq M$. $\overrightarrow{\mu_{ij}}$ is a mean vector composed by D real numbers. We used diagonal covariance matrix, hence it can be represented by D real numbers. The complete model can be represented using $n * M * (1 + D + D) = n * M * (2D + 1)$ real numbers.

2.3. **Phonetic decoding module.** Let us denote by $O = \{\overrightarrow{o_1}, \overrightarrow{o_2}, \ldots \overrightarrow{o_T}\}$ the acoustic observation sequence, which has to be decoded into a phoneme sequence. The set of all phoneme sequences will be denoted by F. Essentially the task here is to find $\hat{f} \in F$ defined by

$$\hat{f} = \arg\max_{f \in F} P(f|O) = \arg\max_{f \in F} \frac{P(O|f) \cdot P(f)}{P(O)} \tag{11}$$

where $P(f)$ is known as the phonetic language model. Assuming that any observation sequence is equally likely, equation (11) becomes

$$\hat{f} = \arg\max_{f \in F} P(O|f) \cdot P(f) \tag{12}$$

Equation (12) expresses that we face a search problem. Phonetic transcription reduces to the task of finding the most likely phoneme sequence for the input sequence of acoustic vectors.

For HMM phoneme models Viterbi algorithm [5] solves the problem of finding the most probable state sequence.

We review the classic Viterbi algorithm, which will be adapted to our phoneme models in the following section. Let $\alpha_t(j)$ denote the maximum likelihood of $\overrightarrow{o_1}, \overrightarrow{o_2}, \ldots \overrightarrow{o_t}$ over all state sequences terminating in state j. This quantity can be evaluated recursively according to

$$\alpha_t(j) = \max_{1 \leq i \leq n} [\alpha_{t-1}(i) \cdot a_{ij}] \cdot b_j(\overrightarrow{o_t}) \tag{13}$$

where a_{ij} represents the state transition probability between state i and state j, $b_j(\overrightarrow{o_t})$ is the observation probability of $\overrightarrow{o_t}$ in state j

At every time instance we retain $B_t(j) = \arg\max_{1 \leq i \leq n} [\alpha_{t-1}(i) \cdot a_{ij}]$, $1 \leq j \leq n$ in order to be able to back trace the optimal path through the trellis.

The Viterbi algorithm presented previously is based on dynamic programming technique. Essentially it is a planar search algorithm through a lattice, where the lattice consists of points representing phoneme likelihoods for each

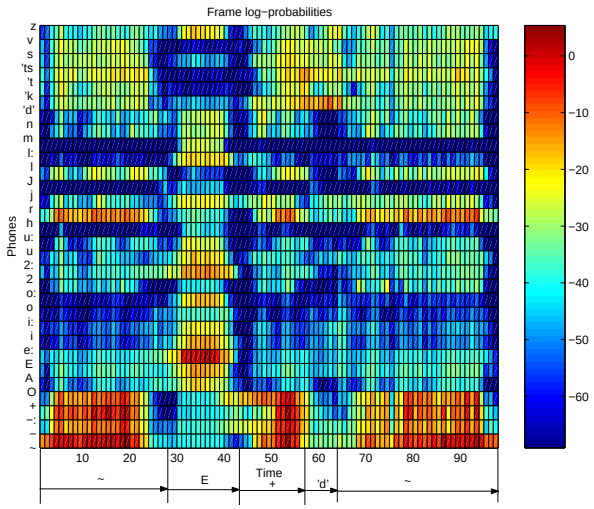


FIGURE 1. Search space

time instance. This search space is shown for the digit one ("egy" in Hungarian) in figure 1.

2.3.1. *Explicit usage of durations.* According to paper [14] it can be useful for the recognition stage to set a minimum number of frames, which can constitute a phoneme. This setting will help the decoder to decrease the number of insertion errors.

Instead of using complicated durational models we propose a simple modification of equation (13), which can be expressed as

$$\alpha_t(j) = \max_{1\le i\le n}\left\{\max_{\tau_{min}\le\tau\le\tau_{max}}\left\{\alpha_{t-\tau}(i)\cdot a_{ij}\cdot\prod_{\Theta=0}^{\tau-1} b_j(\overrightarrow{o_{t-\Theta}})\right\}\right\} \tag{14}$$

for $1 \le j \le n,\ 1 \le t \le T$, where τ_{min} and τ_{max} are the minimum and maximum allowable durations for any phonetic unit. It is supposed that observations are independent then $\prod_{\Theta=0}^{\tau-1} b_j(\overrightarrow{o_{t-\Theta}})$ computes the probability of the observation sequence $\overrightarrow{o_{t-\tau+1}}, \overrightarrow{o_{t-\tau+2}}, \ldots \overrightarrow{o_t}$ in the state j. Esentially we compute this probability for every allowed length τ, where $\tau_{min} \le \tau \le \tau_{max}$. Retaining at each stage of the recursion the values i and τ that maximize (14), makes possible back tracing through $\alpha_t(j)$ in order to obtain the best state and duration sequences.

In our system every phoneme is modeled by a one-state HMM so we could not use state transition probabilities, instead we used the following formula:

$$a_{ij} = \begin{cases} 1, & phone_j \; is \; allowed \; to \; follow \; phone_i \\ 0, & otherwise \end{cases} \tag{15}$$

which can be seen as a very simple language model. Using (15) reduces substantially the search space.

We computed the minimum $\tau_{min}(j)$ and maximum duration $\tau_{max}(j)$ of every phoneme $j = 1 \ldots n$, which were incorporated in formula (14) resulting in

$$\alpha_t(j) = \max_{1 \le i \le n} \left\{ \max_{\tau_{min}(j) \le \tau \le \tau_{max}(j)} \left\{ \alpha_{t-\tau}(i) \cdot a_{ij} \cdot \prod_{\Theta=0}^{\tau-1} b_j(\overrightarrow{o_{t-\Theta}}) \right\} \right\} \tag{16}$$

Section 3 presents experiments using both formulae: (14), (16).

2.3.2. *Implicit duration modelling.* Another approach to phoneme decoding is to use the Viterbi algorithm directly for the context independent phoneme models. In this case the state transition probabilities do not exist and we should omit in equation (13). Omitting state transition probabilities resulted in a huge number of insertion errors, which should be somehow overcome.

Firstly, we used the logarithmic form of Viterbi approximation as shown in the following formula:

$$\log \alpha_t(j) = \max_{1 \le i \le n} \{\log \alpha_{t-1}(i) + \log a_{ij}\} + \log b_j(\overrightarrow{o_t}) \tag{17}$$

Instead of omitting the term $\log a_{ij}$, we propose replacing it by I_{ij}, which is given as

$$I_{ij} = \begin{cases} \beta, \; if \; i = j \\ 0, \; otherwise \end{cases} \tag{18}$$

with $\beta > 0$, and the final formula for Viterbi approximation became:

$$\log \alpha_t(j) = \max_{1 \le i \le n} \{\log \alpha_{t-1}(i) + I_{ij}\} + \log b_j(\overrightarrow{o_t}) \tag{19}$$

Because larger β values will result in larger phoneme durations in the decoded phoneme sequence, this decoding process incorporates implicitly the average phoneme duration. It can be proved experimentally that the optimal value of the β parameter and the average phoneme duration for a given language

are directly proportionals. We should note that our method proposed for decoding is very similar to that proposed by Robinson in [13]. Robinson used a recurrent neural network for phoneme classification and for the decoding process he used a dynamic programming approach. In the decoding formula it was introduced a transitional cost, similar to the state transition probability in HMM. He worked with distances instead of probabilities, but the ideas are very similar. Moreover, he tried to introduce duration information and bigram probabilities into the transition function and observed that these additional information did not increased significantly the recognition accuracy.

3. Experiments

For measurements we used our software written in C++ language, which has a modular design being composed by a signal processing module for MFCC feature extraction, a Gaussian mixture module and a decoder module. The signal processing and the Gaussian mixture modules were successfully used for speaker identification systems too[1].

3.1. **Evaluation.** The standard evaluation metric for phoneme recognition systems is the phoneme error rate (PER). The PER measures the difference between the phoneme string returned by the recognizer and the correct reference transcription. The distance between the two phoneme strings is computed by the classical minimum edit distance algorithm [5]. The result of computation will be the minimum number of phoneme substitutions, insertions and deletions necessary to map between the correct and hypothesized strings. This can be expressed by the formula

$$PER = 100 \cdot \frac{I + S + D}{N} \tag{20}$$

where N represents the number of phonemes in the correct transcription, I, S and D represent the number of insertions, substitutions and deletions. Recognition accuracy is computed as

$$A = 100 - PER \tag{21}$$

Another performance measure could be the number of correct phonemes returned by the recogniser, which can be computed by the minimum distance algorithm. This will be denoted by C.

3.2. **Corpora.** We used two corpora for the experiments, the first one was TIMIT, a well known American English corpus, and the second one was OASIS Numbers, a small Hungarian corpus designed for number recognition. Because TIMIT is well known, we describe shortly only the Hungarian corpus.

The OASIS corpus is a small isolated -number corpus being developed at the Research Group on Artificial Intelligence of the Hungarian Academy of Sciences. The segmented part of the corpus contains speech from 26 speakers: 1 child, 9 female and 16 male voices. Each speaker reads the same 26 words twice. Any Hungarian number can be formed by concatenation from this set of 26 numbers. Each word is manually segmented and labelled phonetically. Twenty speakers were used for training and six speakers for testing. The corpus contains 31 phonemes, annotated using SAMPA symbols. The only modification was made for the notation of stop symbols, where instead of using one symbol, the closure part and the burst part were annotated separately. For the voiceless closure part it is used the symbol $/-/$ while for the voiced closure part the symbol $/+/$. For example instead of the symbol $/t/$ it were used two symbols:$/-/$and $/'t/$. Further information on this corpus can be found in [6, 7, 15]. Table 1 presents the exact content of the corpus used for training and test.

For the TIMIT corpus the phoneme models consist of $61*32*(2*39+1) = 154208$ real numbers and for the OASIS corpus $31 * 16 * (2 * 39 + 1) = 39184$ real numbers.

3.2.1. *Explicit usage of durations.* Our first attempt to phoneme duration modelling was a data driven approach. The minimum, maximum and the average phoneme durations were determined based on corpora. Figure 2 shows these values for the OASIS corpus.

In the first three experiments we used the same minimum and maximum duration for every phoneme and in the fourth experiment we introduced those shown in figure 2. Results are reported in table 2.

3.2.2. *Implicit duration modelling.* Using the second approach, firstly we present the phoneme recognition experiments for the TIMIT corpus. We used all the 61 phonemes of the corpus without grouping allophones. This will serve as a baseline for further system improvements. The first step was to determine the optimal value for β parameter introduced to the Viterbi decoding algorithm.

For training we used the whole training part of the corpus. For evaluation we used two sets, a smaller *timit_test_core* and a larger one *timit_test*, both had

Phoneme	Training	Test	Phoneme	Training	Test
-	519	156	i:	40	12
-:	40	12	j	80	24
'd'	118	36	J	80	24
'k	200	60	l	200	60
't	399	120	l:	40	12
'ts	200	60	m	120	36
+	120	36	n	559	168
~	2080	624	O	240	72
2	80	24	o	160	48
:2	40	12	o:	40	12
A:	120	36	r	160	48
E	600	180	s	160	48
e:	160	48	u	80	24
h	306	96	u:	40	12
i	240	72	v	240	72
			z	240	72

TABLE 1. Phoneme frequencies in training and test part of the corpus

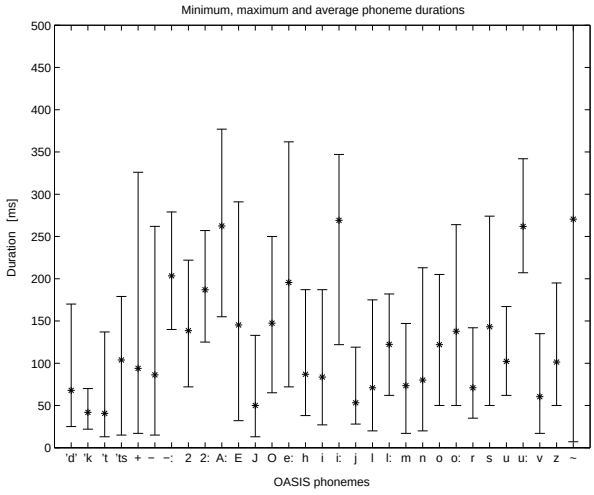


FIGURE 2. OASIS-Minimum, maximum and average phoneme durations

Dur.	D	I	S	A	C
1..50	0.04%	73.10%	2.54%	24.30%	97.40%
2..50	0.17%	27.72%	3.45%	68.65%	96.37%
3..50	0.69%	17.09%	4.66%	77.54%	96.64%
$\tau_{min}..\tau_{max}$	1.12%	15.71%	6.21%	76.94%	92.65%

TABLE 2. Phoneme recognition -OASIS - explicit phoneme durations

Evaluation	β	D	I	S	A	C
timit_test_core	11	7.41%	7.68%	37.76%	47.14%	54.81%
7525 phonemes	13	9.75%	4.81%	36.37%	49.06%	53.87%
TR: 61 models	15	12.14%	3.21%	34.79%	49.84%	53.06%
TE: 61 models	17	14.35%	2.19%	33.27%	50.17%	52.37%
	20	18%	1.26%	30.88%	49.86%	51.12%
timit_test	11	6.79%	7.39%	34.54%	51.27%	58.66%
65825 phonemes	13	9.13%	4.76%	33.12%	52.98%	57.74%
TR: 61 models	15	11.42%	3.24%	31.66%	53.67%	56.91%
TE: 61 models	17	13.7%	2.21%	30.18%	53.90%	56.11%
	20	17.16%	1.31%	28.12%	53.40%	54.71%
timit_test	11	7.03%	7.63%	25.68%	59.65%	67.28%
65825 phonemes	13	9.35%	4.97%	24.48%	61.18%	66.16%
TR: 61 models	15	11.55%	3.37%	23.38%	61.68%	65.05%
TE: 39 models	17	13.81%	2.32%	22.16%	61.70%	64.00%
	20	17.22%	1.37%	20.45%	60.95%	62.32%

TABLE 3. Phoneme recognition - TIMIT

been proposed by the creators of the corpus. While the smaller set contains 192 sentences, the larger one is formed by 1680 sentences. Table 3 shows the recognition accuracies together with the three type of errors for various values of the β parameter. As phoneme models 32 Gaussians models were used with diagonal covariance matrices. The first evaluation set *timit_test_core* contains 7525 phones, while *timit_test* contains 65825 phones. These results were obtained without using language model, which means that it was allowed for every phoneme to follow every other phoneme.

In the third part of the table 3 we present the results obtained by the same experiments with a modification in the interpretation of the decoding process.

Paper	Method	LM	TR	TE	ACC.	CORR.
Ostendorf[4]	SSM+CI	bigram	61	39	64.20%	70.00%
This paper	GMM+CI	0gram	61	39	61.70%	64.00%
Robinson[13]	REPN+CD	0gram	61	61	61.70%	69.10%
Robinson[13]	REPN+CD	bigram	61	61	63.50%	70.00%
Robinson[13]	REPN+CD	bigram	61	39	69.80%	76.50%

TABLE 4. TIMIT - Phoneme recognition. LM - Language Model, TR - Number of phoneme models trained, TE - Number of phoneme models for decoding, ACC -Accuracy, CORR -Correct, SSM - Stochastic Segment Model, REPN - Recurrent Error Propagation Network, CI - Context Independent phoneme models, CD - Context Dependent phoneme models

We used 61 phoneme models for decoding, but before applying the minimum distance algorithm, we converted the 61 phonemes to the 39 phoneme groups, as suggested by the creators of the corpus. This step reduced substantially the substitution errors, which suggests us that the phoneme grouping influences mainly the substitution errors.

Table 4 presents comparative results obtained on TIMIT. It can be seen that the best results were obtained by the neural network modelling, however this model is not fully comparable to the other two papers because this represents a context dependent modelling of the phonemes.

In the following we present results obtained for the Hungarian corpus. In this case, due to the limited amount of training data we used as phoneme models mixtures of 16 Gaussians.

Table 5 presents the recognition results obtained for various values of the β parameter. Figure 3 shows β parameter tuning for both corpora.

Our results compare favorably with those published in [15] on this corpus. They reported 82.05% recognition accuracy, using a hybrid ANN-HMM framework.

4. Discussion and conclusions

One of the most important finding of this work is that we achieved very good phoneme recognition accuracy with a very simple phoneme modelling and an even simpler phonetic decoding strategy. The second decoding strategy, in which we introduced the parameter β performed better than the first

β	D	I	S	A	C
11	0.52%	27.93%	5.18%	66.36%	94.30%
13	0.82%	21.80%	5.53%	71.85%	93.65%
15	1.42%	17.35%	5.65%	75.56%	92.91%
17	1.68%	14.35%	5.95%	78.02%	92.05%
20	2.37%	11.27%	6.00%	80.35%	91.62%
23	3.49%	9.15%	5.82%	81.51%	90.67%
25	4.06%	7.77%	5.78%	82.38%	90.15%
27	4.83%	7.08%	5.83%	82.25%	89.33%
30	6.43%	5.82%	5.70%	82.03%	87.86%

TABLE 5. Phoneme recognition - OASIS - implicit duration modelling

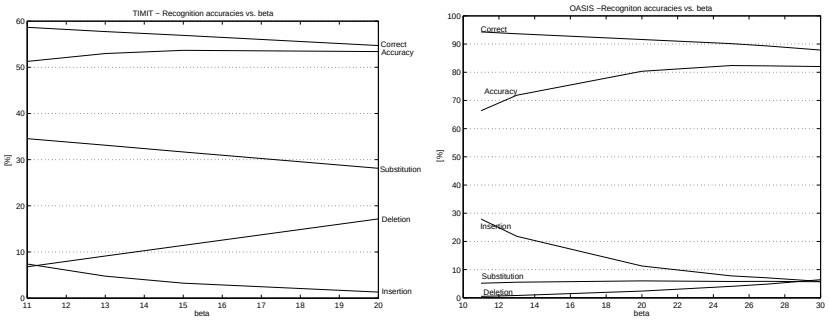


FIGURE 3. Phoneme recognition vs. β parameter

one, which considers different phoneme durations. Not only the recognition accuracy is better in the second approach, but the algorithm itself is a very efficient one.

Most of the papers working with the TIMIT corpus report phoneme recognition results for the reduced phoneme set. In order to produce comparable results, before computing the minimum edit distance between the recognised phoneme string and the original one, we converted the phonemes to their phoneme groups. This yields a better recognition accuracy, decreasing especially the substitution errors. In this way working with the reduced phoneme set increased approximately with 8% the recognition accuracy. For the 39 phoneme groups we obtained 61.70% recognition accuracy without using any phoneme level language model. The first decoding approach was not used for

TIMIT as this algorithm is a very inefficient one and has increased the time for decoding.

For the OASIS corpus both the proposed decoding techniques were evaluated. For the first decoding technique we have found that imposing a minimum phoneme duration (3 frames in our case) yields the same good result as using the phoneme specific minimum and maximum durations. This could be due to the limited amount of training data in this corpus. We should note that 3 frames roughly corresponds to the average of minimum durations over the whole phoneme set. The second decoding technique has shown its superiority over the first one. With the parameter β tuned for maximum accuracy we obtained 82.38% recognition accuracy, which compares favorably to 82.05% found in [15].

We believe that the most important finding is that we obtained these results by using only models and algorithms which do not contradict in their functionality human speech recognition. Despite the fact that phoneme recognition accuracy was not increased, our simple phoneme recognition system warrants stable and reliable behaviour with a good recognition performance.

References

[1] Antal, M., Toderean, G., Speaker Recognition and Broad Phonetic Groups, Proc. 24th IASTED International Multi-Conference on Signal Processing, Pattern Recognition and Applications, Febr. 15-17, Innsbruck, Austria, pp. 155-158, 2006.

[2] Bourlard, H., Hermansky, H., Morgan., N., Towards Increasing Speech Recognition Error Rates, Speech Communication, Vol. 18., pp. 205-231, 1996.

[3] Deller, J.R., Hansen, J. H. L., Proakis, J. G., Discrete-Time Signal Processing of Speech Signals, John Wiley & Sons, 2000.

[4] Digalakis V., Ostendorf M., Rohlicek, J. R., Fast Search Algorithms for Connected Phone Recognition Using the Stochastic Segment Model, IEEE Trans. on Signal Processing, December, pp. 173-178, 1992.

[5] Huang, X., Acero, A., Hon, H-W., Spoken Language Processing, A Guide to Theory, Algorithm and System Development, Prentice Hall, 2001.

[6] Kocsor, A., Kuba, A. Jr., Toth, L., An Overview of the OASIS Speech Recognition Project, Proceedings of the 4th International Conference on Applied Informatics, August 30 - September 3, Eger-Noszvaj, Hungary, pp. 94-102, 1999.

[7] Kocsor, A., Toth, L., Kuba, A., Kovacs, K., Jelasity, M., Gyimothy, T., Csirik J., A Comparative Study of Several Feature Transformation and Learning Methods, International Journal of Speech Technology, pp. Vol. 3, Nr 3/4, pp. 253-262, 2000.

[8] Lee, K-F., Hon, H-W., Speaker-independent Phone Recognition Using Hidden Markov Models, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 37, No. 11, 1989.

[9] Levinson, S. E., Ljolje, A., Miller, L. G., Continuous Recognition from Phonetic Transcription, Proceedings of a workshop on Speech and Natural Language, Pennsylvania, U.S., pp. 190-199, 1990.

[10] Mari, J. F., Fohr, D., Junqua, J-C., A second order HMM for high performance word and phoneme-based speech recognition, IEEE Transactions on Speech and Audio Processing, vol. 23, no. 2, pp. 435-438, 1996.

[11] Pylkkonnen, J., Phone Duration Modeling Techniques in Continuous Speech Recognition, Master thesis Helsinki University of Technology, 2004.

[12] Rabiner, L., R., Juang, B.H., Fundamentals of Speech Recognition, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[13] Robinson, T., Fallside, F., A Recurrent Error Propagation Network Speech Recognition System, Computer Speech and Language, vol. 5, no. 3, pp. 259-274, 1991.

[14] Toth, L., Kocsor, A.: Explicit Duration Modelling in HMM/ANN Hybrids, Matousek et al. (eds.): Proceedings of TSD 2005, LNAI 3658, pp. 310-317, Springer, 2005.

[15] Toth, L., Kocsor, A., Csirik, J., On naive Bayes in Speech Recognition, In. J. Appl. Math. Comput. Sci., Vol. 15, No. 2, pp. 287-294, 2005.

Sapientia - Hungarian University of Transylvania,Faculty of Technological and Human Sciences, 540053 Tg.-Mures,540485, Soseaua Sighisoarei 1C, Romania
E-mail address: manyi@ms.sapientia.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LII**, Number 2, 2007

KNOWLEDGESENSE: ENCYCLOPEDIC SYSTEM BASED ON SEMANTIC SEARCH THROUGH NLP

PAUL-VALENTIN BORZA, DANIEL GHIŢĂ, MIHAI NADĂŞ, OVIDIU SABOU, AND SIMONA MOTOGNA

Abstract. The article presents the solution that won the Imagine Cup 2007 National Finals and represented Romania at the Imagine Cup 2007 Software Design Invitational Worldwide Finals in Seoul, Korea August 5-11, 2007. In 2007, the Imagine Cup Theme was "Imagine a world where technology enables a better education for all". We created a unique solution that enables styles of natural interaction with a smart omnipresent humanlike entity, as it exploits human thinking within the education domain on an instant question answering based type of learning - self-education: KnowledgeSense. The solution takes advantage of intelligent devices and technologies that increasingly pervade modern environments - humans interact and learn intuitively while having a rich and complete experience with a virtual human person - Anna - that recognizes speech and gestures. Plus, it is also reachable through widely used communication channels like phones - SMS messages, VoIP or PSTN calls - and instant messaging networks. This approach emphasizes natural behavior by providing the opportunity to adapt to human needs and not reverse.

1. Introduction

Information is usually defined as "organized data", "data endowed with relevance and purpose" or "interpreted data" etc.; these definitions point to the fact that information includes human participation in the purposeful organization of raw data. Knowledge can only reside in one's mind and is the result of human experience and reflection based on a set of beliefs that are at the same time individual and collective; for instance, Nonaka and Takeuchi

Received by the editors: September 15, 2007.

2000 *Mathematics Subject Classification.* 68N01, 68T50.

1998 *CR Categories and Descriptors.* H.3.1 [**Information storage and retrieval**]: Content Analysis and Indexing – *Dictionaries, Indexing methods, linguistic processing*; H.3.3 [**Information storage and retrieval**]: Information Search and Retrieval – *retrievel model, search process*; H.4.m [**Information System Applications**]: Miscellaneous – .

define knowledge as "true and justified belief". The key difference between knowledge compared to information can be summarized by the role played by human beings. In the case of knowledge, as simple as it may seem, individuals play a prominent role as creators, carriers, conveyors and users; in contrast, in the case of information, these same functions can happen outside humans and without their direct influence.

KnowledgeSense uses the human knowledge that is continuously published on safe information sources through the Internet - Wikipedia, Encarta, Britannica and The World FactBook etc. - ensuring the trustworthy property of the vast content that the system is prepared to process. Finding answers through keywords - an approach that is often used in search engines like Windows Live and Google etc. - is extremely unintuitive; sometimes, choosing the right keywords becomes a time consuming challenge even to the most experienced user. People have always asked questions to quickly solve their misunderstandings, so the most natural way of solving the "international system units smallest prefix" puzzle, is to ask "What's the smallest metric prefix unit¿' - The answer is "yocto".

2. Architecture

Take a look at the architecture level of abstraction of the Enterprise Solution Patterns [5] - KnowledgeSense Pattern Frame from figure 1. From the application viewpoint, as Figure 2 shows, KnowledgeSense is an Object-Oriented Application that is logically structured as a Three-Layered Services Application. From the database viewpoint, the application is based on the OLTP processing model. From the infrastructure viewpoint, the hardware and network architecture are based on Four-Tiered Distribution (see figure 3), which calls for separate physical tiers for Web server and application server functionality. And finally, from the deployment viewpoint, we have created a Deployment Plan to map components to servers, based on a Smart Client Application.

Running on Windows Vista, the desktop application was designed and built upon Windows Presentation Foundation to offer a modern GUI and Experience. With the clear separation between the logic and the look of the program, WPF enabled quick development of a good looking interface that remains intuitive and serves its purpose - show information in a fast and convenient way - while maintaining the Windows Vista UX guidelines.

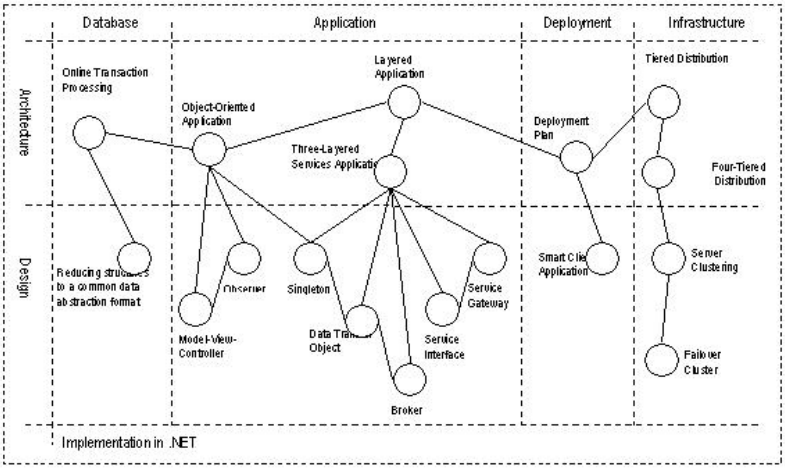


FIGURE 1. Enterprise Solution Pattern Frame

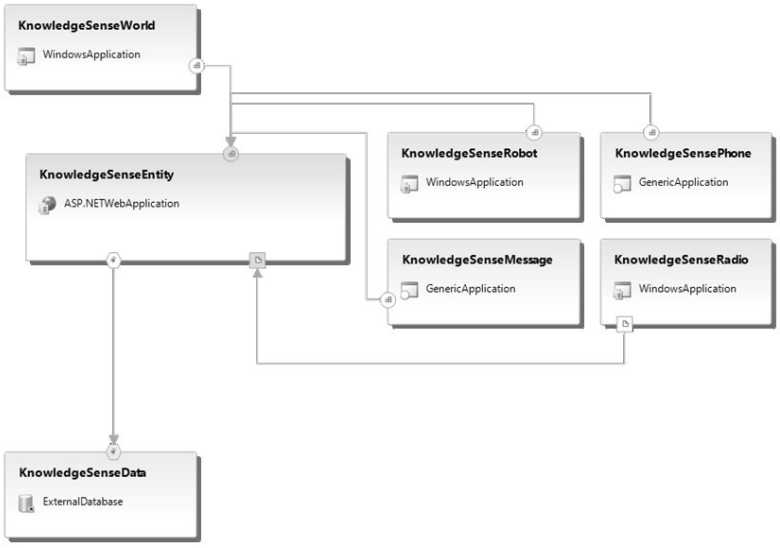


FIGURE 2. Enterprise Solution Pattern Frame

In addition to the classical way of presenting information (text, images, video etc.), a new dimension was introduced, bringing interactivity innovation

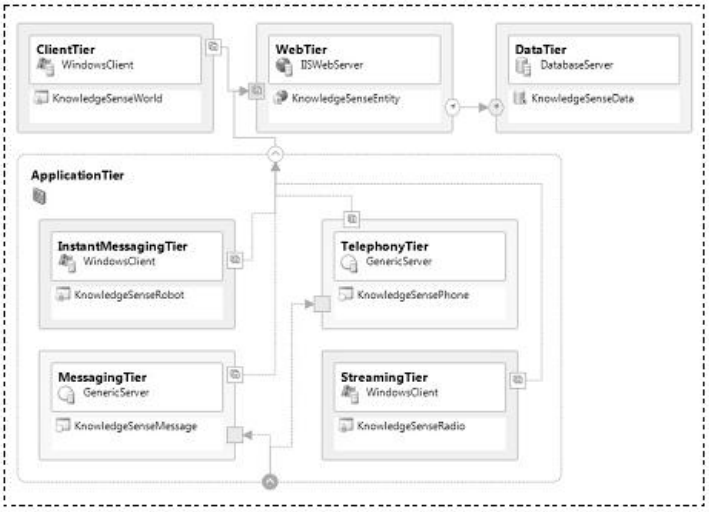


FIGURE 3. Four-Tiered Distribution

in electronic encyclopedias. A virtual assistant and a virtual world were integrated using OpenSceneGraph[2] - a 3D toolkit that enables advanced 3D graphics techniques, already used in many well known 3D graphics solutions like flight simulators and CAD products - achieving a completely new experience.

Gesture recognition data is gathered using a Nintendo Wii Remote Controller[3] (a $40 value) that provides six degrees of freedom through the use of an analog accelerometer device and an optical sensor, and then processed through the Hidden Markov Models[4] with the Forward-backward, Viterbi and Baum-Welch algorithms that recognizes gesture patterns and executes the associated actions. In addition to the sensors, the solution uses the other unique functionalities of the remote: rumble, speaker, direction pad etc.

The VoIP component establishes connections over the TCP implementation of the Session Initiation Protocol. Running on a standard protocol, a multitude of SIP-enabled softphones and hardphones can dial the Windows Workflow Foundation based application that runs inside Speech Server 2007 Beta. Unlike voice services that use the PSTN - with low audio quality - the voice recognizer and synthesizer are able to perform their tasks much better,

as they receive and produce high quality sound. PSTN phone calls are handled by the Cantata Technology TR1000 for MSS Speech - telephony and voice processing board.

The DotMSN Messaging Library[6] is used as a wrapper over the Windows Live Messenger network protocol, being able to trigger all instant messaging events required to chat with a person; the user might not even know it is talking with a machine.

The full product grid that provides a non-detailed, yet comprehensive snapshot of used technologies is presented in Appendix A.

3. Hibride Search Algorithm

KnowledgeSense is integrating a hybrid search mechanism , as in figure 4 that enables high accuracy in the process of getting the right answer for a certain question. We are combining traditional search technologies like the Microsoft SQL Server's Full Text Search feature and the Windows Live Search, with our custom build artificial intelligence driven natural language processing engine.

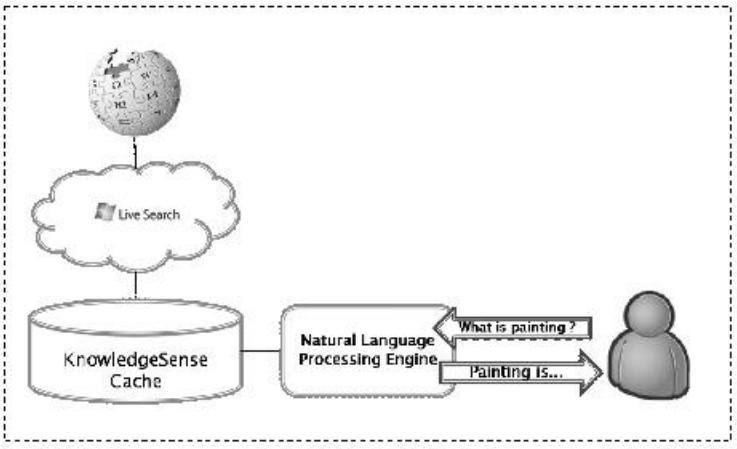


Figure 4. QA Flow

Our system uses a database in which the relevant encyclopedic information is stored. But as the system relies on the knowledge gathered on the Internet to provide users with answers, we need an intelligent process responsible for acquiring data from all the trusted sources assigned to our application.

From a technological point of view, our algorithm is capable of solving three different question answering situations. All of these situations start with the same procedure of breaking up the question into relevant keywords. For example if a user asks What is painting?" the engine will process the question and it will generate the keywords is painting". From this point on, the algorithm works differently for the mentioned situations.

The first case in when the database does not contain a relevant article for the keywords generated by the engine. We know if a set of keywords match an article by using the database's Full Text Search feature that returns for each article stored a match accuracy, which tells us if the article is relevant or not. And because we said above that in this case the database will not contain relevant articles for the given question, the algorithm continues the search on the next level, searching the Internet. The algorithm uses Windows Live Search to search the trusted data sources present on the Internet (e.g. Wikipedia) in order to retrieve articles that match the search criteria. Once this process is over, the engine parses the text of the articles thus creating a semantic tree of the information present in the article and after this is done all the data is stored into the database providing information for further queries. When the parsing process is over, the engine uses the semantic trees from each relevant article and retrieves the answer for the given question (e.g. "Painting is the process of applying color to a surface").

The second case is when the database already contains the semantic tree for the article relevant to the given question. In this case the semantic tree associated with the article is sent back to the engine, where it extracts the relevant answers for the given question. And to assure that our information is updated permanently, we run a comparing test between the information of the source and the article that is present in our database. If there are differences, the article is reparsed in order to deliver the most relevant answers. The third case is possibly the easiest of all. After the system delivers an answer for a question, the answer is stored into the database for a better performance the next time that exact question is asked. Thus, if the asked question is already answered, the system will return the answer without analyzing the semantic tree or searching for new articles on the Internet.

The engine that processes natural language identifies each question and parses the input to construct a tree with word relations[8] ; the WH-part - who, what, how etc. - and statement segment are extracted to classify the input accordingly. Keywords are generated and matched on articles where the

system looks for possible answers; the highest rated answer is rephrased with the additional available information and provided back to the user in human language[9].

4. Anna, the 3D animated assistant

The humanoid assistant from our application had to be as realistic as possible. In order to achieve this, we had to use the best real time rendering technologies available to us at that time. Since we already had some experience with OpenGL and OpenSceneGraph, we chose these technologies for rendering Anna and the objects in the virtual environment.

The first major problem we encountered when developing KnowledgeSense World (the desktop client application) was the integration between WPF (Windows Presentation Foundation) and a 3D graphics API. Although WPF provides some support for rendering 3D objects, it wasn't enough for us. We had to render tens of thousands of polygons at interactive frame rates with the assistant's 3D model included. WPF isn't suitable for this because it doesn't provide a low level way of specifying geometric data and it also doesn't provide hardware skinning support (for animating the character directly on the video card).

In order to render data with OpenGL, a special type of window has to be created. Since access to the operating system and hardware resources is better handled with C++, and OpenSceneGraph is a C++ toolkit, we needed something that could glue together the C# part (user interface and the logic of the application) and the C++ part (for rendering the scene). C++/CLI is a language designed for creating systems that interact with .NET and native components at the same time. The rendering part, which was implemented in native C++, was wrapped by a thin C++/CLI module that provided a simple .NET API for controlling the objects within the virtual world (with features like changing the position and rotation of objects, animating the character, changing the camera properties, etc.).

The realism of the scene was improved by adding static shadows (baked with 3D Studio Max) at high resolutions. It wasn't a problem because the environment was supposed to be changed rarely so the static approach for lighting the virtual world was enough (the resulting images had very high quality).

Throughout the development process, three models were used as the virtual body of Anna. The first two models were custom models made by

us, while the third model was professionally built. For character animation, we used the most popular open source library, named Cal3D, which is available at http://cal3d.sourceforge.net. We have acquired a license to use the third model (named Masha) from Turbosquid.com and integrated it into our scene after it was rigged (rigging is the process of attaching a skeleton to a mesh) by a professional Blender artist. Because the Cal3d exporter for Blender was the only one that worked for us (the 3Dsmax version was out of date), Blender was the main tool for manipulating the 3d content related to the virtual assistant. The final rendering of the character was done using osgCal, an OpenSceneGraph plug-in that renders Cal3d models. The development of the models and theirs transformations can be seen at http://www.borza.ro/demo/knowledgesense.

Table 1: Technologies

Technology Name and KnowledgeSense Component	Core	Message	Phone	IE Add-In	Robot	World	Team W
Microsoft Windows							
• Windows Vista Business Edition				X	X	X	
• Windows Internet Explorer 7.0				X			
• Windows Live Messenger 8.1					X		
• Windows Mobile 5.1 Pocket PC Phone Edition		X	X		X		
Microsoft Servers							
• Windows Server 2003 R2 Standard Edition	X		X		X		X
• Internet Information Services 6	X		X				X
• Windows SharePoint Services 2							X
• SQL Server 2005 Standard Edition	X						X
• Database Engine	X						
• Speech Server 2004 R2			X				
• Office Communications Server 2007 Speech Server			X				
• Visual Studio 2005 Team Foundation Server							X
Microsoft Developer Tools							
• Visual Studio 2005 Team Suite	X	X	X	X	X	X	X
• Visual C#	X	X	X		X	X	
• Visual C++						X	
• Expression Blend						X	
Additional Developer Tools							
• Autodesk 3ds Max 9						X	
• Blender 2.43						X	
• Counterpath X-Lite 3.0			X				
Microsoft Frameworks, APIs and SDKs							
• .NET Framework 2.0	X		X		X	X	
• ADO.NET	X						
• ASP.NET	X						
• .NET Compact Framework 2.0		X					
• .NET Framework 3.0			X			X	

• Windows Presentation Foundation						X	
• Speech API 5.3						X	
• Windows Workflow Foundation			X				
• Windows Mobile 5.0 for Pocket PC SDK		X					
• Windows Live Search API 1.1	X						
Additional APIs and SDKs							
• Xih Solutions DotMSN API 2.0.2					X		
• Princeton WordNet API 3.0	X						
• Proxem Antelope API 0.7.1	X						
• Stanford Parser API 1.5.1	X						
• Wii Remote Controller API 1.1						X	
• OpenSceneGraph SDK 1.2						X	
Devices							
• Cantata Technology TR1000 for MSS			X				
• HP iPAQ hw6915		X					
• MSI Star Key 2.0 Long Range Bluetooth Dongle						X	
• Nintendo Wii Remote Controller						X	

5. Conclusion

The solution delivers trustworthy educational knowledge in an intuitive manner that enables questions to be stated in natural language; subjects are presented in an interactive graphical environment. A complete set of platforms are available for application use.

References

[1] Dion Hinchcliffe: Patterns for High-Integrity Data Consumption and Composition - The Architecture Journal, Data by Design, http://msdn2.microsoft.com/en-us/arcjournal/bb245676.aspx

[2] Robert Osfield and Don Burns: OpenSceneGraph, http://www.openscenegraph.org/

[3] Brian Peek: Managed Library for Nintendo's Wiimote - MSDN Coding4Fun, http://blogs.msdn.com/coding4fun/archive/2007/03/14/1879033.aspx

[4] Lawrence R. Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition - Proceeding of the IEEE, February 1989 http://www.caip.rutgers.edu/∼lrr/Reprints/tutorial on hmm and applications.pdf

[5] David Trowbridge, Dave Mancini, Dave Quick, Gregor Hohpe, James Newkirk and David Lavigne: Enterprise Solution Patterns Using Microsoft .NET - MSDN Patterns and Practices, http://msdn2.microsoft.com/en-us/library/ms998469.aspx

[6] *** Xih Solutions: DotMSN, http://www.xihsolutions.net/dotmsn/

[7] *** National Institute of Standards and Technology: Question Answering Collections, http://trec.nist.gov/data/qa.html

[8] *** Princeton University Cognitive Science Laboratory: WordNet, http://wordnet.princeton.edu/

[9] *** Proxem: Advanced Natural Language Object-oriented Processing Environment, http://www.proxem.com/Antelope/tabid/55/Default.aspx
[10] *** Stanford Natural Language Processing Group: Stanford Parser, http://nlp.stanford.edu/software/lex-parser.shtml

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

E-mail address: bp20416,gd20439,nm20474,so20492@scs.ubbcluj.ro, motogna@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LII**, Number 2, 2007

COMDEVALCO — A FRAMEWORK FOR SOFTWARE COMPONENT DEFINITION, VALIDATION, AND COMPOSITION

BAZIL PÂRV, SIMONA MOTOGNA, IOAN LAZĂR, ISTVAN CZIBULA, AND LUCIAN LAZĂR

Abstract. This paper introduces **ComDeValCo** - a framework for Software **Com**ponent **De**finition, **Val**idation, and **Co**mposition. This is the first paper in a series describing current and further developments of this framework, which includes a modeling language, a component repository and a set of tools. The object-oriented modeling language contains fine-grained constructions, aimed to give a precise description of software components. Component repository is storing valid components, ready to be composed in order to build more complex components or systems. The toolset contains tools dedicated to component definition, validation, and composition, as well as the management of component repository.

1. INTRODUCTION

Software systems become more and more complex. In most situations, where the complexity of the problem to be solved is an important issue, the decomposition is used - the initial problem is splitted into small sub-problems, then each sub-problem is solved independently (or their solutions are identified), and finally the target system is built by composing the solutions of sub-problems. The evolution of software systems development include the use of several paradigms: procedural, modular, object-based and oriented, and component-based; most authors consider component-based development as the paradigm of the third millenium. The drivers of this evolution were at least the following:

- the increased complexity of the problems to be solved (and consequently of the systems to be built);

Received by the editors: November 10, 2007.

2000 *Mathematics Subject Classification.* 68N30.

1998 *CR Categories and Descriptors.* D.2.4 [**SOFTWARE ENGINEERING**]: Software/Program Verification – *Formal methods, Model checking, Validation*; D.2.13 [**SOFTWARE ENGINEERING**]: Reusable Software – *Reuse models*; I.6.5 [**SIMULATION AND MODELING**]: Model Development – *Modeling methodologies* .

- the need for performance (w.r.t. time, money, and throughput): producing new state-of-the-art systems in short time, and with less money.

One of the drivers of this evolution was *reuse*. Early forms of software reuse are collectively known as *code reuse*. Nowadays, software reuse covers also *design reuse*. Successful design fragments are collected into catalog form and collectively known as *design patterns*; they represent not complete designs, but partial solutions, ready to be reused into new designs or contexts. Also, class libraries evolved into *frameworks*, which represent complete system architectures. They are an incarnation of *inversion of control* design principle, being a set of cooperating classes that make up a reusable design from a specific application domain.

The paper is organized as follows: after this introductory section, the second one is discussing component-based development process, and the current status of research and industry efforts in the field. Third section presents the proposed solution, **ComDeValCo** framework, detailing its components: modeling language, component repository and the toolset. The last section contains some conclusions and plans further efforts.

2. THE PROBLEM

2.1. **The problem: component-based software development.** The process of component-based software development (or CBD for short) has two subprocesses more or less independent: component development process and system development process. Naturally, the requirements on components are derived from system requirements; the absence of a relationship, such as causal, may produce severe difficulties in both sub-processes mentioned above.

The system construction by assembling software components [CL02] has several steps: component specification, component evaluation, component testing, and component integration. The system development sub-process focuses on identifying reusable entities and selecting the components fulfilling the requirements, while in the component development sub-process the emphasis is on component reuse: from the beginning, components are designed as reusable entities. Component's degree of reuse depends on its generality, while the easiness in identification, understanding, and use is affected by the component specification. The sole communication channel with the environment is the component's interface(s). In other words, the client components of a component can only rely on the contracts specified in the interfaces implemented by the component. Thus, it is obvious that component development must be interface-driven. One of major CBD challenges is to design appropriate interfaces.

In our opinion, the main CBD challenge is to provide a general, flexible and extensible model, for both components and software systems. This model

should be language-independent, as well as programming-paradigm independent, allowing the reuse at design level.

2.2. **CBD design process models.** The design process of a component-based system [HW00] follows the same steps as in the classical methods: the design of architecture, which depicts the structure of the system (which are its parts) and the design of behavior (how these parts interact in order to fulfill the requirements). The structural description establishes component interconnections, while behavioral description states the ways in which each component uses the services provided by interconnected components in order to fulfill its tasks.

The main idea of CBD is to build the target system from existing components; this has several consequences on the target system's life-cycle. First, the system development sub-process [HW00] is separated from the component development sub-process. Second, a new sub-process arises: component identification and evaluation. Third, the activities in both sub-processes differ from the traditional methods: the focus is on component identification and verification (for system development), and on component reuse (in the case of component development).

The paper [CL02] describes a software systems development model which can be used in component-based development. The classical waterfall life-cycle model was upgraded such that it contains component-centric activities: requirements analysis and design - specific to the waterfall model - are combined with component identification and selection - specific to the component-based development. The design stage includes architectural design and activities related to component identification, selection, and adaptation.

Another viewpoint, given in [WR02], considers the following steps in building a software system from components: a) connecting the components such that they match; b) understanding the interconnections between components, and c) examining the behavior of the whole target system with respect to the requirements.

2.3. **Component and system models.** There are many ways to deal with component-based software development. The simplest one is to add to the contract (interface) of the component all requirements w.r.t. its use (the meaningful interconnections to other components). Unfortunately, this supplement to the component specification is a time and effort-consuming activity. For every newly-created component, one must identify all compatible components, and after that the contracts of these components must be updated in order to include this new component.

An alternative solution, given in [WR02], is to think the behavior of the target system as a separate activity, which is performed without accessing

the target system under construction. Unfortunately, this activity is very complex, but there are models which can help. These models do not reproduce the behavior of the whole system; they will cover only particular aspects of the system which are of interest at a specific time. This approach neglects insignificant details, thus reducing the complexity of the resulting model, total effort and time for building the model.

Building and testing a real target system is more difficult and takes a greater volume of resources than the corresponding model evaluation. This is because the models do not address complexity of the system and the subtleties of its environment. Other potential benefit of using models is their controllability, i.e. their evaluation before the target system is designed. In this case, the models are analyzed, simulated, and evaluated using software tools. The system developer builds a model which is taken by the evaluation tool. The comparison of the model behavior with respect to the system requirements is made by using either the modeling language or some other specialized languages.

In order to be evaluated, the models need to be precise, complete, and consistent. Generally speaking, if the degree of model (i.e. modeling language) formality is low, the model is a good candidate for inconsistencies, because some modeling constructs do not have a unique interpretation. When the model is used to assist the process of designing interactions between different components / parts of a system, or to assess the correctness of the system, preciseness, completeness and consistency are a must.

The success of using models (formal or not) is influenced in part by the availability and the degree of acceptance of modeling tools and techniques developed by the software development community. Those who build models need to perceive the usefulness of the models [HW99], need to find a tradeoff between model complexity and its ease of use. It is convenient to build simple models, without great investments in time and intellectual effort. More important, the resulting models need to be accessible, easy to understand and analyze, and to have a reasonable degree of formality.

It is recognized that modeling is not used today in the software development process at its full strength. Usually, models are only simple design notes, thrown away after the coding is completed. However, model-based approach (or model-driven approach in software development, MDA) gains more adepts. In MDA, the model of the system is the center of software development process. At least four modeling notations are currently used: finite state machines ([EG03], [LL00], [WR00]), statecharts ([GM95]), Petri nets ([AW], [PJ02]) and role-activity diagrams ([HW00], [HHW01], [HWC04], [WR02]).

An important direction concerning the use of models in the CBD process is represented by the Object Management Group consortium (OMG) efforts,

known collectively as executable UML. For example, [OMG05a] describes the semantics of some simple UML constructions, intended to be used in model validation and simulation. Model-level testing and debugging is covered in more detail by [OMG05b], while [OMG05c] contains specifications belonging to different application domains. Also, [OMG], which refers to current OMG technology adoption processes, contains more references regarding model validation and simulation.

Business process modeling comes from another perspective, but has the same final goal as our problem. For example, [Liu04] uses abstract logic trees to represent UML activity diagrams, allowing the study of their properties using graph algorithms. Also, [Hol04] and [Gru07] compare structured programming primitives with UML activity diagrams, studying graphical ways of representing structured processes.

Another industry initiative related to model specification, validation, and simulation is AGEDIS - Automated Generation and Execution of Test Suites for DIstributed Component-based Software [AGEDIS], a project ended in 2004.

3. THE SOLUTION: ComDeValCo

The proposed solution is **ComDeValCo** - a conceptual framework for Software **Com**ponents **De**finition, **Val**idation, and **Co**mposition. Its constituents are meant to cover both sub-processes discussed in 2.1: component development and component-based system development. This paper should be seen as a presentation of a solution for these two interconnected sub-processes and as a plan for the further developments of the framework.

The sub-process of component development starts with its definition, using an object-oriented modeling language, and graphical tools. The modeling language provides the necessary precision and consistency, and the use of graphical tools simplifies developer's work, which doesn't need to know the notations of modeling language. Once defined, component models are passed to a V & V (verification and validation) process, which is indended to check their correctness and to evaluate their performances. When a component passes V & V step, it is stored in a component repository, for later (re)use.

The sub-process of component-based system development takes the components already stored in repository and uses graphical tools, intended to: select components fulfilling a specific requirement, perform consistency checks regarding component assembly and include a component in the already existing architecture of the target system. When the assembly process is completed, and the target system is built, other tools will perform V & V, as well as performance evaluation operations on it.

Constituents of the conceptual framework are: the modeling language, the component repository and the toolset. Any model of a software component is described by means of a modeling language, programming language-independent, in which all modeling elements are objects. The component repository represents the persistent part of the framework and its goal is to store and retrieve valid component models. The toolset is aimed to help developers to define, check, and validate software components and systems, as well as to provide maintenance operations for the component repository.

The rest of this section gives a short description of the above constituents, illustrating their current status and intended further developments. More detalied descriptions will be given in separate papers.

3.1. **Modeling language.** *The software component model* is described by an object-oriented modeling language, all modeling elements being objects. The modeling language is independent from any object-oriented programming language and has the following features:

- all language elements (constructs) are objects, instances of classes defined at logical level, with no relationship to a concrete object-oriented programming language;
- language constructs cover both categories of software component discussed - the target software system - `Program` (the only executable) and proper software components (not executable by themselves, but ready to be assembled into a software system) - `Procedure, Function, Module, Class, Interface, Connector, Component;`
- there is a 1:1 relationship between the internal representation of the component model - seen as aggregated object - and its external representation on a persistent media, using various formats: XML, object serialization, etc.

A software component is fully defined, i.e. its model contains both component specification and component implementation. For example, `Program` components have three main constituents: the name, the state and the body. `Procedure` components, which are a specialization of `Program`, have as specific constituents their in, out, and in-out parameters (seen as lists). Component state contains all declared variables (names and values), while its body is a `CompoundStatement` (modeling construct defined using the *Composite* design pattern). Figure 1 is a UML class diagram showing some of the modeling elements already in place and their relationships. These elements constitute a UML metamodel, and can be used to build new UML profiles, in order to use existing CASE tools to build component models.

Statement subclasses are `SimpleStatement` and `CompoundStatement`, with `SimpleStatement` subclasses covering all control statements in an imperative

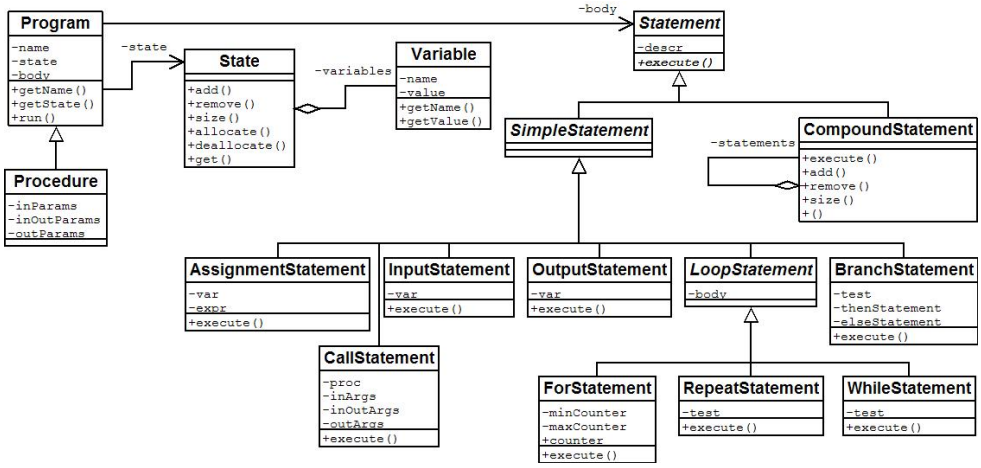


FIGURE 1. Class diagram (procedural paradigm, proof of concept)

programming language: `AssignmentStatement, CallStatement, InputStatement, OutputStatement, LoopStatement,`and `BranchStatement.`

The current version of the modeling language contains constructs belonging to the procedural paradigm, and is described in more detail in [Parv08].

3.2. **Component repository.** *Component repository* represents the persistent part of the framework, containing the models of all full validated components. Its development include the design of its data model, establishing indexing and searching criteria, as well as the format of representation. The ways of describing, indexing and searching considered will exploit XML-based protocols used to describe and discover Web services (WSDL and UDDI).

3.3. **The toolset.** *The toolset* is intended to automate many tasks and to assist developers in performing component definition and V & V tasks, maintenance of component repository, and component assembly. The tools are:

- DEFCOMP - component definition;
- VALCOMP - component V & V;
- REPCOMP - component repository management;
- DEFSYS, VALSYS - software system definition by component assembly, respectively V & V;
- SIMCOMP, SIMSYS - component and software system simulation;
- GENEXE - automatic generation of executable software systems.

From another perspective, the toolset will include some existing CASE tools, covering in part or in whole some of the functions above.

3.4. **Features of the proposed solution.** The proposed solution brings original elements in at least the following directions:

- the object model is precise and fine-grained, because all objects are rigorously defined, and the component behavior is described at statement level. The UML metamodel has no correspondent for modeling constructs more fine-grained than `Program, Procedure` and `Function;`
- the models are executable, verifiable, and evaluable because each component can be executed; moreover, one can use tools for checking validity and evaluating complexity;
- the models are independent of a specific (object-oriented) programming language and programming paradigm;
- modeling language is flexible and extensible; the dimensions of extensibility are: statement set, component definition, data type definition, and the component family;
- the statement set is extensible, by simply considering new (possible) primitive statements; as Figure 1 suggests, inheritance and composition are the main code reuse mechanisms used to define new statements;
- the component definition is extensible (we started with the simplest implementation of the component, using simple data types and expressions; next steps will include component specification, which needs more elaborate data types and expressions);
- data type definitions are also extensible (we started with simple data types; next steps will add structured types to the model, then object types - `Class`and `Interface`);
- the component family is extensible (we started with procedural paradigm components - `Program , Procedure` and `Function;`next steps will add: modular components - `Module -` object-oriented ones - `Class` and `Interface,`and, finally - `Component);`
- modeling language allows automatic code generation for components in a concrete programming language, according to Model Driven Architecture (MDA) specifications. One can define mappings from the modeling elements to specific constructs in a concrete programming language in a declarative way.

4. CONCLUSIONS AND FURTHER WORK

From methodological viewpoint, the main issue is to completely model all theoretical aspects in concrete objects - elements of modeling language. The modeling process is a gradual one, in order to keep its complexity under control. The main principle to be followed is to perform small steps; a step means here either implementing a new concept (transforming the concept into an object), or extending either a model element, a tool, or component repository

(by adding new features). One starts with simple objects and check after each step that things work.

Each modeling step include both theoretical/analytical activities - the abstract model of the concept - and practical/applicative ones - coding, testing and integrating it in the framework.

The intended use of the conceptual framework covers research, education, and industry applications. The competitive advantages are as follows:

- full compliance to the principles and methods of component-based software development, by covering both sub-processes - component and system development;
- high level of abstraction, assured by the independence of a specific programming language;
- ease of use: the model complexity is hidden behind a set of diagrams (model views), easy to define, understand, and manipulate;
- focus on reuse: the framework favors the definition and use of reusable software components by its constituent: component repository.

Other developments will include: maintenance of component repository by including new components, publishing the access interface to the component repository as a Web service, modeling at higher levels of abstraction, like workflows, business processes, application domain frameworks.

5. ACKNOWLEDGEMENTS

This work was supported by the grant ID_546, sponsored by NURC - Romanian National University Research Council (CNCSIS).

6. REFERENCES

[AGEDIS] *AGEDIS, Automated Generation and Execution of Test Suites for DIstributed Component-based Software*, http://www.agedis.de/index.shtml/.

[AW] W.van der Aalst, *PetriNets, tutorial* http://is.tm.tue.nl/staff/wvdaalst/petri_nets.htm.

[CL02] Crnkovic, I., Larsson, M., *Building Reliable Component-Based Software Sistems*, Prentice Hall International, Artech House Publishers, ISBN 1-58053-327-2, Available July 2002. http://www.idt.mdh.se/cbse-book/

[EG03] Eleftherakis, G., *Formal Verification of X-machine Models: Towards Formal Development of Computer-Based Sistems*, PhD, 2003.

[GM95] Glinz, M., *An Integrated Formal Model of Scenarios Based on Statecharts*. In Schfer, W. and Botella, P. (eds.): Software Engineering - ESEC'95. Berlin: Springer, 254-271.

[Gru07] Gruhn, V., Laue, R., *What business process modelers can learn from programmers*, Science of Computer Programming, 16 (2007), No. 1, 4-13.

[Hol04] Holl A., Valentin G., *Structured Business Process Modeling (SBPM)*, Information Systems Research in Scandinavia (IRIS 27), 2004.

[HHW01] Henderson, P., Howard Y., Walters, R.J., *A tool for evaluation of the Software Development Process*, Journal of Systems and Software, Vol 59, No 3, pp 355-362 (2001).

[HW99] Henderson, P., Walters, R.J., *System Design Validation Using Formal Models*, 10th IEEE International Workshop in Rapid System Prototyping, June 99, Clearwater, USA.

[HW00] Henderson, P., Walters, R.J., *Behavioural Analysis of Component-Based Sistems*, Declarative Sistems and Software Engineering Research Group, Department of Electronics and Computer Science, University of Southampton, Southampton, UK, 06 June 2000.

[HWC04] Henderson, P., Walters, R.J., Crouch, S., *Implementing Hierarchical Features in a Graphically Based Formal Modelling Language*, 28th Annual International Computer Software and Applications Conference (COMPSAC 2004), Hong Kong, 2004.

[Liu04] Ying Liu et al., *Business Process Modeling in Abstract Logic Tree*, IBM Research Report RC23444 (C0411-006) November 19, 2004.

[LL00] Jie Liu, Edward A. L., *Component-Based Hierarchical Modeling of Systems with Continuous and Discrete Dynamics*, Proc. of the 2000 IEEE International Symposium on Computer-Aided Control System Design Anchorage, Alaska, USA, September 25-27, 2000, pp 95-100.

[OMG] OMG, *Current OMG Technology Adoption Processes Under Way.* Pending Requests for Proposals, http://www.omg.org/public_schedule/.

[OMG05a] OMG, *Semantics of a Foundational Subset for Executable UML Models RFP*, http://www.omg.org/cgi-bin/doc?ad/2005-4-2/.

[OMG05b] OMG, *Model-level Testing and Debugging*, http://www.omg.org/cgi-bin/doc?ptc/2007-05-14/.

[OMG05c] OMG, *Catalog of OMG Domain Specifications*, http://www.omg.org/cgi-bin/doc?ad/2005-4-2/.

[Parv08] Pârv, B., Lazăr, I., Motogna, S., *ComDeValCo framework - the modeling language for procedural paradigm*, to be published in International Journal of Computers, Communications, and Control (IJCCC), vol. III, 2008.

[PJ02] Padberg, J., *Petri Net Modules*, Journal on Integrated Design and Process Technology vol. 6(4), pp. 121-137, 2002.

[WR02] Walters R. J., *A Graphically based language for constructing, executing and analysing models of software sistems*, PhD, 2002.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, 1, M. Kogălniceanu, Cluj-Napoca 400084, Romania

E-mail address: bparv,motogna,ilazar,czibula@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LII**, Number 2, 2007

GENERALIZED CYLINDERS SURFACES

L. ŢÂMBULEA AND I. GÂNSCA

Abstract. A generalized cylinder surface is generated by moving a 2D continuous curve along a 3D regular spine curve; the generating curve could be scaled and rotated around the spine curve. The shape of generalized cylinder surface induced by the scale functions and the angular velocity of rotation as well as some integral properties are discussed.

Keywords: Generalized cylinders; Generalized cylinder surfaces; Shape; Cusp; Rotation; Angular velocity; Integral properties

1. Motivation

Many industrial and artistic objects can be modelled with the aid of generalized cylinder surfaces. Theoretical and practical investigations in this area have been done by Lee and Requicha [7], Shani and Ballard [9], Bronsvoort and Warts [1], van der Helm, Ebell and Bronsvoort [6], Maekawa, Patrikalakis, Sakkalis and Yu [8], Gansca, Bronsvoort, Coman and Ţâmbulea [5] and others. The paper has the following structure. In Section 2 we recall the vector equation of a generalized cylinder surface. The shape of a generalized cylinder surface induced by the shapes of the scale functions is revealed in Section 3. Section 4 contains generalized cylinder surfaces generated by a scaled curve which makes rotations around the spine curve. Some integral properties of these twisted generalized cylinder surfaces and twisted generalized cylinders (objects) are given in Section 5.

2. Vector equation of a generalized cylinder surface

A generalized cylinder surface is generated by a continuous 2D curve which moves along a 3D regular spine (guide) curve, the plane of curve being perpendicular to the spine curve. The generating curve is referred to the local coordinate system X, Y, situated on the unit principal normal and binomial, respectively, vectors of the spine curve, see Fig.1.

Let us consider that the vector position of an arbitrary point of the spine curve is $\mathbf{C}(u), u \in [a, b]$ and the vector position of an arbitrary point of

Received by the editors: October 1, 2007.

2000 *Mathematics Subject Classification.* 65D18, 68U05, 68U07.

1998 *CR Categories and Descriptors.* J.6 [**Computer Applications**]: Computer-Aided Engineering – *Computer-aided design.*

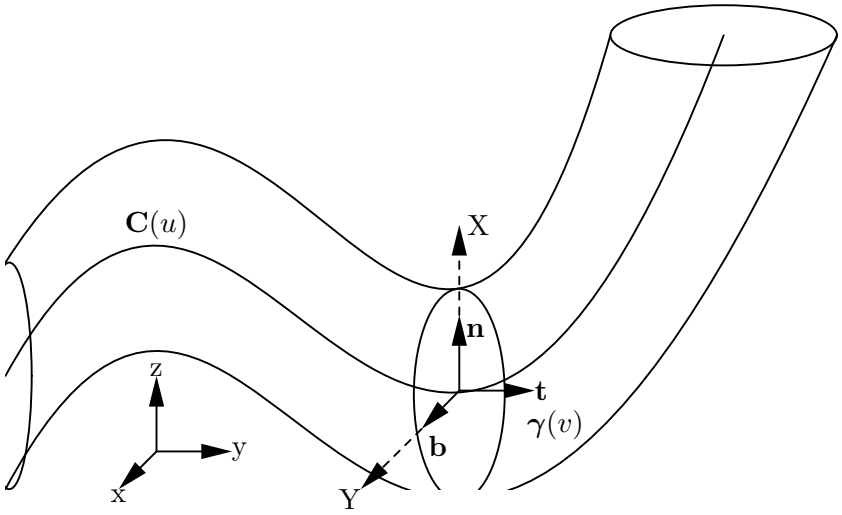


FIGURE 1. A generalized cylinder surface with an intermediary position of the generating curve $\boldsymbol{\gamma}(v)$.

the generating curve, with respect to the X, Y coordinate system, is $\boldsymbol{\gamma}(v) = (\varphi(v), \psi(v))^T$, $v \in [c, d]$. If the generating curve $\boldsymbol{\gamma}(v)$ is scaled into the directions of $\mathbf{n}(u)$ and $\mathbf{b}(u)$ with the aid of the positive and continuous scalar functions $s_1(u)$ and $s_2(u)$, respectively, $u \in [a, b]$, then, from Fig.1, one deduces the following vector equation of the generalized cylinder surface

$$\mathbf{\Gamma}(u, v) = \mathbf{C}(u) + s_1(u)\varphi(v)\mathbf{n}(u) + s_2(u)\psi(v)\mathbf{b}(u), \quad (u, v) \in D, \tag{1}$$

where $D = [a, b] \times [c, d]$.

The unit vectors $\mathbf{t}(u), \mathbf{n}(u)$ and $\mathbf{b}(u)$ form the Frenet trihedrom and are given by the formulas

$$\mathbf{t}(u) = \frac{\mathbf{C}'(u)}{|\mathbf{C}'(u)|}, \quad \mathbf{b}(u) = \frac{\mathbf{C}'(u) \times \mathbf{C}''(u)}{|\mathbf{C}'(u) \times \mathbf{C}''(u)|}, \quad \text{and} \quad \mathbf{n}(u) = \mathbf{b}(u) \times \mathbf{t}(u). \tag{2}$$

Remark 1. From (2) results that the vector function $\mathbf{C}(u)$ must be of the second order continuity class.

In what follows we will firstly focus on the $\mathbf{\Gamma}(u, v)$ surface shape control with the aid of scale functions $s_1(u)$ and $s_2(u)$. Next we will deduce the vector equation of the generalized cylinder surface resulted by rotation of the scaled generating curve $\boldsymbol{\gamma}_s(v; u) = (s_1(u)\varphi(v), s_2(u)\psi(v))^T$ around the spine curve $\mathbf{C}(u)$, with a variable angular velocity.

3. Shape of $\mathbf{\Gamma}(u,v)$ induced by the shapes of $s_1(u)$ and $s_2(u)$

Information about the shape of $\mathbf{\Gamma}(u,v)$ one obtains analysing its coordinate lines $\mathbf{\Gamma}(u=\text{const},v), v\in[c,d]$ and $\mathbf{\Gamma}(u,v=\text{const}), u\in[a,b]$, respectively. From the vector equation (1) we observe that the coordinate line $\mathbf{\Gamma}(u,v=\text{const}), u\in[a,b]$ is, in fact, the scalled generating curve $\boldsymbol{\gamma}_s(v;u), u=\text{const}$ relative to the $xOyz$ coordinate system.

Important information about the shape of coordinate line $\mathbf{\Gamma}(u,v=\text{const})$, $u\in[a,b]$ results from its tangent vector $\mathbf{\Gamma}_u(u,v)$. From (1), taking into account the Frenet-Serret formulas,

$$\begin{aligned}\mathbf{t}'(u) &= \mathcal{K}(u)|\mathbf{C}'(u)|\mathbf{n}(u),\\ \mathbf{n}'(u) &= |\mathbf{C}'(u)|[-\mathcal{K}(u)\mathbf{t}(u)+\mathcal{T}(u)\mathbf{b}(u)],\\ \mathbf{b}'(u) &= -\mathcal{T}(u)|\mathbf{C}'(u)|\mathbf{n}(u),\end{aligned}$$

one obtains

$$\begin{aligned}\mathbf{\Gamma}_u(u,v) &= |\mathbf{C}'(u)|\left[1-\mathcal{K}(u)s_1(u)\varphi(v)\right]\mathbf{t}(u)+\\ &+\left[s_1'(u)\varphi(v)-\mathcal{T}(u)s_2(u)|\mathbf{C}'(u)|\psi(v)\right]\mathbf{n}(u)+\\ &+\left[s_2'(u)\psi(v)+\mathcal{T}(u)s_1(u)|\mathbf{C}'(u)|\varphi(v)\right]\mathbf{b}(u),\end{aligned}\tag{3}$$

where $\mathcal{K}(u)>0$ and $\mathcal{T}(u)$ are the curvature and torsion, respectively, of the spine curve $\mathbf{C}(u)$, and are given by the formulas

$$\mathcal{K}(u)=\frac{|\mathbf{C}'(u)\times\mathbf{C}''(u)|}{|\mathbf{C}'(u)|^3}\quad\text{and}\quad\mathcal{T}(u)=\frac{(\mathbf{C}'(u)\times\mathbf{C}''(u))\,\mathbf{C}'''(u)}{|\mathbf{C}'(u)\times\mathbf{C}''(u)|^2}.$$

Next we recall

Definition. An interior point of a curve $\mathbf{g}(t), t\in I, I\subset\Re$, say $\mathbf{g}(t_0)$, is called a cusp of $\mathbf{g}(t)$ if

$$\lim_{t\to t_0^-}\mathbf{g}'(t)=-\lim_{t\to t_0^+}\mathbf{g}'(t).\tag{4}$$

Remark 2. In the special case when $\mathbf{g}(t)=(t,f(t))^T, t\in I$, the interior point $\mathbf{g}(t_0)$ is a cusp of $\mathbf{g}(t)$ if and only if

$$\lim_{t\to t_0^-}f(t)=-\lim_{t\to t_0^+}f(t)=\infty,(or-\infty).\tag{5}$$

With other words, the interior point $t_0\in I$ is a cusp of the scalar function $f(t)$, if and only if (5) holds.

Regarding to an arbitrary coordinate line $\mathbf{\Gamma}(u,v_0), v_0\in[c,d]$ we will prove the following

Proposition 1. *If $s_1(u)$ and $s_2(u)$ have cusps for $u = u_0$ and*

$$\lim_{u \to u_0} = \frac{s_1^{'}(u)}{s_2^{'}(u)} = m, \tag{6}$$

then the coordinate line $\mathbf{\Gamma}(u, v_0)$ does have cusp for $u = u_0$ in the direction of the vector

$$\varphi(v_0)\mathbf{n}(u_0) + m\psi(v_0)\mathbf{b}(u_0), \tag{7}$$

provided that $|\boldsymbol{\gamma}(v_0)| \neq 0$.

Proof. Let us consider a vicinity of u_0, say V_0, such that $s_1^{'}(u) \neq 0$, if $u \in V_0$. Similar reasoning one does if $s_2^{'}(u) \neq 0, u \in V_0$. From (3), if $u \in V_0$, we can write

$$\begin{aligned}\mathbf{\Gamma}_u(u, v_0) = \; & s_1^{'}(u) \Big\{ \varphi(v_0)\mathbf{n}(u) + \tfrac{s_2^{'}(u)}{s_1^{'}(u)}\psi(v_0)\mathbf{b}(u) + \\ & + \tfrac{|\mathbf{C}^{'}(u)|}{s_1^{'}(u)} \left[(1 - \mathcal{K}(u)s_1(u)\psi(v_0))\, \mathbf{t}(u) - \mathcal{T}(u)s_2(u)\psi(v_0)\mathbf{n}(u) + \right. \\ & \left. + \mathcal{T}(u)s_1(u)\varphi(v_0)\mathbf{b}(u) \right] \Big\}.\end{aligned}$$

If the scalar functions s_1 and s_2 do have cusps in u_0, then, taking into account Remark 2, (5) and (6) results

$$\lim_{u \to u_0^-} \mathbf{\Gamma}_u(u, v_0) = - \lim_{u \to u_0^+} \mathbf{\Gamma}_u(u, v_0),$$

and the direction of $\mathbf{\Gamma}_u(u, v_0)$, when $u \to u_0$, approaches to the direction of vector given at (7).

Remark 3. If $\varphi(v_0) = 0$ and $\psi(v_0) \neq 0$, then from (7) results that the cusp of coordinate line $\mathbf{\Gamma}(u, v_0)$, in $u = u_0$, is in the direction of $\mathbf{b}(u_0)$. Analogously, if $\varphi(v_0) \neq 0$ and $\psi(v_0) = 0$, then the cusp of $\mathbf{\Gamma}(u, v_0)$, in $u = u_0$, is in the direction of $\mathbf{n}(u_0)$.

Figures 2 (a), (b) and Figure 3 illustrate this theoretical part. The generalized cylinder surface from Figure 3 has the spine curve

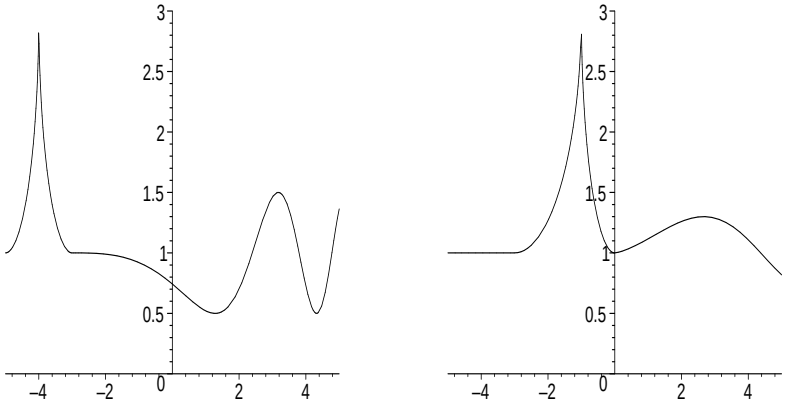
$$\mathbf{C}(u, a^*) = (u, a^* u^2, 0)^T, u \in [-5, 5], \quad a^* = 0.05,$$

and the generating curve with

$$\varphi(v) = cos(v), \; \psi(v) = sin(v),$$

$$s_1(u) = s((u+5)/10, 0.1, 1, 0.1, 2, 0.5, 20, 3, \frac{\pi}{2}),$$

$$s_2(u) = s((u+5)/10, 0.4, 1, 0.2, 0.1, 2, -0.3, 10, 1.4, \frac{\pi}{2}), \quad u \in [-5, 5],$$



(a) (b)

FIGURE 2. Cusps of $s_1(u)$ and $s_2(u)$.

where $s(t, u_0, a, b, c, d, e, p, q, r) =$

$$\begin{cases} a, & t \in [0, u_0 - b], \\ a + d - \frac{d}{b}\sqrt{b^2 - (t - u_0 + b)^2}, & t \in (u_0 - b, u_0], \\ a + d - \frac{d}{c}\sqrt{c^2 - (t - u_0 - c)^2}, & t \in (u_0, u_0 + c], \\ a + e \cdot cos(p(t - u_0 - c)^q + r), & t \in (u_0 + c, 1]. \end{cases}$$

Figures 2 (a) and (b) present the cusps of $s_1(u)$ and $s_2(u)$ respectively, which determine the cusps to the coordinate lines $\mathbf{\Gamma}(u, v = \text{const})$, shown in Figure 3.

4. Rotation of the scaled generating curve

In what follows we consider that the scaled generating curve

$$\boldsymbol{\gamma}_s(v; u) = (s_1(u)\varphi(v), s_2(u)\psi(v))^T, \ v \in [c, d],$$

makes rotations around the spine curve $\mathbf{C}(u)$, with angular velocity $\omega = \omega(u), u \in [a, b]$, while it moves along $\mathbf{C}(u)$.

The angle of rotation between the initial and an intermediary position of $\boldsymbol{\gamma}_s(v; u)$, if $\omega(u) \geq 0$ (or $\omega(u) \leq 0$) is

$$\alpha(u) = \int_a^u \omega(t)dt. \tag{8}$$

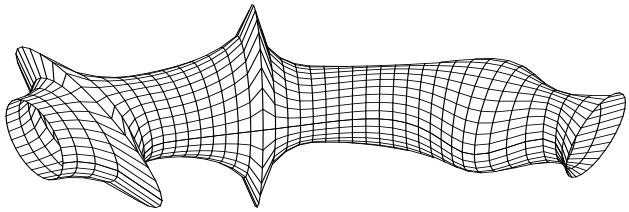


FIGURE 3. Cusps of coordinate lines $\mathbf{\Gamma}(u, v = \text{const})$.

The twisted generalized cylinder surface, in this case, is represented by the following vector equation

$$\begin{aligned} &\mathbf{\Gamma}_1(u,v) = \mathbf{C}(u) + \\ &+ \left[s_1(u)\varphi(v)cos(\alpha(u)+\alpha_0) + s_2(u)\psi(v)sin(\alpha(u)+\alpha_0)\right]\mathbf{n}(u) + \\ &+ \left[-s_1(u)\varphi(v)sin(\alpha(u)+\alpha_0) + s_2(u)\psi(v)cos(\alpha(u)+\alpha_0)\right]\mathbf{b}(u), \end{aligned} \tag{9}$$

$(u, v) \in D$; α_0 is the angle of $\boldsymbol{\gamma}_s(v, u)$-rotation around $\mathbf{C}(u)$, before the starting generation of $\mathbf{\Gamma}_1(u, v)$.

The number of rotations, if $\omega(u) \geq 0$ (or $\omega \leq 0$), when $u \in [0, u^*]$ is

$$n^* = \frac{|\alpha(u^*)|}{2\pi}, \tag{10}$$

where $\alpha(u^*)$ is given by the formula (8).

For example, if the angular velocity $\omega = k|u - u_0|^\beta, u \in [0, 1]$, where the parameter $u_0 \in [0, 1]$ and α, β are real and positive numbers, using formula (10), one obtains

$$\alpha(u) = \begin{cases} \frac{k}{\beta+1}\left[u_0^{\beta+1} - (u_0 - u)^{\beta+1}\right], & 0 \leq u \leq u_0, \\ \\ \frac{k}{\beta+1}\left[u_0^{\beta+1} + (u - u_0)^{\beta+1}\right], & u_0 \leq u \leq 1. \end{cases} \tag{11}$$

Denoting by ν the rotations number of curve $\boldsymbol{\gamma}_s(v; u)$ around the spine curve $\mathbf{C}(u)$, when $u \in [0, 1]$ then, using formulas (10) and (11), results

$$\nu = \frac{k}{2(\beta+1)\pi}\left[u_0^{\beta+1} + (1 - u_0)^{\beta+1}\right].$$

Therefore, if one wants ν rotations then, the angular velocity must be

$$\omega(u) = \frac{2\pi\nu(\beta+1)}{u_0^{\beta+1} + (1-u_0)^{\beta+1}} |u-u_0|^{\beta}, \quad u \in [0,1]. \tag{12}$$

Corresponding to this angular velocity, the angle of rotation is

$$\alpha(u) = \begin{cases} \frac{2\pi\nu}{u_0^{\beta+1}+(1-u_0)^{\beta+1}} \left[u_0^{\beta+1} - (u_0-u)^{\beta+1}\right], & 0 \le u \le u_0, \\ \frac{2\pi\nu}{u_0^{\beta+1}+(1-u_0)^{\beta+1}} \left[u_0^{\beta+1} + (u-u_0)^{\beta+1}\right], & u_0 \le u \le 1. \end{cases} \tag{13}$$

In Figs. 4 and 5 are presented two particular twisted cylinder surfaces $\mathbf{\Gamma}_1(u,v)$ of equation (9), for which

$$\mathbf{C}(u) = \sum_{i=0}^{6} \mathbf{b}_i B_i^6(u), \ u \in [0,1], \tag{14}$$

where $B_i^6(u) = \binom{6}{i}(1-u)^{6-i}u^i$, $\mathbf{b}_0 = (3,0,9)$, $\mathbf{b}_1 = (8,1,5)$, $\mathbf{b}_2 = (11,9,2)$, $\mathbf{b}_3 = (13,25,0)$, $\mathbf{b}_4 = (7,29,2)$, $\mathbf{b}_5 = (3,26,6)$, $\mathbf{b}_6 = (0,23,11)$ and the angle of rotation is of the form (13). Fig.4 corresponds to

$$\begin{array}{l} \varphi(v) = cos(v), \ \psi(v) = sin(v), \ v \in [0,2\pi], \\ s_1(u) = 1 + 0.5sin(12u), \ s_2(u) = 1/s_1(u), \ u \in [0,1], \\ \nu = 1.5, \ \beta = 0.01, \ u_0 = 0.1 \end{array} \tag{15}$$

and the defining elements of Fig.5 are

$$\begin{array}{l} \varphi(v) = 4cos^3(v), \ \psi(v) = 4sin^3(v), \ v \in [0,2\pi], \\ s_1(u) = s_2(u) = 1, \ u \in [0,1], \\ \nu = 0.5, \ \beta = 0.2, \ u_0 = 0.3. \end{array} \tag{16}$$

5. Some integral properties

Firstly we recall that a generalized cylinder (solid) is the body bounded by a generalized cylinder surface and two planes perpendicular to the spline curve in its initial and final points. Next we will give some formulas regarding twisted generalized cylinder surfaces and twisted generalized cylinder, without self-intersections.

Throughout this section we will make use of the

Remark 4. Rotations and other maps, characterized by orthonormal matrices, leave lengths, areas and angles unchanged (Farin, 1990).

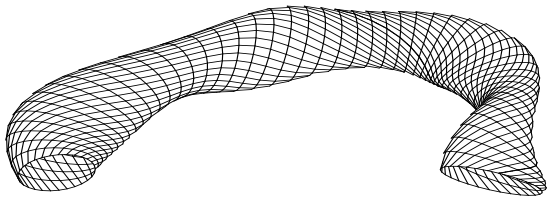


FIGURE 4. Twisted generalized cylinder surface $\mathbf{\Gamma}_1(u, v)$ corresponding to (14) and (15).

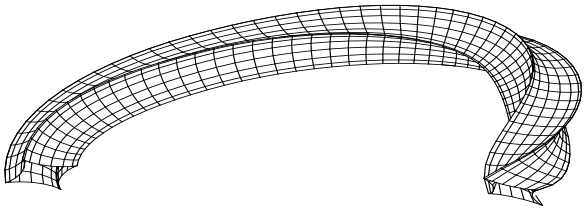


FIGURE 5. Twisted generalized cylinder surface $\mathbf{\Gamma}_1(u, v)$ corresponding to (14) and (16).

5.1. **Gravity center line and area of $\mathbf{\Gamma}_1(u, v)$.** Gravity center line of $\mathbf{\Gamma}_1(u, v)$ is evidently the locus of the gravity centers of the generating curve

$$\boldsymbol{\gamma}_r(v; u) = (x(v; u), y(v; u))^T ,$$

where

$x(v;u) = s_1(u)\varphi(v)cos(\alpha(u)+\alpha_0) + s_2(u)\psi(v)sin(\alpha(u)+\alpha_0),$
$y(v;u) = -s_1(u)\varphi(v)sin(\alpha(u)+\alpha_0) + s_2(u)\psi(v)cos(\alpha(u)+\alpha_0),\ v\in[c,d].$

Let $G\left(X_g(u), Y_g(u)\right)$ be the gravity center of the curve

$$\boldsymbol{\gamma}_s(v;u) = \left(s_1(u)\varphi(v), s_2(u)\psi(v)\right)^T.$$

The coordinates $X_g(u)$ and $Y_g(u)$ are given in our paper [5], by formulas (13) and (17). Denoting by $G_r\left(X_g^r(u), Y_g^r(u)\right)$ the gravity centre of the curve $\boldsymbol{\gamma}_r(v;u)$, in virtue of the Remark 4 we have,

$$\begin{array}{rl} X_g^r(u) = & X_g(u)cos\left(\alpha(u)+\alpha_0\right) + Y_g sin\left(\alpha(u)+\alpha_0\right), \\ Y_g^r(u) = & -X_g(u)sin\left(\alpha(u)+\alpha_0\right) + Y_g cos\left(\alpha(u)+\alpha_0\right), \end{array} \tag{17}$$

Locating G_r with respect to the $xOyz$ coordinate system we have

$$\mathbf{G}_r(u) = \mathbf{C}(u) + X_g^r(u)\mathbf{n}(u) + Y_g^r(u)\mathbf{b}(u);\ u\in[a,b]. \tag{18}$$

If $\mathbf{C}(u) = \left(x(u), y(u), z(u)\right)^T$, $\mathbf{n}(u) = (a_1(u), a_2(u), a_3(u))$ and $\mathbf{b}(u) = (b_1(u), b_2(u), b_3(u))$, then, from (18) results

$$\mathbf{G}_r(u) = \begin{pmatrix} x_g^r(u) \\ y_g^r(u) \\ z_g^r(u) \end{pmatrix} = \begin{pmatrix} x(u) + X_g^r(u)a_1(u) + Y_g^r(u)a_2(u) \\ y(u) + X_g^r(u)b_1(u) + Y_g^r(u)b_2(u) \\ z(u) + X_g^r(u)c_1(u) + Y_g^r(u)c_2(u) \end{pmatrix}; \tag{19}$$

where $u\in[a,b]$, $X_g^r(u)$ and $Y_g^r(u)$ being given by (17).

Denoting by CG_r the locus of $G_r(u)$, when $u\in[a,b]$ results

Proposition 2. *The gravity center line CG_r of the surface $\mathbf{\Gamma}_1(u,v)$ has the parametric equations (19).*

If S_1 is the area of $\mathbf{\Gamma}_1(u,v)$, then, in virtue of the Remark 4 and formula (23) from our paper (2002) results

$$S_1 = \int_{CG_r}\mathcal{L}(u)ds = \int_a^b \sqrt{\left(x_g^r(u)\right)'^2 + \left(y_g^r(u)\right)'^2 + \left(z_g^r(u)\right)'^2}du. \tag{20}$$

Next we denote by $\mathcal{V}_1$ the twisted generalized cylinder bounded by $\mathbf{\Gamma}_1(u,v)$ and the perpendicular planes to the spine curve in its initial and final points.

5.2. **Gravity center line and volume of $\mathcal{V}_1$.** In our paper [5] we have established (formulas (23) and (24)) that if $G^0\left(X_G^0, Y_G^0\right)$ is the gravity centre of the domain bounded by the closed curve $\boldsymbol{\gamma}(v) = \left(\varphi(v), \psi(v)\right)^T, v\in[c,d], \boldsymbol{\gamma}(c) = \boldsymbol{\gamma}(d)$, then the gravity centre of the domain bounded by the curve $\boldsymbol{\gamma}_s(v;u) = \left(s_1(u)\varphi(v), s_2(u)\psi(v)\right)^T, v\in[c,d]$ is $G^*\left(X_G^*, Y_G^*\right)$, where

$$\begin{array}{l} X_g^*(u) = s_1(u)X_G^0, \\ Y_g^*(u) = s_2(u)Y_G^0. \end{array} \tag{21}$$

Now, if $\boldsymbol{\gamma}_s(v;u)$ makes rotations around the spine curve, with the angular velocity $\omega(u)$, then its gravity centre becomes $G_r^*\left(X_r^*, Y_r^*\right)$, where

$$(22)\quad \begin{array}{l} X_r^*(u) = s_1(u)X_G^0 cos\left(\alpha(u)+\alpha_0\right) + s_2(u)Y_G^0 sin\left(\alpha(u)+\alpha_0\right), \\ Y_r^*(u) = -s_1(u)X_G^0 sin\left(\alpha(u)+\alpha_0\right) + s_2(u)Y_G^0 cos\left(\alpha(u)+\alpha_0\right), \end{array}$$

where $\alpha(u)$ is given by the formula (8).

Denoting by CG_r^* the locus of $G_r^*\left(X_r^*(u), Y_r^*(u)\right)$ when $u \in [a,b]$ and proceedings as before, we can state

Proposition 3. *The gravity center line CG_r of the generalized cylinder $\mathcal{V}_1$ has the following parametric equations*

$$(23)\quad \mathbf{G}_r^*(u) = \begin{pmatrix} x_r^*(u) \\ y_r^*(u) \\ z_r^*(u) \end{pmatrix} = \begin{pmatrix} x(u) + X_r^*(u)a_1(u) + Y_r^*(u)a_2(u) \\ y(u) + X_r^*(u)b_1(u) + Y_r^*(u)b_2(u) \\ z(u) + X_r(^*u)c_1(u) + Y_r^*(u)c_2(u) \end{pmatrix};$$

where $u \in [a,b]$.

With regard to the volume of $\mathcal{V}_1$, similary to the formula (26) from our paper [5], we have

$$(24)\quad \begin{array}{ll} \mathcal{V}_1 & = \int_{CG_r^*} \mathcal{A}(u)du = \\ & = \mathcal{A}_0 \int_a^b s_1(u)s_2(u)\sqrt{\left(x_r^*(u)\right)'^2 + \left(y_r^*(u)\right)'^2 + \left(z_r^*(u)\right)'^2}du. \end{array}$$

References

[1] Bronsvoort W. F. and Waarts J.J. (1992), A method for converting the surface of a generalized cylinder into a B-spline surface. Computers & Graphics, 16(2), 175–178.

[2] Bronsvoort W. F. (1992), A surface-scanning algorithm for displaying generalized cylinders. Visual Computer, 8(3), 162–170.

[3] de Voogt E., van der Helm A., Bronsvoort W. F. (2000), Ray tracing deformed generalized cylinders. Visual Computers, 16, 197–207.

[4] Farin G. (1990), *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, New York, Second Edition.

[5] Gansca I., Bronsvoort W. F., Coman Gh., Tambulea L. (2002), Self-intersection avoidance and integral properties of generalized cylinders. Computer Aided Geometric Design, 19, 695–707.

[6] van der Helm A., Ebell P. and Bronsvoort W. F. (1998), Modelling mollusc shells with generalized cylinders. Computers & Graphics 22(4), 505–513.

[7] Lee Y.T., Requicha A.A.G. (1982), Algorithms for computing the volume and other integral properties of solids. I. Known methods and open issues. Communications of the ACM 25(9), 635–641.

[8] Maekawa T., Patrikalakis N. M., Sakkalis T. and Yu G. (1998), Analysis and applications of pipe surfaces. Computer Aided Geometric Design, 15(5), 437–458.

[9] Shani U., Ballard D.H. (1984), Splines as embeddings for generalized cylinders. Computers Vision, Graphics and Image Processing, 27, 129–156.

Babes-Bolyai University, Faculty of Mathematics and Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

E-mail address: leon@cs.ubbcluj.ro

16/4 Muscel St., Cluj-Napoca, Romania

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LII**, Number 2, 2007

INDEXING THE EVOLUTION OF MOVING OBJECTS WITHIN A 2D SPACE USING THE BRICKR STRUCTURES

ANDREEA SABAU

Abstract. A growing number of applications manage mobile objects. The storage and the organization within databases of data describing the evolution of these objects is an open challenge. Data must be managed in efficient structures with respect to both the storage space consumed and the data access through these structures. An indexing method that organizes the evolutions of spatial objects within a 2D space is proposed in this paper. The Dynamic-BrickR access method uses two structures: an underlying permanent R*-Tree structure, and an in-memory dynamic space grid structure, that it used for building the terminal nodes to feed the R*-Tree. Experiments show significant improvements of the Dynamic-BrickR method over the R*-Tree index, regarding the dead space and the overlapping volumes. The Dynamic-BrickR inherits from the R*-Tree the capability to be used in answering spatial, temporal and spatio-temporal queries.

1. Introduction

The organization and management of spatio-temporal objects required lately a lot of attention. The growing interest in this area is justified by the proliferation of a large range of applications that can benefit of efficient management methods for spatio-temporal data. Communication and localization in mobile objects networks is one such application domain.

The spatio-temporal data represents the evolution of spatial objects in time. A spatial attribute with discretely or continuously evolving values may represent the shape and / or the location of an object. For example, land parcels positions and extents evolve discretely in time; while the cars on a road are continuously changing their position, but not the shape. The most

Received by the editors: November 1, 2007.

2000 *Mathematics Subject Classification.* 68P05, 68P20.

1998 *CR Categories and Descriptors.* H.2.2 [**Information Systems**]: Database Management – *Physical Design*; H.2.4 [**Information Systems**]: Database Management – *Systems*; H.2.8 [**Information Systems**]: Database Management – *Database Applications* .

significant challenge in the management of spatio-temporal objects is to efficiently organize information about the continuous change in time of their spatial features values

The extended version of the Dynamic-BrickR access method for indexing 3D spatio-temporal data is presented in this paper. The described method is based on the previous work presented in [9, 10]. Dynamic-BrickR organizes the continuous evolutions of spatial objects within a 2D spatial domain. These objects shape is not relevant and they are represented as points (or no extent) objects. A real-world example of such objects is that of the cars on a network of roads that are moving with different speeds, in any way.

Related Work. Many researchers have worked on organizing spatio-temporal objects in index structures. These structures may be classified as structures that index: past data, present and past data or data about present and future.

Several index structures, such as STR-Tree [7] and SETI [4] organize past information. The STR-Tree is an R-Tree like structure that attempts to achieve trajectory preservation for each object by storing its trajectory segments in the same tree node. SETI divides the spatial domain into a static partition and the data corresponding to one cell of the partition are organized into an R-Tree.

There are many spatio-temporal access methods that manage present (and past) data. The 2-3 TR-Tree [1] is an index structure that contains two R-Trees: one tree for points that represent present data and another R-Tree for the trajectories from the past; the search might have to consult one or both R-Trees. The MR-Tree [13] and the HR+-Tree [11] are overlapping R-Trees, where a node may have one or more parents. The LUR-Tree [5] indexes only current positions of objects, so historical queries are not supported.

PR-Tree [3], STAR-Tree [8], and TPR*-Tree [12] belong to a class of spatio-temporal access methods, which manage present data and data for prediction of future movement.

The paper is organized as follows. The Dynamic-BrickR access method, its extended structures and the management of spatio-temporal data are described in Section 2. Section 3 presents the comparative results for Dynamic-BrickR, BrickR and R*-Tree [2] methods in organizing and querying spatio-temporal data. The paper ends with conclusions and future work.

2. The 2D Dynamic-BrickR Spatio-Temporal Access Method

This section presents the structures and the involved management operations corresponding to the Dynamic-BrickR access method. These structures

represent the upgrade applied on the 1D Dynamic BrickR [9, 10] access method in order to index the continuous evolutions of mobile objects within a 2D space.

The Dynamic-BrickR access method uses a temporary structure and a permanent structure in order to index efficiently the recently received and past data. The permanent physical structure is designed as an R*-Tree and it is called DBR-Tree. The temporary structure, called DBR-Grid, is an in-memory space grid, that it used for building terminal tree nodes to feed the R*-Tree. The dimensionality of the grid corresponds to the space dimensionality of the indexed objects. In the particular studied case, the grid is 2D. The grid has a dynamic evolution, as the strips it is composed of are split or merged to best adjust to the objects evolution in time. The name of the Dynamic-BrickR access method originates in the visual aspect the grid gives to the indexed space, which resembles a brick wall; it also reflects the grid dynamic behavior and the fact it uses an R*-Tree permanent structure.

As it was mentioned before, the structures of the Dynamic-BrickR indexing method organizes data corresponding to the spatial evolutions of objects in time. It is considered that the objects shape is not relevant (it is not changing in time and it is not significant). Therefore, the objects are modeled as points within the 2D spatial domain.

The spatial domain is considered to be (relatively) constant in time, without affecting the organization rules and the generality of the indexing method. If a set of objects moved beyond the initially spatial domain borders, the working space would be extended in a straightforward fashion. In order to facilitate the generation and the visualization of data, the spatial domain is considered to be $[0, L] \times [0, L]$. The temporal domain is considered to be isomorphic to the set of real numbers. Therefore, the temporal domain is treated as an auxiliary domain to the spatial one.

Regarding the indexed spatio-temporal data, usually the mobile objects are moving within the spatial domain with a variable speed. In order to facilitate the representation of their trajectories, the speed of an object is considered to be constant during a certain time interval. Therefore, the trajectory on that time interval is approximated by a linear function of time and it is represented geometrically as a 3D line segment (also called trajectory segment). The set of trajectory segments corresponding to a mobile object represents the trajectory of that object, or its spatial evolution in time.

Theoretically, there is no restriction on the manner the mobile objects are sending the data to the system. The mobile objects can be equipped with some GPS devices and can send the positioning information with regularity or when the value of some parameter of movement (the direction and / or the speed) is changed. Furthermore, the received information can be a spatio-temporal point (the new position at a certain time instant) or a spatio-temporal line

segment (the trajectory segment corresponding to a certain time interval). However, at physical level, the received information is stored as linear functions of time (as 3D spatio-temporal line segments).

The Dynamic-BrickR access method contains two sub-structures [9, 10]:

- A temporary structure,
- A persistent structure.

The permanent structure in the Dynamic-BrickR method, called the DBR-Tree, is essentially an R*-Tree structure used to index spatio-temporal data (3D trajectory segments) from the remote past. It is assumed to be stored in secondary memory; therefore the paginated storage of the nodes is facilitated.

The temporary structure, called the DBR-Grid, is a grid structure that indexes the newest spatio-temporal data received by the system. This structure is stored in the main memory, having the advantage of a short data access time and efficient operations.

Regarding the receiving of data, it is natural to consider that data about an object trajectory arrives in ascending order of the timestamps of the trajectory segments end points.

As it was described in [9, 10], the temporary structure is designed to create an extra "thinking" moment of how to group trajectory segments into MBRs. The main idea is not to insert a segment into the main R*-Tree structure as soon as it is received by the system. The most recent segments are kept in memory and clustered in in-memory tree nodes, which will be entirely inserted in the tree afterwards. This way, there are almost void chances to get overlapping areas between the resulting leaf nodes MBRs and this conducts to smaller overlapping at superior levels.

As in the 1D Dynamic-BrickR structures [9, 10], the DBR-Grid is obtained by partitioning the 3D spatio-temporal domain using two sets of hyper-planes parallel with the temporal axis (Ot). One set's hyper-planes are also parallel to the Oy axis, and perpendicular on the Ox axis, and the hyper-planes of the other set are parallel with the Ox axis, and perpendicular on the Oy axis. The initial number of hyper-planes and their positions are set at the beginning of the grid's construction. The partitioning of the spatial domain is accomplished by using (initially) equidistant hyper-planes on each of the two spatial axes. The pieces of the DBR-Grid are 3D orthogonal polyedra and, as in the case of 1D Dynamic-BrickR structures, these elements are called strips.

It is also initially considered that the number of hyper-planes on the Ox axis (nbx) is equal to the number of divisions on the Oy axis (nby). This number is noted here nb. Therefore, the initial number of grid strips is nb^2. Let the DBR-Grid obtained in this manner be built by the strips B_{ij}, i, j:=1..nb. The length of the spatial projection of a strip on the Ox and Oy axis is initially

$\Delta b = lS/nb$. The coordinates of the partitioning hyper-planes are given by the two arrays XD = (xd_0, xd_1, ..., xd_{nb}), and YD = (yd_0, yd_1, ..., yd_{nb}), respectively, where $xd_k = k * \Delta b$ and $yd_k = k * \Delta b$, k:=0..*nb*. Let *nSegm_ij* be the number of trajectory segments contained within a grid strip B_{ij}, and SB_k^{ij} the segments included in B_{ij}, i, j:=1..*nb*, k:=1..*nSegm_ij*. The initial configuration of the DBR-Grid is considered in Fig. 1.

According to these observations, the partitioning hyper-planes initially determine on the spatial domain (xOy) a 2D rectangular regular grid (see the initial partitioning of the spatial domain xOy in Fig. 1). Because of the grid's dynamicity, the spatial partitioning may become non-regular by performing some division and / or merge operations (see the Examples 1 and 2). Furthermore, the numbers of partitioning hyper-planes on the two spatial axes may differ in time ($nbx \neq nby$).

As in the 1D case, the trajectory segments are first inserted in the DBR-Grid. One segment may be clipped according to the grid strips it intersects, and then the resulted sub-segments are inserted in the corresponding strips of the DBR-Grid. In other words, if a segment intersects two or more strips, it is divided into pieces, each piece being totally enclosed within a strip. Fig. 1 shows the result of a fragmentation operation performed on a trajectory segment, having as result three sub-segments.

The MBR of a 3D strip B_{ij} is given by the points (x_1^{MBR}, y_1^{MBR}, t_1^{MBR}) and (x_2^{MBR}, y_2^{MBR}, t_2^{MBR}), where $x_1^{MBR} = xd_{i-1}$, $x_2^{MBR} = xd_i$, $y_1^{MBR} = yd_{j-1}$, $y_2^{MBR} = yd_j$, $t_1^{MBR} = \min\{SB_k^{ij}.t_1|$ k:=1..*nSegm_ij*$\}$, and $t_2^{MBR} = \max\{SB_k^{ij}.t_2|$ k:=1..*nSegm_ij*$\}$.

The modification of the `Insert_segment` and `Cut_MBR` algorithms corresponding to the 2D DBR-Grid so that to manage 3D trajectory segments is straightforward. The major difference between the 2D DBR-Grid and the 3D DBR-Grid is found at operational level (the management of the partitioning hyper-planes, the division and merge operations). Furthermore, an object *B* of type *Strip* [10] is characterized by its spatial borders, given by *B.xmin*, *B.xmax*, *B.ymin*, *B.ymax*, and an object *g* of type DBR-Grid is a 2D array with elements of type *Strip*. The structure of *g* is determined by the XD and YD arrays: *nbx* is the number of rows of *g*, *nby* gives the number of columns of *g*, and *g*[*i*][*j*] represents the strip that contains a set of SO segments, so that $SO.x_1$, $SO.x_2 \in [xd_{i-1}, xd_i)$ and $SO.y_1$, $SO.y_2 \in [yd_{j-1}, yd_j)$, i:=0..*nbx*, j:=0..*nby*.

It can be observed that the partioning determined by the object *g* on the spatial domain does not concur with the physical delimitation of the strips. Therefore, it have to be mentioned that *g* determines a *virtual partitioning* of the spatial domain, and the borders of the strips (the hyper-planes of XD

and YD of which coordinates are found within the strips member data *xmin*, *xmax*, *ymin*, *ymax*) determine the *physical partitioning* of the spatial domain.

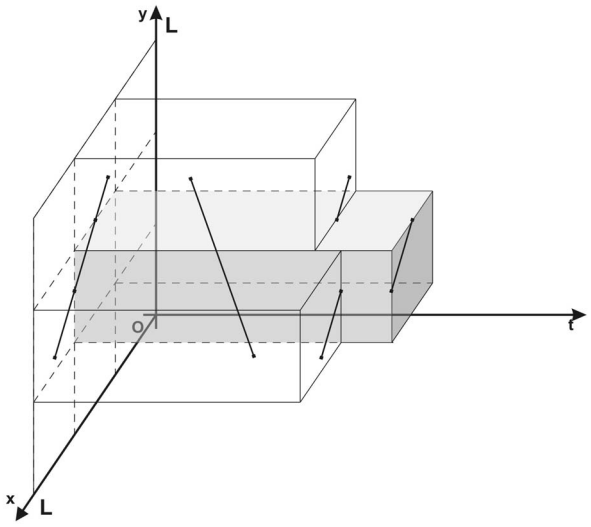


FIGURE 1. The fragmentation of the trajectory segments within the strips of the DBR-Grid. The graphical representation shows the inserted segment, its projection on the xOy plane, and the spatial projections of the obtained three subsegments.

Two aspects have to be mentioned in order to understand the division algorithm that is performed on a 3D strip:

(1) The division is performed on a single spatial dimension.
(2) The division is physically affecting a single strip.

The second observation is related to the fact that choosing a new partitioning hyper-plane *h_div* for a strip *B'* virtually affects all the strips intersected by it, unlike the division of a 2D strip. But it is unlikely that another strip than *B'* needs to be divided in that moment. Therefore, only the strip *B'* is physically affected by the division by *h_div*. If another strip *B"* needs later to be divided and *h_div* intersects *B"*, then the hyper-plan *h_div* is considered as candidate in performing the division of *B"*. The algorithm by which the partitioning hyper-plan is chosen is described next (`ChooseDivision`).

ChooseDivision(B, D, h_div)

```
// Input:
//   B - the strip of the DBR-grid which has to be divided
```

```
// Output:
//   D - the spatial dimension on which the division will
//    be performed
//   h_div - the partitioning hyper-plane on D dimension

// The ChooseDivision routine determines the dimension D
// on which the division will be performed and the
// position of the partitioning hyper-plane h_div

Let R be the projection of B on the spatial domain
Let dx and dy the projections of R on the Ox, and Oy,
respectively
If dx ≤ dy then D := Ox
Else D := Oy
End if
If D = Ox then
   If ∃xd_i ∈ XD such as B.xmin < xd_i < B.xmax then
      h_div := xd_i^*, where xd_i^* ∈ XD, B.xmin < xd_i^* < B.xmax,
                  |xd_i^* − (B.xmin + B.xmax)/2| =
                  min{xd_i − (B.xmin + B.xmax)/2|xd_i ∈ XD}
      // If there exists at least one hyper-plan in XD
      // that intersects B, then h_div is chosen so that
      // to be the closest to the median of R on Ox
   Else
      Choose_x_division(b, h_div)
      // The routine is similar to the Choose_x_division
      // algorithm presented in [10]
   End if
Else // D = Oy
   Choose h_div on the Oy axis as it was chosen in the
   first case
End if
```

End ChooseDivision

The following two examples show the updates on the *DBR_Grid* object (g) during the division and merge operations. The notation $B = (g[i][j]|$ i:=1..nbx, j:=1..nby) is used in order to specify all the g's elements that refer the strip B.

Example 1. Let L = 100, $nbx = nby = 4$, XD = YD = (0, 25, 50, 75, 100) be the initial configuration data of the DBR-Grid g. The initial virtual partitioning concure with the physical partitioning (see Fig. 2(a)). Let $x\_div = 35$ be the position of the division hyper-plane used to divide the strip $B = (g[2][2])$ (the marked cell).

Fig. 2(b) shows the configuration of the spatial projections of the strips, the physical and the virtual partitioning, after the strip has been divided. It can be noticed that the strips that are intersected by x_div are not physically divided; they preserve the position and the content, but they are referred by two elements of the virtual grid. For example, the strip $B' = (g[2][1])$ (see Fig. 2(a)) is given next by $B' = (g[2][1], g[3][1])$. The virtual partitioning lines that are not part of the physical partitioning are represented by dotted lines.

Later, the coordinate of a new partitioning hyper-plan $y\_div = 60$ on Oy axis is determined in order to divide the strip $B = (g[4][3])$. After updating the object g, it can be observed that all the strips referred by elements on the 4^{th} row (with the exception of $g[4][3]$) are now referred by one more element (see Fig. 2(c)). For example, the strip $B' = (g[2][3], g[3][3])$ from Fig. 2(b) is now referred as $B' = (g[2][3], g[3][3], g[2][4], g[3][4])$. Fig. 2(d) represents the configurations of the physical and virtual partitions after a division by $x\_div = 65$ have been performed. It can be observed that the shape of the spatial projections of the grid's strips does not depend on the position, dimension or the order in which the divisions are performed; their shape is continuously rectangular.

Corresponding to the definition of neighbor strips given in [10], two 3D strips, B_1 and B_2, are considered to be neighbors if

$$(B_1.xmax = B_2.xmin \text{ or } B_1.xmin = B_2.xmax) \text{ or}$$
$$(B_1.xmax = B_2.xmin \text{ or } B_1.ymin = B_2.ymax).$$

Also similar to the measure of density defined in [10], the density of segments within an object R (strip or MBR of a tree node) is given by

$$C(R) = nSegm * dx * dy/dt,$$

where M denotes the maximum capacity of a tree node.

The merging technique is related to the manner in which the virtual partitioning was managed during the division operations. Another condition that must be fulfilled by two neighbor strips, B_1 and B_2, is that their spatial projections on the dimension complementary to the dimension on which the strips are neighbors to concur (for example, if the dimension on which the two strips are neighbors is Oy, then $B_1.xmin = B_2.xmin$ and $B_1.xmax = B_2.xmax$). Then, among all the candidate neighbor strips, it is selected the pair of strips which minimizes the sum of their densities of segments. For example, it is considered the configuration represented in Fig. 2(d): let $B_1 = (g[5][4])$ and $B_2 = (g[6][3], g[6][4])$ be two neighbor strips; these two strips cannot be merged because their spatial projections on the Oy axis does not concur ($[60, 75) \neq [50, 75)$).

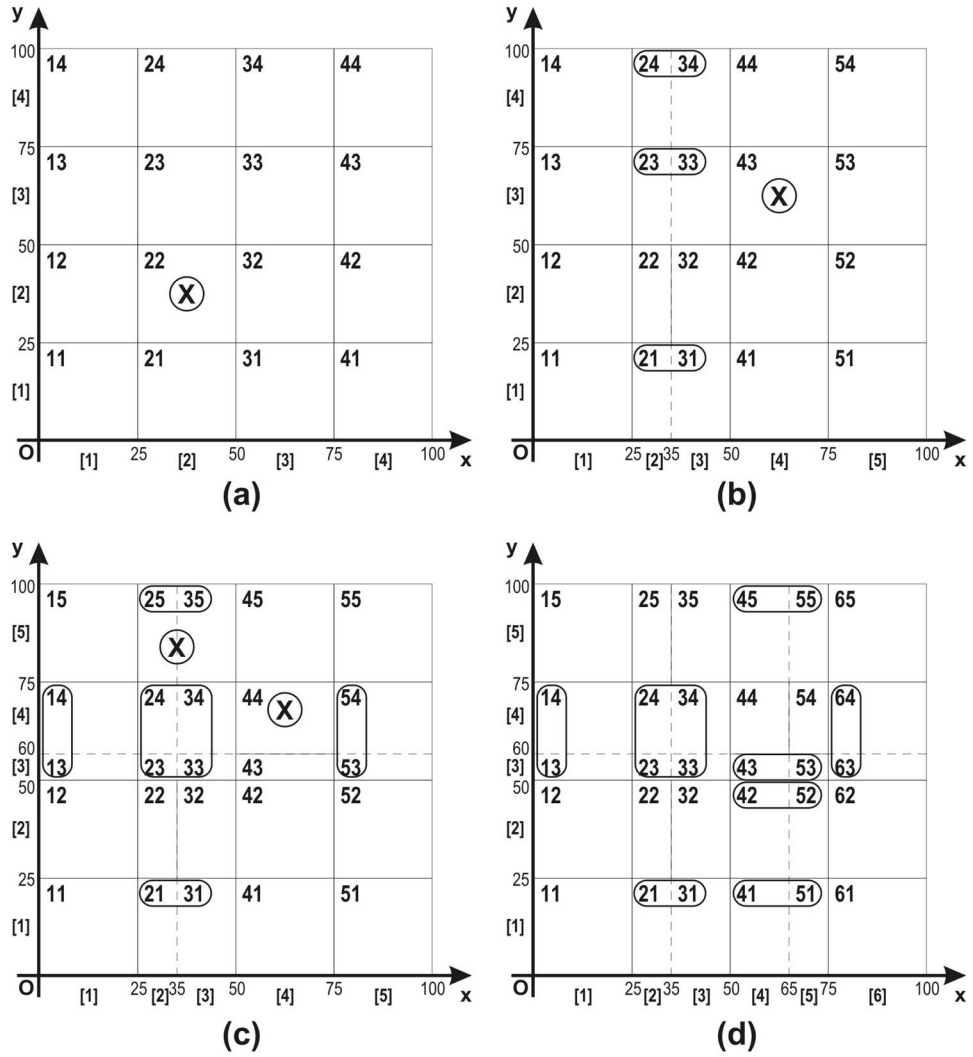


FIGURE 2. The updates performed on the DBR-Grid during the division operations presented in Example 1.

The manner in which the merge of two neighbor strips is performed is presented by Example 2.

Example 2. It is considered the configuration represented in Fig. 2(d). The first merge operation is performed on the strips $B_1 = (g[1][1])$ and $B_2 = (g[1][2])$. Because the hyper-plan $x\_div = 25$ physically delimits other strips,

the reunion affects only the B_1 and B_2 strips, and does not affect the virtual partitioning. A single physical strip $B = (g[1][1], g[1][2])$ results after performing the merge operation (see Fig. 3(a)).

Next, the reunion of strips $B_1 = (g[1][3], g[1][4])$ and $B_2 = (g[2][3], g[3][3], g[2][4], g[3][4])$ is considered. The result of the reunion is given by the strip $B = (g[1][3], g[2][3], g[3][3], g[1][4], g[2][4], g[3][4])$ (see Fig. 3(b)). Fig. 3(c) depicts the spatial configuration obtained by merging the strips $B_1 = (g[2][2])$ and $B_2 = (g[3][2])$. Because the hyper-plane $x\_div = 35$ does no delimit two physical strips, the last merge operation also affects the structure of the virtual grid by eliminating this hyper-plane.

3. Experimental Results

This section presents the comparative results obtained for three access methods: R*-Tree method, BrickR method - similar to Dynamic-BrickR, does not perform strips merging and the Dynamic-BrickR method. These three methods are evaluated in respect to the quality of data organization and the efficiency of answering queries using the corresponding built indexes.

Tests have been run on three synthetic data sets of 3D trajectory segments, which were constructed using sets of points associated to the mobile objects. Each test data set numbers 10000 trajectory segments, recorded for 100 mobile objects. The coordinates (x, y, t) of the data points belong to a well-determined spatio-temporal working interval: the temporal domain was [1, 50000], and the spatial domain was [1..1000] $\times$ [1..1000]. The three test data sets were constructed from: points randomly generated (with uniform distribution), points following a Gaussian distribution and points following a Poisson distribution. For each of the three methods, the capacity of a tree node was set to 25 - considering the size of a disk page equal to 512B. The minimum occupancy of a tree node was set to 40

Regarding the data organization, the measures evaluated on the permanent structures built by the three compared access methods are: the degree of tree nodes occupation, the sum of nodes MBRs areas, the sum of nodes MBRs volumes and the total value of the overlapping volumes between the nodes on the same tree level.

Even if the number of indexed segments by the Dynamic-BrickR permanent structure is greater than the initial number of segments, due to the clipping method, the tests show a better node occupation than in the case of R*-Tree: on the average, the node occupancy in the R*-Tree is 54.15% and in the BrickR and Dynamic-BrickR is 95.27% and 92.76% respectively.

Fig. 4(a) and Fig. 4(b) comparatively present for each sort of data sets the sums of the MBRs area and the sums of the MBRs volumes, and Fig.

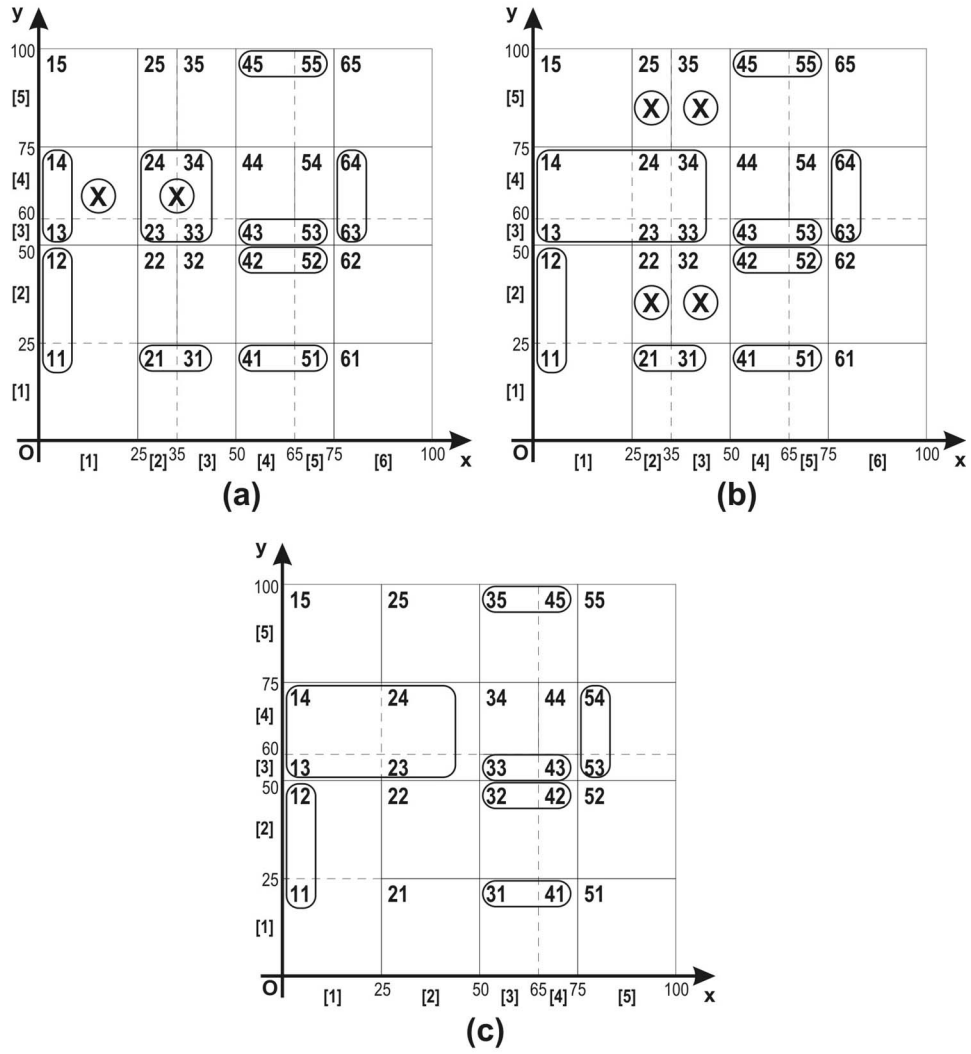


FIGURE 3. The updates performed on the DBR-Grid during the merge operations presented in Example 2.

5 shows the obtained results regarding the overlapping volumes between the nodes on the same tree level. It can be noticed that with only one exception in the case of data sets with Gaussian distribution in evaluating the MBRs areas, the obtained results were better (smaller) for the BrickR methods than for the R*-Tree method.

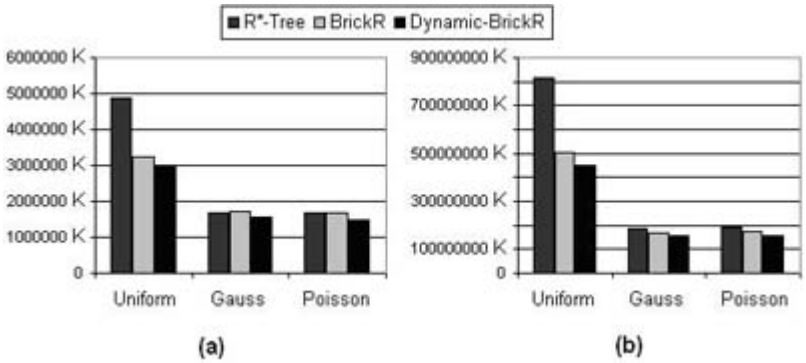


FIGURE 4. (a) The sum of MBRs areas and (b) the sum of MBRs volumes, for the R* Tree, BrickR and Dynamic BrickR methods

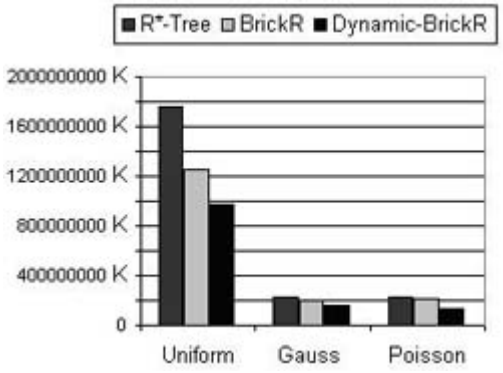


FIGURE 5. Overlapping volumes between the MBRs of nodes on the same tree levels, for all tree nodes, for the R* Tree, BrickR and Dynamic BrickR methods

The queries performed on spatio-temporal data may be classified as: spatial queries, temporal queries and spatio-temporal queries. The types of queries for which tests have been performed consist of combinations of spatial and / or temporal, point and / or window queries, denoted by:

- S-P T-P query: spatial-point temporal-point query;
- S-W query: spatial-window query;
- T-W query: temporal-window query;
- S-W T-W query: spatial-window temporal-window query;
- S-W T-P query: spatial-window temporal-point query;

- S-P T-W query spatial-point temporal-window query.

It can be observed that the S-P T-P, S-W T-P, and S-P T-W types of queries are particular cases of the S-W T-W queries.

The data within query sets follow a uniform distribution on the spatio-temporal working space, and cover the whole working space. On the other hand, two sorts of query sets have been used in the case of window or point-window queries, in accordance with the maximum length of generated intervals: the spatial and temporal length of the query windows represent maximum 25

In all experiments, the average number of tree nodes visited for answering the query was measured. BrickR and Dynamic-BrickR methods obtained better results on uniformly distributed data sets, except in the case of T-W queries. The execution of the T-W queries showed a better performance for the R*-Tree. In the other cases the obtained results for the three compared access methods were relatively closed.

4. Conclusions and Future Work

The Dynamic-BrickR access method that indexes 3D spatio-temporal data preserves the advantages of the previous version of it:

- The trajectory segments are grouped within the grid structure and sent as a node (completely built) in the tree structure. Thus, the number of performed I/O operations is reduced.
- The chances to obtain overlapping volumes between the MBRs of DBR-Tree's terminal nodes are minimized.;
- The inserted terminal nodes are almost fully occupied.

The proposed future work includes the development of a BrickR-like structure for organizing the continuous evolutions of objects with shape. On the other side, the re-organization of the BrickR permanent structure is proposed as future work, so that to limit the number of tree levels by temporal fragmenting of the node set. This way, the enlargement of the indexed data set does not affect in a major way the performance of the structure.

References

[1] M. Abdelguerfi, J. Givaudan, K. Shaw, R. Ladner, *The 2-3 TR-tree, A Trajectory-Oriented Index Structure for Fully Evolving Valid-time Spatio-temporal Datasets*, In Proc. of the ACM Workshop on Adv. in Geographic Information Systems, ACM GIS, 29-34, 2002.

[2] N. Beckmann, H. P. Kriegel, R. Schneider, B. Seeger, *The R*-tree: An Efficient and Robust Access Method for Points and Rectangles*, In Proc. of the Intl. Conf. on Management of Data, SIGMOD, 322-331, 1990.

[3] M. Cai, P. Revesz, *Parametric R-Tree: An Index Structure for Moving Objects*, In Proc. of the Intl. Conf. on Management of Data, COMAD, 57-64, 2000.
[4] V. P. Chakka, A. Everspaugh, J. M. Patel, *Indexing Large Trajectory Data with SETI*, In Proc. of the Conf. on Innovative Data Systems Research, CIDR, 164-175, 2003.
[5] D. Kwon, Sj. Lee, S. Lee, *Indexing the Current Positions of Moving Objects Using the Lazy Update R-tree*, In Mobile Data Management, MDM, 113-120, 2002.
[6] M. A. Nascimento, J. R. O. Silva, Y. Theodoridis, *Evaluation of Access Structures for Discretely Moving Points*, In Proc. of the Intl. Workshop on Spatio-Temporal Database Management, STDBM, 171-188, 1999.
[7] D. Pfoser, C. S. Jensen, Y. Theodoridis, *Novel Approaches in Query Processing for Moving Object Trajectories*, In Proc. of the Intl. Conf. on Very Large Data Bases, VLDB, 395-406, 2000.
[8] C. M. Procopiuc, P. K. Agarwal, S. Har-Peled, *STAR-Tree: An Efficient Self-Adjusting Index for Moving Objects*, In Proc. of the Workshop on Alg. Eng. and Experimentation, ALENEX, 178-193, 2002.
[9] A. Sabau, *Indexing Mobile Objects Using BrickR Structures*, In Studia Universitatis Babes-Bolyai, Informatica, Vol. LI(2), 71-80, 2006.
[10] A. Sabau, A. Campan, *BrickR: The Dynamic-BrickR Access Method for Mobile Objects*, Proc. of the 22nd International Symposium on Computer and Information Sciences, Turkey, 2007.
[11] Y. Tao, D. Papadias, *Efficient Historical R-trees*, In Proc. of the Intl. Conf. On Scientific and Statistical Database Management, SSDBM, 223-232, 2001.
[12] Y. Tao, D. Papadias, J. Sun, *The TPR*-Tree: An Optimized Spatio-temporal Access Method for Predictive Queries*, In Proc. of the Intl. Conf. on Very Large Data Bases, VLDB, 790-801, 2003.
[13] X. Xu, J. Han, W. Lu, *RT-Tree: An Improved R-Tree Indexing Structure for Temporal Spatial Databases*, In Proc. of the Intl. Symp. on Spatial Data Handling, SDH, 1040-1049, 1990.

Faculty of Mathematics and Computer Science, Babes-Bolyai University, Cluj-Napoca, Romania
E-mail address: deiush@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LII**, Number 2, 2007

A STUDY ON CLUSTERING BASED RESTRUCTURING OF OBJECT-ORIENTED SOFTWARE SYSTEMS

ISTVÁN GERGELY CZIBULA AND GABRIELA SERBAN

Abstract. The structure of a software system has a major impact on its maintainability. *Refactoring* is an activity performed through the entire lifecycle of a software system in order to keep the software structure clean and easy to maintain. We have previously introduced in [3] a clustering approach for identifying refactorings in order to improve the structure of software systems. The aim of this paper is to make a comparative analysis on several clustering algorithms (developed based on the approach from [3]) which can be used in order to recondition the class structure of a software system. Based on this analysis, we highlight the advantages of determining refactorings of object-oriented software systems using clustering.

1. Introduction

The structure of a software system has a major impact on the maintainability of the system. This structure is the subject of many changes during the systems lifecycle. Improper implementations of these changes imply structure degradation that leads to costly maintenance.

A continuous improvement of the software systems structure can be made using *refactoring*, that assures a clean and easy to maintain software structure.

In [6] Fowler defines refactoring as "the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure. It is a disciplined way to clean up code that minimizes the chances of introducing bugs". Refactoring is viewed as a way to improve the design of the code after it has been written. Software developers have to identify parts of code having a negative impact on the system's maintainability, and to apply appropriate refactorings in order to remove the so called "bad-smells" [1].

Received by the editors: October 20, 2007.

2000 *Mathematics Subject Classification.* 68N99, 62H30.

1998 *CR Categories and Descriptors.* D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement –*Restructuring, reverse engineering, and reengineering*; I.5.3 [**Computing Methodologies**]: Pattern Recognition – *Clustering*.

We have previously introduced in [3] a clustering approach for identifying refactorings in order to improve the structure of software systems. For this purpose, a clustering algorithm named *kRED* was introduced. To our knowledge, there is no approach in the literature that uses clustering in order to improve the class structure of a software system, excepting the approach introduced in [3]. The existing clustering approaches handle methods decomposition ([15]) or system decomposition into subsystems [10].

We have improved the approach from [3] by developing several clustering algorithms that can be used to identify the refactorings needed in order to recondition the class structure of an object-oriented software system: *HAC* [14], *PAMRED* [11], *HARED* [2], *HARS* [12] and *PARS* [13]. These algorithms are based on the on the idea of *partitional* and *hierarchical* clustering.

The rest of the paper is structured as follows. The main aspects related to clustering, to the approach for determining refactorings using clustering [3] and to the clustering algorithms previously developed are presented in Section 2. The comparative study between the clustering algorithms for identifying refactorings of object-oriented software systems is made in Section 3. An experiment on a real software system is reported in Section 4. Some conclusions and further work are given in Section 5.

2. Background

2.1. **Clustering.** *Clustering* [8], also known as unsupervised classification, is a data mining activity that aims to differentiate groups (classes or clusters) inside a given set of objects, $\mathcal{O}$. The measure used for discriminating objects can be any *metric* or *semi-metric* function $d : \mathcal{O} \times \mathcal{O} \longrightarrow \Re$, called *distance*. A large collection of clustering algorithms is available in the literature ([8]). Most clustering algorithms are based on two popular techniques known as *partitional* and *hierarchical* clustering.

2.2. **A Clustering Approach for Refactorings Determination - *CARD*.** In this subsection we briefly describe the clustering approach (*CARD*) that was previously introduced in [3] in order to find adequate refactorings to improve the structure of software systems. *CARD* approach consists of three steps:

- **Data collection** - The existing software system is analyzed in order to extract from it the relevant entities: classes, methods, attributes and the existing relationships between them.
- **Grouping** - The set of entities extracted at the previous step are regrouped in clusters using a clustering algorithm. The goal of this step is to obtain an improved structure of the existing software system.

- **Refactorings extraction** - The newly obtained software structure is compared with the original software structure in order to provide a list of refactorings which transform the original structure into an improved one.

As described above, at the **Grouping** step of *CARD*, the software system S has to be re-grouped. This re-grouping can be viewed as a ***partition*** of S. We mention that a software system S is viewed in [3] as a set $S = \{s_1, s_2, ..., s_n\}$, where $s_i, 1 \leq i \leq n$ is an *entity* from the system (it can be an *application class*, a *method* from a class or an *attribute* from a class). In our clustering approach, the objects to be clustered are the entities from the software system S. Our focus is to group similar entities from S in order to obtain high cohesive groups (clusters).

2.3. **Clustering Algorithms for Refactorings Determination.** We have developed several clustering algorithms that can be used in the *Grouping* step of *CARD* in order to find an improved structure of a software system: *kRED* [3], *HAC* [14], *PAMRED* [11], *HARED* [2], *HARS* [12] and *PARS* [13].

In order to apply a clustering method for refactorings extraction, a distance function between the entities from a software system has to be defined. This distance has to express the idea of cohesion between the entities from the software system.

We have defined in [3] a modality to compute the dissimilarity degree *diss* between any two entities from the software system S. *diss* is a semi-metric and expresses the distance between the entities from the software system and emphasizes the idea of cohesion. The dissimilarity degree *diss* highlights the concept of cohesion, i.e., entities with low distances are cohesive, whereas entities with higher distances are less cohesive.

In developing our clustering algorithms, we have used two approaches:

- The first approach is to use a vector space model based clustering. We have defined a vector space that characterizes the entities from S and, based on *diss*, we have used distance metrics (*Euclidian distance*, *Manhattan distance*, *Hamming distance*) in order to express the dissimilarity between the entities from the software system. In this direction we have introduced two vector space model based clustering algorithms *kRED* and *HAC* that can be used in the *Grouping* step of *CARD* in order to obtain an improved structure of a software system.
- The second approach is to use only the distance between the entities from S given by the semi-metric *diss*. In this direction we have introduced four clustering algorithms *PAMRED*, *HARED*, *HARS*, and *PARS* that can be used in the *Grouping* step of *CARD* in order to obtain an improved structure of a software system.

Another classification of the developed algorithms is based on the clustering method used:

- *kRED*, *PAMRED*, and *PARS* are *partitional* clustering algorithms;
- *HAC*, *HARED* and *HARS* are *hierarchical* clustering algorithms.

3. Comparative analysis

In order to comparatively analyze the proposed clustering algorithms, we consider as case study the open source software JHotDraw, version 5.1 ([7]). It is a Java GUI framework for technical and structured graphics, developed by Erich Gamma and Thomas Eggenschwiler, as a design exercise for using design patterns. It consists of **173** classes, **1375** methods and **475** attributes. The reason for choosing JHotDraw as a case study is that it is well-known as a good example for the use of design patterns and as a good design.

Our focus is to test the accuracy of our approach on JHotDraw, i.e., how accurate are the results obtained after applying the algorithms in comparison with the current design of JHotDraw.

For evaluation, we use two measures:

- **Accuracy of classes recovery - ACC.** *ACC* defines the degree to which the partition obtained by the clustering algorithm is similar to the initial structure of the analyzed software system.
- **Precision of entities discovery - PREC.** *PREC* defines the percentage of entities (methods and attributes) from the software system that were correctly (in comparison with the current design of the system) discovered in the partition reported by a clustering algorithm.

The evaluation of the results obtained by applying the above defined algorithms on JHotDraw case study are made using the following characteristics:

- *ACC* measure that has to be maximized;
- *PREC* measure that has to maximized;
- the running time of the algorithm that has to minimized.

All these algorithms provide better results than the approaches existing in the literature in the field of refactoring. Table 1 gives the comparative results.

A graphical representation of the results illustrated in Table 1 is given in Figure 1.

Based on the results presented in Table 1 and Figure 1, we can conclude that *PARS* algorithm provides the best results.

4. Case Study

As shown in Section 3, from the analyzed clustering algorithms for identifying refactorings, *PARS* algorithm provides the best results. That is why

Algorithm	ACC	PREC	Running time (min.)
kRED	**0.9829**	**0.9966**	**5**
HAC	**0.9899**	**0.9945**	**6**
PAMRED	**0.9939**	**0.9994**	**1.5**
HARED	**0.974**	**0.9978**	**3.5**
HARS	**0.974**	**0.9978**	**3.68**
PARS	***1***	***1***	***1.5***

TABLE 1. Comparative results.

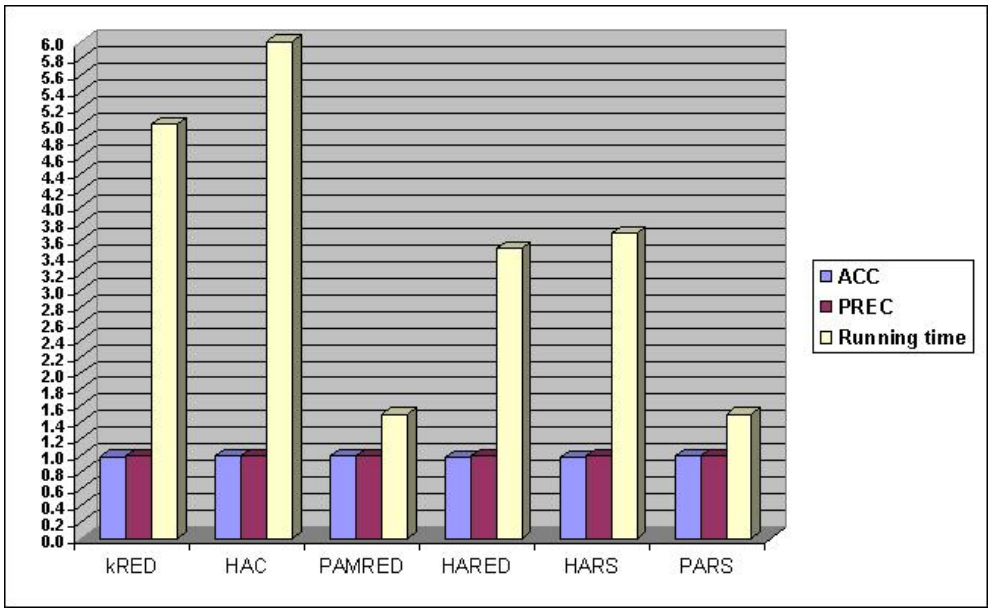


FIGURE 1. The comparative results.

in this section we present a real software system as a case study for evaluating *PARS* algorithm. It is DICOM (*Digital Imaging and Communications in Medicine*) [5] and HL7 (*Health Level 7*) [9] compliant PACS (*Picture Archiving and Communications System*) system, facilitating the medical images management, offering quick access to radiological images, and making the diagnosing process easier.

The analyzed application is a large distributed system, consisting of several subsystems in form of stand-alone and web-based applications. We have applied *PARS* algorithm on one of the subsystems from this application.

The analyzed subsystem is a stand-alone Java application used by physicians in order to interpret radiological images. The application fetch clinical images from an image server (using DICOM protocol), display them, and offer various tools to manage radiological images.

Even if the application is currently used, it also continuously evolves in order to satisfy change requirements and to provide better user experience based on feedback. That is why, the developers are often faced with the need of structural and conceptual changes.

The analyzed application consists of **1015** classes, **8639** methods and **4457** attributes.

After applying *PARS* algorithm, a total of 90 refactorings have been suggested: 10 *Move Attribute* refactorings, 78 *Move Method* refactorings, and 2 *Inline Class* refactoring.

The obtained results have been analyzed by the developers of the application and the following conclusions were made:

- 28.8% from the refactorings identified by *PARS* were accepted by the developers as useful in order to improve the system.
- 21.1% from the refactorings were acceptable for the developers, but they concluded that these refactorings are not necessary in the current stage of the project.
- 50.1% from the refactorings were strongly rejected by the developers.

Analyzing the obtained results, based on the feedback provided by the developers, we have concluded the following:

- *PARS* successfully identified smart GUI anti-patterns (parts of software were the presentation layer contains bussiness logic), misplaced constants (constants used only on a subtree of a class hierarchy, but defined in some base class). These kind of weaknesses can be discovered only if the developer manually inspects all the classes, or if a bug (related to the misplaced bussiness logic) arises. That is why automatic detection by *PARS* of these kind of weaknesses can prevent system failure or other kind of bugs and also save a lot of manual work.
- A large number of miss-identified refactorings are due to technical issues: the use of Java anonymous inner classes, introspection, the use of dynamic proxies. These kind of technical aspects appear frequently in projects developed in Java. In order to correctly deal with these aspects, we have to improve only the **Data collection** step of our approach, without modifying the *PARS* algorithm.
- Another cause of miss-identified refactorings is due to the fact that the *distance* used for discriminating entities in the clustering process take into account only two aspects of a good design: *low coupling* and

high cohesion. It would be also important to consider other principles related to an improved design, like: *Single Responsibility Principle*, *Open-Closed Principle*, *Interface Segregation Principle*, *Common Closure Principle* [4], etc.

- Our approach is currently implemented as a stand-alone application: the user provides the .jar files containing the classes of the analyzed software system and our application displays the suggested refactorings. The developers have suggested that it would be preferable to integrate our tool existing IDE (as a plugin), instead of a stand-alone application.

5. Conclusions and Future Work

We have presented in this paper a comparative analysis of several clustering algorithms that we have previously developed, algorithms which can be used in order to recondition the class structure of a software system. As a conclusion, the advantages of our approach for determining refactorings using clustering are:

- it can deal with various types of refactorings;
- it can be applied for large software systems;
- it can offer support to software developers for identifying ill-structured software modules.

Further work can be done in the following directions:

- To study the applicability of other learning techniques in order to improve software systems design.
- To use other search based approaches in order to determine refactorings that improve the design of a software system.
- To develop a tool (as a plugin for Eclipse) that is based on *CARD*.
- To apply our approach in order to transform non object-oriented software into object-oriented systems.

References

[1] William J. Brown, Raphael C. Malveau, III Hays W. McCormick, and Thomas J. Mowbray, *Antipatterns: refactoring software, architectures, and projects in crisis*, John Wiley & Sons, Inc., New York, NY, USA, 1998.

[2] I.G. Czibula and Gabriela Serban, *A Hierarchical Clustering Algorithm for Software Systems Design Improvement*, KEPT 2007: Proccedings of the First International Conference on Knowledge Engineering: Principles and Techniques, 2007, pp. 316–323.

[3] Istvan G. Czibula and Gabriela Serban, *Improving Systems Design using a Clustering Approach*, IJCSNS International Journal of Computer Science and Network Security **6** (2006), no. 12, 40–49.

[4] Tom DeMarco, *Structured analysis and system specification*, Addison-Wesley Longman Publishing Co., Inc., Prentice Hall, 1979.
[5] *Digital Imaging and COmmunications in Medicine*. at Web: http://medical.nema.org/.
[6] M. Fowler, *Refactoring: Improving the Design of Existing Code*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
[7] E. Gamma, *JHotDraw Project*. http://sourceforge.net/projects/jhotdraw.
[8] Jiawei Han, *Data mining: Concepts and techniques*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
[9] *Health Level 7*. at Web: www.hl7.org/.
[10] Chung-Horng Lung, *Software Architecture Recovery and Restructuring through Clustering Techniques*, ISAW '98: Proceedings of the Third International Workshop on Software Architecture, 1998, pp. 101–104.
[11] Gabriela Serban and Istvan G. Czibula, *A New Clustering Approach for Systems Designs Improvement*, SETP-07: Proceedings of the International Conference on Software Engineering Theory and Practice, 2007, pp. 47–54.
[12] Istvan G. Czibula and Gabriela Serban, *Hierarchical Clustering for Software Systems Restructuring*, INFOCOMP Journal of Computer Science, Brasil (2007), to be published.
[13] Gabriela Serban and Istvan G. Czibula, *Restructuring software systems using clustering*, ISCIS 2007: Proceedings of the 22nd International Symposium on Computer and Information Sciences, 2007, pp. to be published.
[14] Istvan G. Czibula and Gabriela Serban, *Software systems design improvement using hierarchical clustering*, SERP 2007: Proceedings of SERP '07, 2007, pp. 229–235.
[15] Xia Xu, Chung-Horng Lung, Marzia Zaman, and Anand Srinivasan, *Program Restructuring through Clustering Techniques*, SCAM '04: Proceedings of the Source Code Analysis and Manipulation, Fourth IEEE International Workshop, 2004, pp. 75–84.

Department of Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu Street, 400084 Cluj-Napoca, Romania
E-mail address: istvanc@cs.ubbcluj.ro

Department of Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu Street, 400084 Cluj-Napoca, Romania
E-mail address: gabis@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LII**, Number 2, 2007

AN AGILE MDA APPROACH FOR EXECUTABLE UML STRUCTURED ACTIVITIES

I. LAZĂR, B. PÂRV, S. MOTOGNA, I.-G. CZIBULA, AND C.-L. LAZĂR

Abstract. Agile processes allow developers to construct, run and test executable models in short, incremental, iterative cycles. However, the agile development processes tend to minimize the modeling phase and the usage of UML models, because UML is a "unified" (too general) language with a lot of semantic variation points. The current version of UML together with its Action Semantics provide the foundation for building object-oriented executable models. But, constructing executable models using the existing tools and the current standard notations is a tedious task or an impossible one because of the UML semantic variation points. Agile MDA processes try to apply agile principles in the context of executable models. This paper presents a procedural action language for UML structured activities that allows developers to apply agile principles for executable models that contains structured activities. New graphical notations for structured activities are also introduced for rapid creation of tests and procedures.

1. Introduction

UML 2 [16] is the de-facto standard for modeling software systems. However, most commonly, UML models are used as blueprints that are fill in with code, and the current agile development processes (e.g. agile model-driven development [2], test-driven development [3]) tend to minimize the modeling phase and the usage of UML models.

MDA framework [10] provides an approach for specifying systems independently of a particular platform and for transforming the system specification into one for a particular platform. But development processes based on MDA are not widely used today because they are viewed as heavy-weight processes - they cannot deliver (incrementally) small slices of code as soon as possible.

Received by the editors: November 19, 2007.

2000 *Mathematics Subject Classification.* 68N15, 68N30.

1998 *CR Categories and Descriptors.* D.2.2 [**Software Engineering**]: Design Tools and Techniques – *Computer-aided software engineering, Flow charts, Object-oriented design methods*; D.2.4 [**Software Engineering**]: Software/Program Verification – *Programming by contract, Assertion checkers*; D.2.5 [**Software Engineering**]: Testing and Debugging – *Debugging aids, Testing tools*;

In this context, executing UML models became a necessity for development processes based on extensive modeling. For such processes *models must act just like code* [18]. UML 2 and its Action Semantics [16] provide the foundation to construct executable models. In order to make a model executable, the model must contain a complete and precise behavior description. But, creating a model that have a complete and precise behavior description is a tedious task or an impossible one because of many UML semantic variation points.

Executable UML [19] means an execution semantics for a subset of actions sufficient for computational completeness. Two basic elements are required for such subsets: an *action language* and an *operational semantics*. The action language specifies the elements that can be used and the operational semantics establishes how the elements can be placed in a model, and how the model can be interpreted.

Several tools [5, 24, 1, 7] have defined *non-standard* subsets of actions that make UML a computational-complete specification language. The operational semantics of a *standard* subset of actions sufficient for computational completeness is still in the process of standardization [12].

Debugging and testing executable models early in the development process help to validate parts of the model and to improve it. Model-level Testing and Debugging Specification [15] and UML Testing Profile [13] define a standard infrastructure for testing and debugging at the PIM (Platform Independent Model), PSM (Platform Specific Model), and implementation levels. The above specifications allow *glass box* and *black box* testing of application based on models.

1.1. **The Problem and Motivation.** As identified above, a framework for executing UML structured activities should be based on the following elements:

- An agile MDA process that allows developers to construct, run and test executable models in short, incremental, iterative cycles. *Glass box* and *black box* testing must also be provided.
- A small subset of actions sufficient for computational completeness together with simple graphical and textual notations for representing the action language elements.
- Model management operations for model transformation and validation.

Agile MDA processes. An agile MDA process [18] applies the main Agile Alliance principles (e.g. testing first, imediate execution [2, 3]) into a classical MDA process [10, 8].

Some of the existing tools provide *glass box* and *black box* testing using *non-standard* infrastructures, but these techniques must be alligned today with the standard specifications for debugging and testing [15, 13].

Subsets of actions sufficient for computational completeness. As noted before, the standardization efforts for defining a subset of actions sufficient for computational completeness are in progress [12], while existing tools provide several action languages.

As described in [9] the ideas behind existing proprietary tools are quite similar. The process of creating executable models can be generalized as follows: (a) the system is decomposed as a set of components, (b) each component is detailed using class diagrams, (c) the behavior of each class is detailed using state machines, and (d) the actions used in state diagrams are specified using a proprietary action language.

There are action languages [5, 24, 1] whose elements are translatable into the basic concepts defined by the UML 2 Action Semantics, and action languages based on OCL [21] which extends OCL query expressions and adds sideefects capability to OCL. However, all these languages provide only concrete syntaxes and do not provide simple graphical notations for activity diagrams.

Model management operations. Meta-Object Facility [11] is a metamodeling language that provides core facilities for defining new modeling languages including model transformation languages. Another well-known and widely used framework for implementing model management is the Eclipse Modelling Framework (EMF) [6].

There are several defined languages for model transformation and validation. The Epsilon Object Language (EOL) [17] is a metamodel independent language built on top of OCL [14]. Kermeta [7] is a metamodelling language, compliant with the EMOF component of MOF 2.0, which provides an action language for specifying behaviour. Kermeta is intended to be an imperative language for implementing executable metamodels [7].

1.2. **The Solution.** The proposed solution is a framework for constructing and executing UML structured activities. The framework refers only to UML structured activities because our first objective is to allow model transformation from PIM to procedural constructs of imperative languages. This framework for structured activities is part of ComDeValCo - Component Definition, Validation, and Composition framework [23].

As part of this framework we define a *procedural action language* (PAL), that is a concrete syntax for UML structured activities, and graphical notations for some UML structured activity actions.

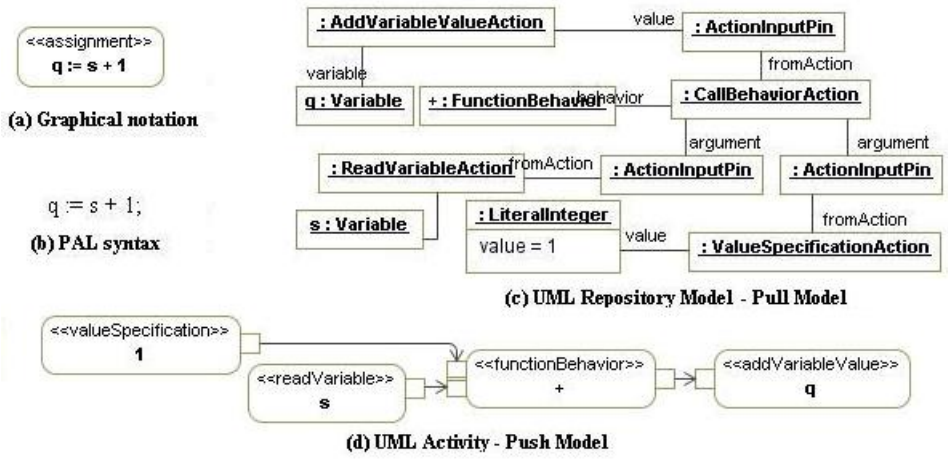


FIGURE 1. Assignment: *q := s + 1*

One of the main idea for simplifying the construction of UML structured activities is to use the *pull data flow* for expression trees. The pull model means that actions requiring data initiate other actions that provide it. Figure 1-(d) shows the push model for evaluating the expression *s+1* and adding the result to an activity variable *q*. As shown in [4], when modeling expressions using the push data flow the control arrives at leaves of the expression tree, then the data cascades through the root, producing the final result. But modeling expression trees using the push model is a tedious task.

The proposed framework uses the pull model for expression trees. Figures 1-(a) and 1-(b) shows the graphical and textual notations for the assignment *q := s + 1*. Both notations can be compiled to the same UML repository model presented Figure in 1-(c).

We also propose new graphical notations for conditionals and loops. The graphical notations do not follow Nassi-Schneiderman notations [22] for structured programming. For simplicity we propose the classical flowchart graphical notations.

The framework also includes an Agile MDA approach for constructing, running and testing models. Debugging and testing techniques are also included according to the new released standards [15, 13].

In order to be able to exchange executable models with other tools, a UML profile is also defined. The profile defines the mapping between PAL and UML constructs and is similar to the profile defined for AOP executable models [9].

The paper is organized as follows: after this introductory section, the next one presents our agile MDA approach. The third section presents the

Procedural Action Language. The last section contains some conclusions and future work.

2. Our Agile MDA Approach

Our approach is illustrated using an example program that computes the integer square root (*isqrt*) of a positive integer. $isqrt(s)$ is the positive integer r which is the greatest integer less than or equal to the square root of s.

In order to develop a program we construct a UML model that contains functional model elements and test case model elements. Functional model elements correspond to the program and its operations and are represented as UML activities. Test case model elements are also UML activities and they represent automated tests written for some selected functional model elements.

For instance, our model for the above example contains the following elements: an *isqrtProgram* activity for program, an *isqrt* activity for computing the integer square root, and a *testIsqrt* activity which is a test case for *isqrt* activity. The creation order of these model elements is as follows.

1: First we create the test case model (i.e. *testIsqrt* activity for *isqrt* operation) starting from the above *informal* specification. At this stage we try to understand the requirements by writing test scenarios using UML structured activity constructs.

2: Formal *pre* and *post* conditions of *isqrt* are written after the test case is created. We can return at step 1 to complete the test scenarios based on the defined formal specification.

3: Finally, we define *isqrt* and *isqrtProgram* activities using UML structured activity nodes. To allow *glass box testing* we can mark the functional model elements according to UML Model-level Testing and Debugging Specification.

The examples presented in this section contain PAL graphical and textual notations that will be described in the next section.

2.1. Test-first Design Steps.

Our proposed agile MDA process includes the test-first design steps [3] as follows. For each new feature of the system we apply the bellow steps.

Add a test. The first step is to quickly add a test. Figure 2 shows a test case for *isqrt*, expressed using (a) a graphical notation and (b) a textual notation. Figure 2-(a) contains an activity diagram that shows *testIsqrt* activity stereotyped with *testCase* defined by UML Testing Profile [13]. The *assert* stereotype defined by our profile is used to make assertions and can be applied for UML 2 *CallBehaviorAction*s. Figure 2-(b) presents the concrete syntax of PAL corresponding to *testIsqrt* activity.

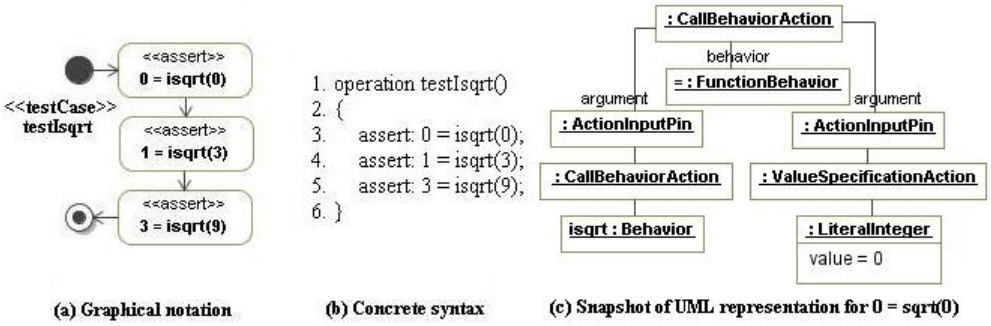


FIGURE 2. Isqrt Test Case

Developers can write the tests using the graphical or the textual notations. Both notations are compiled into the same UML repository model as shown in Figure 2-(c), where a snapshot is presented without pin and parameter objects. For easy of use the framework allows developers to write *inline* expressions when they construct activities. Inline expressions are represented and evaluated according to the pull model for actions.

Run the tests. The second step is to run all the tests to ensure that the new test fails. In order to run the tests the model is verified and the missing elements are reported - in this example *isqrt* operation. The framework helps developers to generate the missing elements as Figure 3 shows.

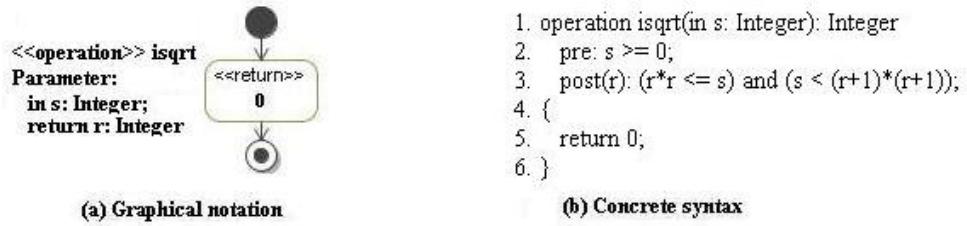
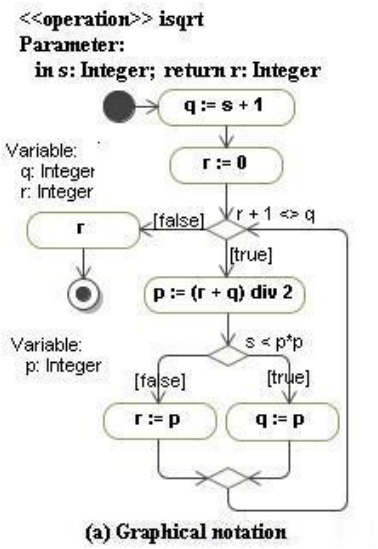


FIGURE 3. Automatically Generated isqrt Operation

At this stage developers can write *pre* and *post* conditions expressed as OCL expressions [14]. The syntax of PAL includes *pre* and *post* constructs as Figure 3-(b) shows. The expressions specified in *pre* and *post* sections will be used when the system is run - see section 2.2.




```
1.  operation isqrt(in s: Integer): Integer
2.    pre: s >= 0;
3.    post(r): (r * r <= s) and (s < (r+1)*(r+1)) ;
4.  {
5.    var q: Integer := s + 1;
6.    var r: Integer := 0;
7.    while (r + 1 <> q) {
8.      loopInvariant: (r * r <= s) and (s < q * q);
9.      loopVariant: r + 1 - q;
10.     var p: Integer;
11.     p := (r + q) div 2;
12.     if (s < p*p) { q := p; }
13.     else { r := p; }
14.   }
15.   assert: r + 1 = q;
16.   return r;
17.}
```

(b) Concrete syntax

FIGURE 4. Isqrt Operation Definition

Add production code. The third step is to update the functional code to make it pass the new test. Figure 4 shows the definition of *isqrt* operation, without showing the stereotypes in order to save space.

As for writing test cases, developers can use either the graphical notation or the concrete syntax of PAL. Figure 4-(b) contains statements that corresponds to an assertion based language [25]. The framework allows and encourages developers to apply design by contract principles [20]. The *assert* statement corresponds to *data assertions* - conditions that must hold at a particular location in the code, as defined in [25]. The *loopInvariant* statement can be used inside loops and it is a particular data assertion that states what must hold in each repetition of a loop. The *loopVariant* statement introduces a strictly monotonic decreasing function used for loop termination. All these constructs can be used when the program is run - see section 2.2.

Run the tests. The fourth step is to run the tests again. Once the tests pass the next step is to start over implementing a new system feature.

2.2. **Debugging Techniques.** How to enter input data for executable models and how to start the execution represent two requirements for executable models [12]. Programs represent in our framework the entry points for model execution. Like operations, programs are also modeled as UML activities. PAL contains *input* and *output statements* that allow developers to enter data

before model execution and to view the program results. In this context running a model means starting the execution from an activity stereotyped with *program*.

Figure 5 shows a program that reads an integer, computes the integer square root of that value, and writes the result. When the program is run the user is prompted to enter an integer value and the results are sent to a console.

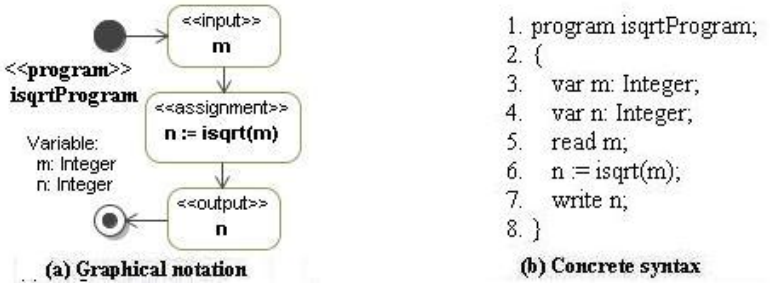


FIGURE 5. Isqrt Program

The debugging techniques are defined according to Model-level Testing and Debugging Specification [15]. Figure 6 presents an extract of the infrastructure of our framework. All classes except *ModelEditor* and *Debugger* classes, belong to the Test Instrumentation Interface (TII) metamodel from [15]. In our context, the system under test (*SUT*) contains only a *DeployedComponent* which is a program. *Breakpoint* represents a location or incident within the program that is of interest. *IncidentBreakpoint*s can be set on any named element within a model and *ActionSemanticBreakpoint*s can be set only on actions. Incident and action breakpoints can be set *manually* on

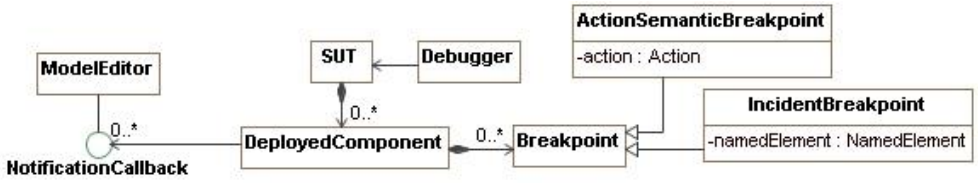


FIGURE 6. Debugging Infrastructure

model elements when the model is constructed (using the ModelEditor). After *Debugger* is started, it notifies the editor when incident and action breakpoints are encountered.

Another option is to inspect the program execution regarding the built-in assertion based constructs (*pre*, *post*, *assert*, *loopVariant*, *loopInvariant*). The *Debugger* component can automatically generate incident breakpoints (a) when encountering assertions, loop invariants, and loop variants, (b) before entering a method - breakpoint set on precondition, and (c) before returning from an operation - breakpoint set on postcondition.

When the debugger is paused developers can inspect the program state, evaluate expressions that use program elements, including the expressions of assertion based constructs.

3. Procedural Action Language

The Procedural Action Language (PAL) is introduced to simplify the construction of UML structured activities. PAL defines a concrete syntax for representing UML structured activity nodes for loops, sequences of actions and conditionals. The PAL syntax is also used for writing assignment statements and expressions in structured activity nodes. PAL also includes assertion based constructs as described in the previous section. For these expressions, PAL uses OCL expressions.

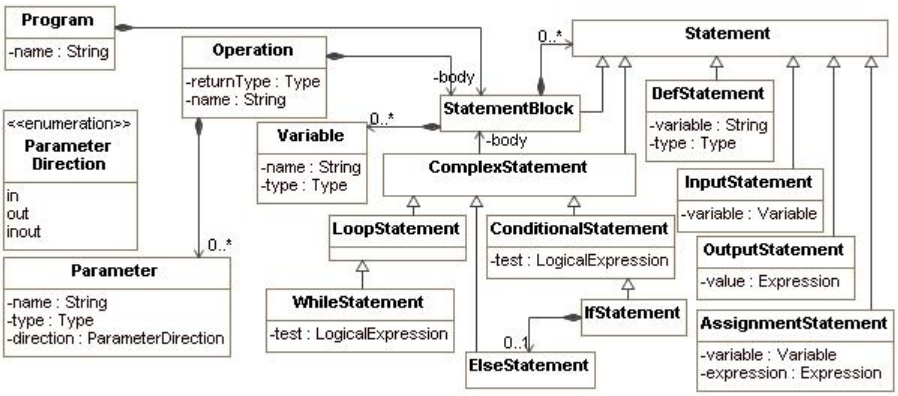


Figure 7. Snapshot PAL Abstract Syntax

Figure 7 presents a snapshot of the core part of the abstract syntax of the language. The missing parts of the abstract syntax refer to expressions and assertion based statements. A PAL profile (see Figure 8) is defined in order to be able to exchange models with other UML 2 compliant tools.

3.1. **Operations and Program.** As the examples from Figure 5 and 4 show, the programs and procedures corresponds to UML activities. A UML Activity

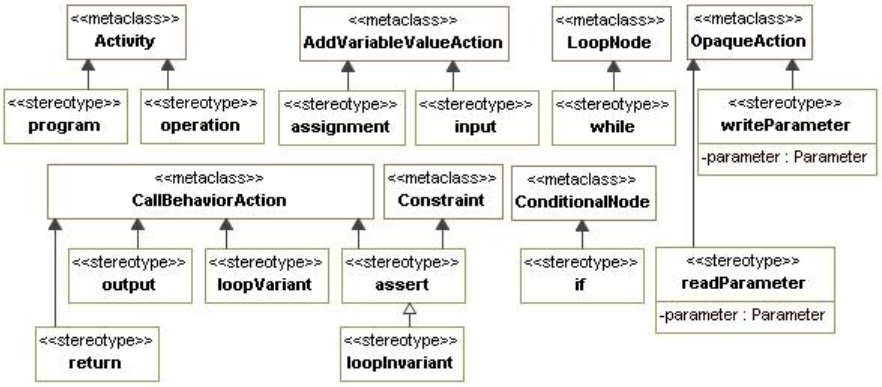


FIGURE 8. PAL UML Profile

has parameters, local variables, preconditions and postconditions, so we have a direct mapping from PAL *Program* and *Procedure* meta classes to the UML *Activity* meta class.

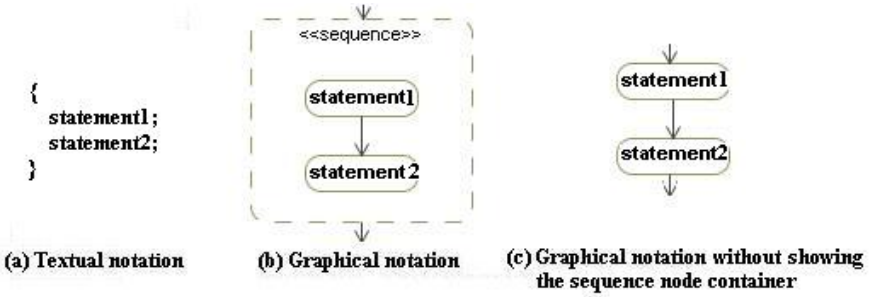


FIGURE 9. Statement Blocks and UML Sequence Nodes

3.2. **Statement Blocks and UML Sequence Nodes.** An UML sequence node is a basic structured node that executes a series of actions in order. The PAL statement blocks correspond to UML sequence nodes. The UML 2 standard does not indicate a standard graphical notation for sequence nodes. Our proposed graphical notations for sequence nodes are presented in Figure 9-(a) and (b).

3.3. **Variable Definition and Assignment Statements.** The PAL variable definition statements can be placed inside statement blocks and can also have expressions for initializing their values. The PAL variables are mapped

to UML *Activity* or *StructuredActivityNode* variables. For instance the variable p defined in line 10 of Figure 4-(b) belongs to the UML loop node that contains the variable definition, while the local variables q defined in line 5 of Figure 4-(b) belongs to *isqrt* activity.

The UML *AddVariableValueAction*s correspond to PAL assignment statements because the left hand side of a PAL assignment is restricted to be a variable. As noted in section 1 we add the constraint for using the pull action model for evaluating the right hand side expression - which is represented and evaluated as a *CallBehaviorAction*.

3.4. **If Statement and UML Conditional Node.** The PAL *IfStatement*s correspond to UML *ConditionalNode*s. In case the else part is missing, the corresponding UML *ConditionalNode* has only one *Clause*, otherwise two *Clause*s. For simplicity we restrict the body of UML clauses to be sequence nodes. The proposed graphical notations for *if* statements are presented in Figure 10-(a) and (b) (UML 2 standard does not indicate a standard graphical notation for sequence nodes).

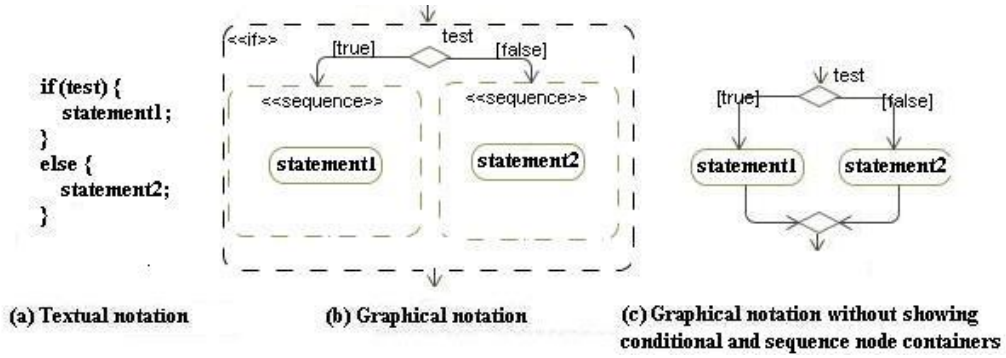


FIGURE 10. If Statement and UML Sequence Nodes

3.5. **While Statement and UML Loop Node.** Pre tested UML *LoopNode*s correspond to *PAL* while statements. Similar to conditional nodes we restrict the body part loop nodes to be sequence nodes.

3.6. **Other Statements.** The PAL *input* statements correspond to UML *AddVariableValueAction*s. The grahical notation must only indicate the variable - the right hand side must be undefined.

The *output*, *return*, and *loop variant* statements are *CallBehaviorAction*s, that is all indicate an expression to be printed, returned, respectively checked.

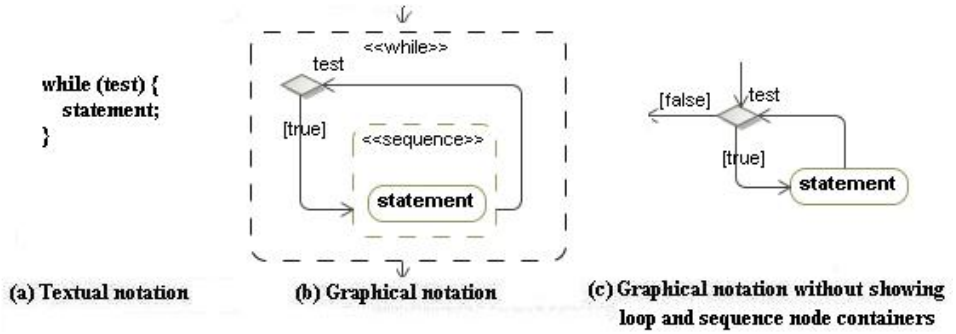


FIGURE 11. While Statement and UML Loop Node

The assertion based statements, *assert* and *loop invariant*, are mapped to UML *Constraint*s or *CallBehaviorAction*s. The loop invariant statement is restricted to be applied only inside loop nodes.

4. Conclusions and Future Work

In order to obtain an agile MDA framework for UML structured activities, this paper has introduced a *Procedural Action Language* and a corresponding UML profile. A concrete syntax and new graphical notations for structured activities have also been defined for this language. The introduced textual and graphical notations can be used to easily construct, run and test executable models according to Agile Alliance principles. Models based on the introduced profile can be constructed with any UML tool, or can run in any UML tool with execution capabilities.

As future work we intend to extend the language with object oriented constructs. Such a language should also support mappings to general UML 2 activities. Additionally, model transformation capabilities must also be extended.

We also intend to add refactoring techniques to the presented Agile MDA approach in order to become a test-driven development method for executable models.

ACKNOWLEDGEMENTS

This work was supported by the grant ID_546, sponsored by NURC - Romanian National University Research Council (CNCSIS).

References

[1] Telelogic AB. *UML 2.0 Action Semantics and Telelogic TAU/Architect and TAU/Developer Action Language, Version 1.0*. 2004.

[2] Scott W. Ambler. *Agile Model Driven Development (AMDD)*. http://www.agilemodeling.com/essays/amdd.htm, 2007.

[3] Kent Beck. *Test-Driven Development By Example*. Addison Wesley, 2002.

[4] Conrad Bock. Uml 2 activity and action models, part 6: Structured activities. *Journal of Object Technology*, 4(4):43–66, 2005.

[5] Kennedy Carter. *The Action Specification Language Reference Manual*. http://www.kc.com/, 2002.

[6] Eclipse.org. *Eclipse Modelling Framework*. http://www.eclipse.org/emf.

[7] Pierre-Alain Muller et al. On executable meta-languages applied to model transformations. In *Model Transformations In Practice Workshop*, Montego Bay, Jamaica, 2005.

[8] Susumu Hayashi et al. Test driven development of uml models with smart modeling system. In *Lecture Notes in Computer Science*, volume 3273, pages 395–409, 2004.

[9] Lidia Fuentes and Pablo Sánchez. Designing and weaving aspect-oriented executable uml models. *Journal of Object Technology*, 6(7):109–136, 2007.

[10] Object Management Group. *MDA Guide Version 1.0.1*. http://www.omg.org/docs/omg/03-06-01.pdf, 2003.

[11] Object Management Group. *Meta Object Facility (MOF) 2.0, Core Specification*. http://www.omg.org/cgi-bin/doc?ptc/04-10-15/, 2004.

[12] Object Management Group. *Semantics of a Foundational Subset for Executable UML Models RFP*. http://www.omg.org/cgi-bin/apps/doc?ad/05-04-02.pdf, 2005.

[13] Object Management Group. *UML 2.0 Testing Profile Specification*. http://www.omg.org/cgi-bin/apps/doc?formal/05-07-07.pdf, 2005.

[14] Object Management Group. *Object Constraint Language Specification, version 2.0*. http://www.omg.org/cgi-bin/apps/doc?formal/06-05-01.pdf, 2006.

[15] Object Management Group. *Model-level Testing and Debugging*. http://www.omg.org/cgi-bin/doc?ptc/2007-05-14/, 2007.

[16] Object Management Group. *UML 2.1.1 Superstructure Specification*. http://www.omg.org/cgi-bin/doc?ptc/07-02-03/, 2007.

[17] Dimitrios S. Kolovos, Richard F. Paige, and Fiona A.C. Polack. The epsilon object language (eol). In *Proc. of European Conference in Model Driven Architecture (EC-MDA)*, pages 128–142, Bilbao, Spain, 2006.

[18] Stephen J. Mellor. Agile mda. Technical report, Project Technology, Inc., 2005.

[19] Stephen J. Mellor and Marc J. Balcer. *Executable UML: A Foundation for Model-Driven Architecture*. Addison Wesley, 2002.

[20] Bertrand Meyer. Applying design by contract. *Computer*, 25(10):40–51, 1992.

[21] P.-A. Muller, P. Studer, F. Fondement, and J. Bzivin. Platform independent web application modeling and development with netsilon. *Software and System Modeling*, 4(4):424–442, 2005.

[22] I. Nassi and B. Schneiderman. Flowchart techniques for structured programming. *ACM Sigplan Notices*, 8(8):12–26, 1973.

[23] Bazil Parv, Simona Motogna, Ioan Lazar, Istvan-Gergely Czibula, and Codrut-Lucian Lazar. Comdevalco - a framework for software component definition, validation, and composition. *Studia Univ. Babes-Bolyai*, LII(2), 2007.

[24] Inc ProjTech AL: Project Technology. *Object Action Language*. 2002.

[25] Herbert Toth. On theory and practice of assertion based software development. *Journal of Object Technology*, 4(2):109–129, 2005.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu, Cluj-Napoca 400084, Romania

E-mail address: {ilazar,bparv,motogna,czibula}@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LII**, Number 2, 2007

EVOLUTIONARY COALITION FORMATION IN COMPLEX NETWORKS

LAURA DIOŞAN, DUMITRU DUMITRESCU

Abstract. An optimal clusterization model is introduced and studied - an approach that combines an evolutionary algorithm by the principles of the physical spin systems. The method is used to investigate the process of coalition formation that appears in complex systems. The numerical experiments show that the proposed hybrid model is able to detect the optimal clusterization in small and large systems by a reasonable cost of complexity (seen in terms of time and physical computational resources).

1. Introduction

Almost all interesting processes in nature are highly cross-linked. In many systems, however, we can identify the elements that interact to form compound structures or functions. The interconnected simple elements can form a complex system if they, together, exhibit a high degree of complexity from which emerges a higher order behaviour. Examples of complex systems include ant-hills, ants themselves, human economies, climate, nervous systems, cells and living things, including human beings, the brain, the immune system, the metabolic networks, the economic markets, and the human social networks, as well as modern energy or telecommunication infrastructures, the Internet and World Wide Web. Therefore, a complex system is any system featuring a large number of interacting components, whose aggregate activity is nonlinear and typically exhibits hierarchical self-organization under selective pressures.

More formally, a complex system is any system featuring a large number of interacting components (agents, processes, etc.) whose aggregate activity is nonlinear (not derivable from the summations of the activity of individual components) and typically exhibits hierarchical self-organization under selective pressures [20].

Received by the editors: October 9, 2007.

2000 *Mathematics Subject Classification.* 82B20, 68T20.

1998 *CR Categories and Descriptors.* code C.4 [**Performance of systems**]: – *Modeling techniques*; code I.6.1 [**Simulation Theory**]: – *Systems theory* .

An interesting problem that appears in such complex systems is the process of coalition formation. The optimization of several models proposed for studying the coalition formation in complex system (politics, economics or sociological systems) are based on simulated annealing [18, 16, 8] or on extremal optimization [4, 6] methods. Both optimization techniques are rather time-consuming and the necessary computational time increases strongly with the system size. The evolutionary techniques could overtake this weakness. It will be shown, in this paper, that the evolutionary methods can simulate very well the true dynamic of such complex systems and they allow analysing the phase transition from the viewpoint of the much-discussed social percolation [24], where the emergence of a giant cluster is observed in many social phenomena.

Physical concepts might prove useful in describing collective social phenomena. Indeed models inspired by statistical physics are now appearing in scientific literature [22]. The process of aggregation among a set of actors seems to be a good candidate for a statistical physics like model [21]. These actors might be countries, which ally into international coalitions, companies that adopt common standards, parties that make alliances, individuals that form different interest groups, and so on. Given a set of actors, there always exists an associated distribution of bilateral propensities towards either cooperation or conflict. The question then arises as to: *How to satisfy such opposing constraints simultaneously?*. In other words, *what kind of alliances, if any, will optimize all actor bilateral trends to respectively conflict or cooperation?*

It turns out that a similar problem does exist in spin glasses (as Ising or Potts models [11, 12, 14]). For these systems, the magnetic exchanges are distributed randomly between the ferro and anti-ferromagnetic couplings. Indeed such an analogy has been used in the past in a few models [21].

The aim of this paper is to provide a new hybrid model in order to investigate the process of coalition formation that can appear in a complex system. Coalition setting among a set of actors is studied using concepts from the theory of spin glasses and from the theory of evolution. Those of evolutionary computation combine the principles of the Potts model. Unlike other solutions proposed until now in order to study the dynamic of such complex system (simulated annealing or Monte Carlo methods), the proposed model is able to deal with large systems and helps us to investigate different phenomena that appear in such systems. The numerical results indicate that the hybrid approach is able to identify the characteristics of the coalition formation process.

The rest of the paper in organised as follows. A short description of the coalition formation problem (and its real implications) is given in Section 2. The Potts model is briefly described in Section 3. Section 4 proposes the

hybrid evolutionary approach for investigating the process of coalition formation. Several numerical experiments are presented and discussed in Section 5. Finally, the last section concludes the paper.

2. Problem formulation

Often-cited examples of complex systems in nature and society include the gene networks, the immune networks that preserve the identity of organisms, the social insect colonies, the neural networks in the brain that produce intelligence and consciousness, the ecological networks, the social networks comprised of transportation, utilities, and telecommunication systems, as well as the economies [20].

The field of complex systems seeks to explain and uncover common laws for the emergent, self-organizing behaviour seen in complex systems across disciplines. Many scientists also believe that the discovery of such general principles will be essential for creating artificial life and artificial intelligence [20]. Complex systems, as their name implies, are typically hard to understand. Traditionally the more mathematically oriented sciences such as physics, chemistry, and mathematical biology have concentrated on simpler model systems that are more tractable via mathematics. The rise of interest in understanding general properties of complex systems has paralleled the rise of the computer, because the computer has made it possible for the first time in history to make models of complex systems in nature that are more accurate.

In recent years, there has been a strong upsurge in the study of networks in many disciplines, ranging from computer science and communications to sociology and epidemiology. Some of the areas can profit from the application of complex systems modelling research and development: computational biology (DNA sequencing, micro-array data analysis, genetic regulatory networks, models of genetic regulatory processes), social systems (social networks, decision processes and knowledge structures of multi-agent systems, economic and financial markets modelling, homeland defence and intelligence community), distributed knowledge systems(information retrieval, web technology and digital libraries, knowledge integration), optimization, local search methods, extremal optimization, combinatorial optimization in biology, evolutionary systems (evolutionary algorithms, cellular automata and artificial life).

A network (or graph) is simply a collection of nodes (vertices) and links (edges) between nodes. The links can be directed or undirected, and weighted or un-weighted. Many natural phenomena can be usefully described in network terms.

Recent work [28, 3] have emphasized the importance of "network thinking" in dealing with complex systems in the real world. The purpose of characterizing networks according to degree distribution, clustering coefficient and average path length, is both to better understand networks from a scientific point of view and to develop better technologies for designing and managing networks in desired ways.

Another important application of network analysis is the problem of finding clusters, or community structures, in a given network. This is the problem of finding sub-networks ("communities") that contain dense interconnections, meaning that nodes in a given community are much more likely to be related to nodes in that same community than to nodes in other parts of the network. Finding such communities is related to the problem of graph partitioning in computer science, and to the problem of hierarchical clustering in sociology.

A complex system can be reduced to a full-connected graph this time in order to investigate the clusterization phenomena that appear in these systems. The following clusterization problem is considered: we have a complete graph on n vertices (items), where each edge (u, v) is labelled either $+$ or $-$ depending on whether u and v have been deemed to be "similar" or "different". The notion of similarity of two vertices could be understood as a propensity for cooperation or as a relation of sympathy or agreement between the two nodes. The goal is to produce a partition of the vertices (a clustering) that agrees as much as possible with the edge labels: a clustering that maximises the number of nodes that collaborate within clusters and that minimises, at the same time, the number of nodes that have antipathy relations

3. The Potts model

Understanding human thinking and learning has always been a great challenge for all the scientists and not only. The challenge has taken another dimension as scientists are trying to simulate the learning and thinking processes by using the computers and other devices. The nature inspired and Physics models have been of great help.

Even though very simple, the Ising model and its generalization, the Potts model, have been applied successfully in several computational problems. In its original form, the Ising model describes the evolution of a grid of up and down spins over time. Each spin can change its orientation in time, according to the external temperature and the values of its orthogonal neighbours [1]. The Potts system involves similar dynamics for the spins, but each spin can have more than two (up and down) orientations.

A simple version of a spin glass [19] consists of a d-dimensional hyper-cubic lattice with a spin variable $\sigma_i \in \{-1, 1\}$ placed on each site i, $1 \leq i \leq n$. A

spin is connected to each of its neighbours j via a bond variable $J_{i,j}$ drawn from some distribution $P(J)$ with zero mean and unit variance [21].

The infinite-range p-state Potts glass is usually defined by the Hamiltonian: $H(\sigma) = -p\sum_i \sum_j J_{ij}\delta_{\sigma_i\sigma_j}$, where the $\sigma(i)$ Potts states can take the $0, 1, 2, \ldots, p-1$ values. The sum is extended over all $N(N-1)/2$ pairs and $\delta_{mn} = 1$ if $m = n$ and $\delta_{mn} = 0$ otherwise. The J_{ij} bonds are randomly distributed quenched variables with J_0/N mean, and the variance is presumed to scale as N^{-1}. The system is non-trivially frustrated and computing the thermodynamic parameters is a complex task. The above model has been extensively studied by many authors through different methods [11, 14]. The main idea of these models is to find the "ground states", *i.e.*, the lowest energy configuration S_{min} of the Hamiltonian.

Neda et al. have considered a model resembling the infinite-range Potts glass [21], which can be useful for considering the optimal clusterization problem or for understanding the coalition formation phenomena in sociological systems. A difference to the Potts glass is that now the variance of the J_{ij} bonds scales as N^{-2}. The authors [21] have considered an unrestricted number of Potts states ($p = N$), and limit the study on the ground state ($T = 0$).

Therefore, this non-trivial optimization problem can be mathematically formulated resembling a zero-temperature Potts glass type model. To prove this, a cost-function, K, (a kind of energy of the system) has been defined. This function has been increased by $S_iS_j|Z_{ij}|$ whenever two conflicting actors (i and j) are in the same coalition or two actors which have a tendency towards collaboration are in different coalition. The cost-function is zero, when no propensity is in conflict with the formed coalitions. The number of possible coalitions is unrestricted (the maximal possible number is N), and the coalition in which actor i is denoted by $\sigma(i)$ [21]. The cost function then writes as

$$K = -\sum_{i<j} \delta_{\sigma(i)\sigma(j)} Z_{ij} S_i S_j + \frac{1}{2}\sum_{i<j}(Z_{ij}S_iS_j + |Z_{ij}S_iS_j|) \tag{1}$$

The order parameter considered by Neda et al. in [21] has been the relative size r of the largest cluster:

$$r = \left\langle \overline{max_i \left\{ \frac{C_x(i)}{N} \right\}} \right\rangle_x , \tag{2}$$

where $C_x(i)$ stands for the number of elements in state i for an x realization of the system [21].

4. The evolutionary-based coalition model

Evolutionary algorithms (EAs) have been successfully applied in various domains: mathematics, engineering, chemistry, physics, medicine, etc. The great advantage consists of their ability to obtain more solutions in a single run due to their capacity to deal with a population of solutions.

These algorithms have been introduced in 1965 by John Holland [15]. Many surveys in EAs and their applications can be found [7, 9, 13]. The EAs use a population of feasible solutions. The population is randomly generated initially over the search space, which is the definition domain. These solutions (called also chromosome, individuals) are improved by applying genetic operators (like selection, mutation, crossover, etc.). Each individual from the population is evaluated based on its fitness function. The best individuals are selected at each generation by using this quality function. Many selection mechanisms have been implemented [2, 13]. The chosen individuals are modified by applying the crossover and/or mutation operator. Various forms of these operators can be found [10, 23, 25, 26]. New solutions are obtained in this way. Some of these new solutions can be better than the existing ones. There are many modalities to accept the new solutions (also called offspring) in population. Some algorithms accept the new solution only if this solution is better than his parent (or parents).

4.1. **Motivation.** Evolutionary Computation (EC) methods allow a quickly and non-restrictive optimization, which is so useful in order to model the complex systems. Why? Because the nature solved many problems, so any algorithm showing the same behaviour might be good. EC can also handle non-linear, high dimensional problems without requiring differentiability or explicit knowledge of the problem structure. In addition, the evolutionary algorithms (EAs) are very robust to time-varying behaviour, even though they may exhibit low speed of convergence.

All these strengths of the EAs allow investigating the process of coalition formation in complex systems: the appearance of social percolation and the emergence of a giant cluster that is observed in many social phenomena.

For implementing a realistic dynamics for coalition formation one should also take into account that coalitions are not formed instantaneously and simultaneously. Once an agent is assigned to a coalition, it can (and probably will) change its propensities towards other agents. Agents will adjust their propensities according to the already formed coalitions and this feedback presumably reduces the frustration in the system.

4.2. **Representation.** A hybrid model that uses the GAs in order to evolve the realizations $\sigma(i)$ of a network configuration who's energy reaches a minimal

value is proposed. Actually, each GA individual is a fixed-length string of genes. The length of a chromosome is equal to the number of nodes from the network. Thus, a GA chromosome represents a possible clusterization of the network nodes in order to form an optimal coalition. Because the maximal number of coalitions (or clusters) is equal to the number of network's nodes, each gene is associated to the index of such a cluster. Therefore, each gene is an integer number from $\{1, 2, \ldots, N\}$ set (where N represents the number of nodes from the network). Or, in terms of the Potts model, each gene g_i from a GA chromosome is associated to a Potts state $\sigma(i)$.

For instance, for a network with $N = 5$ nodes (actors, elements) two possible chromosomes could be:

a) $C_1 = (1, 2, 3, 4, 5)$ - this chromosome encodes a clusterization where each group contains just an actor: $g_i \neq g_j$ or $\sigma(i) \neq \sigma(j)$, $i = 1, 2, \ldots N$ (see Figure 1(a));
b) $C_2 = (1, 1, 3, 1, 5)$ - this chromosome encodes a coalition with 3 clusters (see Figure 1(b));

FIGURE 1. Two possible configurations of the network in which: (a) each group is only formed by an element (it corresponds to the chromosome C_1); (b) one group contains 3 nodes (n_1, n_2, n_4) and the other two groups contain 2 nodes each (n_3 and n_5, respectively) – it corresponds to the chromosome C_2. A dashed edge between two nodes i and j means that there is no sympathy (no tendency for cooperation) between these nodes and a solid edge means that the nodes i and j tend to collaborate. The value associated to each node represents the index of the cluster where that node is placed.

4.3. **Initialisation.** Regarding the chromosome initialization each gene of a chromosome is initialized with a random value from the {1, 2, ..., N} set; in this case two or more nodes could take part to the same coalition from the start of the search process (like in the previous example from Figure 1(b)).

4.4. **Fitness assignment.** The array of integers encoded into a GA chromosome represents the structure of a coalition. In order to compute the quality of a coalition, the cost function proposed by Neda et al. [21] has been used. The simple case when $S_i = S_j = 1$ and $Z_{ij} = +1$ with a probability q and -1 with a probability $1 - q$ has been considered:

$$f = -\sum_{i,j=1}^{N} Z_{ij} \times \delta_{g_i,g_j}, \text{ where: } Z_{ij} = \pm 1, \text{ and } \delta_{g_i,g_j} = \begin{cases} 1, & \text{if } g_i = g_j \\ 0, & \text{if } g_i \neq g_j \end{cases} \quad (3)$$

A lower value of this function indicates a better quality. Therefore, the GA has to solve a minimization problem.

For instance, the chromosome C_2 (Figue 1(b)) is better than the chromosome C_1 (Figure 1(a)) because:

- $Fitness(C_1) = 0$ (because all Z_{ij} are -1 and all δ_{g_i,g_j} are 0);
- $Fitness(C_2) = -3$.

4.5. **Search operators.** The search operators mainly used within the GA are the crossover and mutation. Note that the action of the genetic operators does not change the structure of the network (the interactions between the actors). The crossover and the mutation change the coalitions only, which are formed in the system.

4.5.1. *Crossover.* By crossover, two selected parents are recombined. For instance, within the cutting-point recombination, two possible coalitions (one from each parent) exchange the elements placed between the cutting-points. A cutting point is considered within the following two parent chromosomes after the third gene. The offspring provided by the recombination operation are:

$$P_1 = (\ 1,\ 2,\ 3,|4,\ 5\ ) \qquad O_1 = (\ 1,\ 2,\ 3,\ 2,\ 4\ )$$
$$P_2 = (\ 3,\ 2,\ 1,|2,\ 4\ ) \qquad O_2 = (\ 3,\ 2,\ 1,\ 4,\ 5\ )$$

In Figure 2 is presented the network-based visualization of this crossover operation.

4.5.2. *Mutation.* By mutation, some information inside a chromosome could be changed. In other words, some of the network actors could change their group affiliation. Therefore, by mutation, a gene change its value into another one (off course, from the same discrete domain $\{1, 2, \ldots, N\}$).

FIGURE 2. Network-based crossover. Remark that the crossover operation does not change the tendency for cooperation of the nodes from the network. The cluster affiliation of some nodes is only modified.

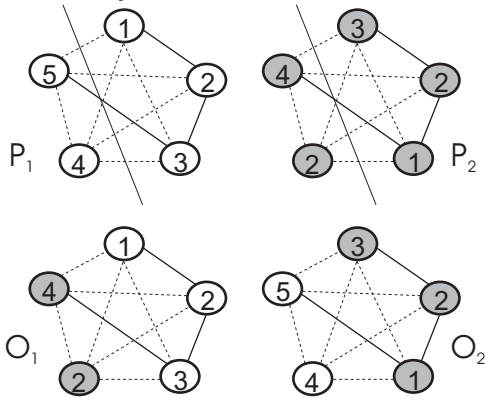
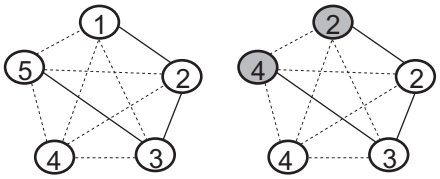


FIGURE 3. Network-based mutation. Note again that the network structure is not changed.



4.6. **The algorithm.** A GA [13] is used in order to evolve the coalition formation. The steady-state evolutionary model [27] is used as an underlying mechanism for the GA implementation. The algorithm starts by creating a random population of individuals. The following steps are repeated until a given number of generations is reached: two parents are selected by using a binary tournament selection procedure. The parents are recombined in order to obtain two offspring by performing a one cutting-point crossover. The offspring are considered for mutation. The best offspring O replaces the worst individual W in the current population if O is better than W.

5. NUMERICAL EXPERIMENTS

In this experiment the dynamic of the coalition formation through the order r parameter (like in [21]) is studied in full connected networks (there is a positive connection - a sympathy - or a negative connection - an antipathy -

between every 2 nodes of the network). In addition to this study, the results are compared to those computed by using Monte Carlo methods for small systems (up to $N \leq 10$). Several numerical results are presented also for larger systems (e.g. $N = 100$ or $N = 150$).

5.1. **Experiment 1.** The GAs are used in order to obtain the optimal coalition formation for small systems in which an exact enumeration is possible. The exact enumeration means that one can computationally map the whole phase-space (all $\sigma(i)$ realizations) for a generated Z_{ij} configuration and determine the minimum energy state. The order parameter considered is the relative size r of the largest cluster.

Moreover, for $N \leq 7$ it was also possible to map all the Z_{ij} configurations as well. The results from [21] up to $N \leq 7$ are thus exact. In the $7 < N \leq 10$ interval, although the minimum energy states are exactly found, due to greatly increased computational time and memory needed, it was possible to generate only a reasonable ensemble averaged for Z_{ij} (5000 configurations) [21]. The results obtained by the evolutionary approach proposed in this paper up to $N \leq 10$ are averaged over the same reasonable ensemble of 5000 network configurations.

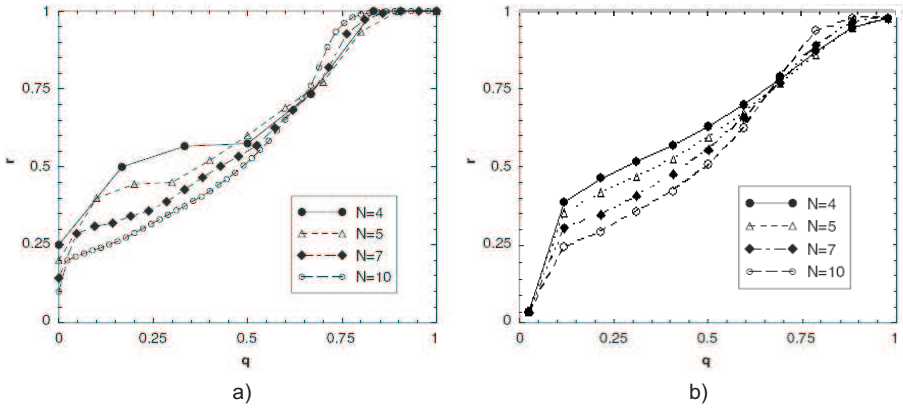
Note that for each structure of the network Z_{ij}, $i, j \in \{1, 2, \ldots, N\}$ a GA is run in order to find the optimal coalition formation. Therefore, the results presented here are the corresponding r values for the best solutions found by the evolutionary algorithm in the last generation. Moreover, in order to obtain a realistic approach, the results are averaged over all 5000 network configurations.

In this experiment, the GA works by 100 chromosomes that are evolved during 100 generations. The crossover and the mutation operators are applied by $p_c = 0.8$ and $p_m = 0.1$, respectively, probabilities.

The comparison exact enumeration is performed of two purposes. First, the trends of the $r(q)$ curves as a function of increasing system size is checked. Secondly, these results offer a good "standard" for the proposed optimisation method, used for larger system sizes (in the next experiment). As the results in Figure 4 show, the $r(q)$ curves have a similar trend as those suggested by Neda et al. in [21], i.e., as the system size increases, the slopes for $r(q)$ are increased around a non-trivial q value.

The GA results are in perfect agreement with the ones from exact enumerations [21], giving confidence in to use evolutionary optimisation methods. In addition, the complexity of the proposed approach is smaller than the complexity of the traditional methods, which have been applied in order to investigate the coalition formation process. Therefore, the time that is needed in order to identify the optimal clusterization of the "actors" in such system by the

FIGURE 4. Results of the dependence of the order parameter as a function of q for different sizes of the network (N). For comparison purposes on (a) the exact enumeration results are shown and on (b) the GA optimisation results.



evolutionary methods is reduced. Another, and maybe the most important advantage of the proposed hybrid approach is the given by its ability of handle non-linear, high dimensional problems without requiring differentiability or explicit knowledge of the problem structure. This characteristic is very important and it favours a new direction in studying the phenomena that appear in very large complex systems.

5.2. **Experiment 2.** In [21] the authors have considered two Monte Carlo optimisation methods: the classical simulated annealing [17] and the recently proposed extremal optimization method [4, 5]. Both approaches are rather time-consuming and the necessary computational time increases sharply with system size. The computational resources allowed the authors to study only the systems by sizes up to $N \leq 60$.

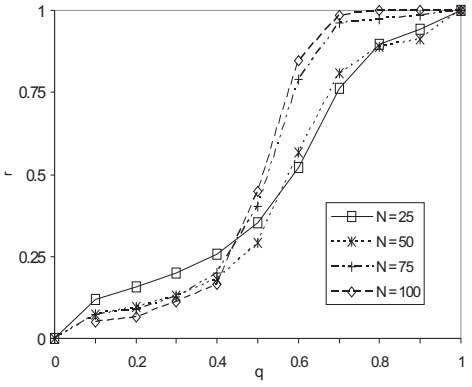
The evolutionary approach proposed in this paper is time-consuming also, but the computational resources allow investigating the coalition formation in systems by larger size. Therefore, the evolution of the order parameter $r(q)$ is analysed in four large systems: $N = 25$, $N = 50$, $N = 75$ and $N = 100$ with a statistic of 100 realisations.

The GA parameters used in these experiments are presented in Table 1. Even if, for the large systems, large populations are evolved during more generations than those used for small systems, the computational time that is needed in order to obtain good solutions is reasonable.

TABLE 1. GA parameters

N	#Generations	Population Size
25	100	100
50	100	200
75	2000	500
100	5000	5000

FIGURE 5. Results of the dependence of the order parameter as a function of q for large systems. The optimal values of the order parameter r are obtained by the evolutionary approach.



The relationship of the order parameter as a function of probability q are presented in Figure 5. From the numerical results presented in Figure 5 several aspects can be remarked:

- when in the system there are more relations of collaboration than the conflict ones ($r \to 1$), usually the nodes tend to form a single cluster in order to satisfy the conflicting interactions.
- when the tendency for collaboration is the same with that of conflict, the order parameter is changing strongly with small variations of q. Therefore, in this case, the process of coalition formation is sensitive to the structure of relations within the network.
- when in the system there are more relations of conflict than the collaboration ones ($r \to 0$), usually the nodes tend to form a large number of clusters. For very small values of q (for very few relations of collaborations), the nodes tend to form the one's own cluster (the number of clusters tends to be equal to the number of elements from the network).

6. Conclusions

A hybrid evolutionary framework has been proposed in this paper in order to study the process of coalition formation that appears in complex systems. Two types of full connected networks have been investigated: small networks (up to 10 nodes) and large networks (from 10 up to 100 nodes), which are closer to the real systems than the smaller ones. The numerical results obtained in both cases indicate the relationship between the number of coalitions and the structure of the network.

Future works will be focused on the study of the coalition formation in full connected networks in which the relations (of collaboration or conflict) between the elements are weighted (instead to have only $+1$ or -1 links between 2 nodes, some fuzzy relations will be defined on $[-1, 1]$ range). The optimal clusterization will be also investigated in networks that are more sophisticated: random networks, small world networks, and scale-free networks.

Acknowledgements

The present study was supported by the National Research Grant "Developing and optimisation of hybrid methods based on evolutionary techniques. Applications for NP-complete optimisation problems" – CNCSIS, Romania. We also thank to prof. Zoltan Neda for his interesting discussion.

References

[1] Bak, P. *How nature works: The science of self-organized criticality.* Springer-Verlag, 1996.

[2] Baker, J. E. Adaptive selection methods for genetic algorithms. In *Proceedings of the First International Conference on Genetic Algorithms and their Applications (ICGA'85)* (1985), J. J. Grefenstette, Ed., Lawrence Erlbaum Associates, pp. 101–111.

[3] Barabási, A. L. *Linked: The New Science of Networks.* Perseus, 2002.

[4] Boettcher, S., and Percus, A. Nature's way of optimizing. *Artificial Intelligence 119*, 1–2 (2000), 275–286.

[5] Boettcher, S., and Percus, A. G. Extremal optimization for graph partitioning. *CoRR cond-mat/0104214* (2001).

[6] Boettcher, S., and Percus, A. G. Optimization with extremal dynamics. *Physical Review Letters*, 86 (2001), 5211–5214.

[7] BOX, G. E. P. Evolutionary operation: a method for increasing industrial productivity. *Appl. Statist. 6*, 2 (1957), 81–101.

[8] Cerny, V. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45 (1985), 41–51.

[9] Davis, L. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[10] Eshelman, L. J., Caruna, R. A., and Schaffer, J. D. Biases in the crossover landscape. In *Proc. of the Third Int. Conf. on Genetic Algorithms* (1989), J. D. Schaffer, Ed., Morgan Kaufmann, pp. 10–19.

[11] ET AL, A. E. The infinite-ranged potts spin glass model. *J. Phys. C: Solid State Phys.*, 16 (1983), 555–560.
[12] ET AL, D. E. The curious case of the potts spin glass. *J. Phys. C: Solid State Phys.*, 16 (1983), 497–503.
[13] GOLDBERG, D. E. *Genetic algorithms in search, optimization and machine learning.* Addison Wesley, 1989.
[14] GROSS, D. J., KANTER, I., AND SOMPOLINSKY, H. Mean-field theory of the potts glass. *Phys. Rev. Lett. 55*, 3 (1985), 304–307.
[15] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.
[16] KIRKPATRICK, S., GELATT, C. D., AND VECCHI, M. P. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983 220, 4598* (1983), 671–680.
[17] KIRKPATRICK, S., GELATT, C. D., AND VECCHI, M. P. Optimization by simulated annealing. *Science 220*, 4598 (1983), 671–680.
[18] METROPOLIS, N., ROSENBLUTH, A., ROSENBLUTH, M., TELLER, A., AND TELLER, E. Equations of state calculations by fast computing machines. *Journal of Chemical Physics 6*, 21 (1953), 1087–1092.
[19] MÉZARD, M., PARISI, G., AND VIRASORO, M. A. *Spin Glass Theory and Beyond.* World Scientific, 1987.
[20] MITCHELL, M. Complex systems: Network thinking. *Artif. Intell 170*, 18 (2006), 1194–1212.
[21] NÉDA, Z., FLORIAN, R., RAVASZ, M., LIBÁL, A., AND GYÖRGYI, G. Phase transition in an optimal clusterization model. *Physica A Statistical Mechanics and its Applications 362* (2006), 357–368.
[22] S. MOSS DE OLIVEIRA, P. D. O., AND STAUFFER, D. *Evolution, Money, War, and ComputersNon-Traditional Applications of Computational Statistical Physics.* Teubner, 1999.
[23] SCHAFFER, J. D., AND MORISHIMA, A. An adaptive crossover distribution mechanism for genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (1987), Lawrence Erlbaum, pp. 36–40.
[24] SOLOMON, S., WEISBUCHA, G., DE ARCANGELISC, L., JANC, N., AND STAUFFER, D. Complex systems: Network thinking. *Physica A: Statistical Mechanics and its Applications 277*, 1-2 (2000), 239–247.
[25] SPEARS, W. M., AND JONG, K. A. D. On the virtues of parameterised uniform crossover. In *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA)* (1991), R. K. Belew and L. B. Booker, Eds., Morgan Kaufmann, pp. 230–236.
[26] SYSWERDA, G. Uniform crossover in genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms* (1989), Morgan Kaufmann, pp. 2–9.
[27] SYSWERDA, G. A study of reproduction in generational and steady state genetic algorithms. In *Proceedings of Foundations of Genetic Algorithms Conference* (1991), G. J. E. Rawlins, Ed., Morgan Kaufmann, pp. 94–101.
[28] WATTS, D. J. *Six Degrees: The Science of a Connected Age.* W. W. Norton, New York, 2003.

DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF MATHEMATICS AND COMPUTER SCIENCE, BABEŞ-BOLYAI UNIVERSITY, CLUJ-NAPOCA, ROMANIA
E-mail address: lauras,ddumitru@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LII**, Number 2, 2007

TOWARDS AN UTILITY-BASED TCP-FRIENDLY RATE CONTROL

STERCA ADRIAN

Abstract. We present a method for constructing TCP-friendly rate controls that are at the same time media-friendly. These types of rate controls are more suitable for multimedia streaming application than the classical TCP rate control. The method is developed by combining the notion of TCP-friendliness with a general optimization framework for bandwidth sharing in computer networks.

1. TCP-Friendliness

During the years 1980 when the Internet (Arpanet) grew from tens of computers to thousand of computers, Internet researchers noticed that Internet's core transport protocol, namely TCP, can not handle in an efficient manner the growing number of connections. Because the process of sharing bandwidth amoung users was not strongly regularized at the TCP level, the phenomenon of congestion collapse occured, reducing drastically the utility of the network. To avoid the occurence of congestion collapse, TCP incorporated Jacobson's AIMD (*Additive Increase Multiplicative Decrease*) congestion control algorithm [9] which increases the send rate by one packet per RTT in the absence of congestion indications and decreases the send rate by half when congestion does occur. Chiu and Jain proved in [1] that a simple AIMD congestion control algorithm like the one employed by TCP converges to a fair and efficient equilibrium state when the congestion feedback is received at the same time by all flows sharing the network and all flows react to it together synchronously. The fairness criterion towards which an AIMD algorithm converges, in the aforementioned conditions, is *max-min fairness* [4]. However, in real world conditions, different flows don't react to congestion synchronously and don't receive network feedback in the same time. Consequently, TCP

Received by the editors: October 20, 2007.

2000 *Mathematics Subject Classification.* 90B18, 68M20.

1998 *CR Categories and Descriptors.* 90B18 [**Operations research, mathematical programming**]: Operations research and management science – *Communic ation networks*; 68M20 [**Computer science**]: Computer system organization – *Performance evaluation; queueing; scheduling*;

doesn't reach maximum efficiency in practice and is approximately fair only across flows having the same RTT and the same congestion measure which doesn't happen in the real world.

Another significant characteristic of TCP is that it treats flows having the same RTT and sharing the same bottleneck link identically because it aims at max-min fairness. TCP does not distinct between elastic applications (i.e. applications which can tolerate bandwidth fluctuations, e.g. file transfer applications) and inelastic applications (i.e. applications having strict bandwidth requirements because of real-time constraints, e.g. multimedia streaming applications). There are several characteristics of TCP that makes it rather unsuitable for multimedia streaming applications. First of all, by implementing congestion control and guaranteed retransmission, TCP trades timeliness over reliability: it is more important the data arrives safely and in-order than it is to arrive in time (i.e., bandwidth is sacrificed for retransmissions). This philosophy is counterproductive for multimedia streams, for which timeliness is more important than reliability. Secondly, TCP's congestion control algorithm determines a steep variation in the sending bitrate, a variation that is not well coped with by current codecs. Steep degradations in the sending bitrate of a multimedia stream has very bad consequences on the quality perceived by the final receiver.

In an effort to steer the development of a congestion control mechanism for multimedia streaming, the scientific community has advertised the notion of TCP-friendly flow [6] as a flow which receives, on average, approximately the same bandwidth as a TCP flow under the same network traffic conditions. When the packet loss rate, p, is smaller than 0.3, the transmission rate of such a TCP-friendly flow should approximately be [6]

$$X = \frac{1.5 * \sqrt{\frac{2}{3}}}{RTT * \sqrt{p}} \text{ packets/second} \tag{1}$$

where RTT is the round-trip time and p is the packet loss rate this flow sees. [7] presents an equation which characterizes more accurately the throughput of a TCP flow, because it takes into account retransmission timeouts and doesn't restrict p to values smaller than 0.3. However we do not use this equation in our study because it is difficult to invert it.

2. Rate pricing

A different approach in sharing bandwidth among competing applications is taken in [2,3,4] where each application has a bandwidth utility function and bandwidth sharing is done in such a way that it maximizes the sum of all users'

utility functions. The problem of bandwidth allocation among flows reduces to finding the solution to the following concave optimization problem:

$$(2) \quad \begin{cases} \max_{x>0} \sum_{s\in S} U_s(x_s) & x = (x_1, ..., x_n), S = \{s_1, ..., s_n\} \\ \text{subject to:} \sum_{s\in s(l)} x_s \le c_l & \forall l \in L \end{cases}$$

In this model the network is abstracted as a set of links $l \in L$ and each link l has the capacity c_l. The network is shared by sources $s \in S$ and each source s transmits data at rate x_s. When the source s sends data at rate x_s, it gets a utility $U_s(x_s)$ which is assumed to be a concave function twice differentiable. Also, let $S(l)$ denote the set of sources which use link $l \in L$ and $L(s)$ the set of links that source s uses.

Problem (2) is hard to solve in a decentralized way because of the coupling of transmit rates of sources at links in the inequality constraints of the problem. Instead of looking at this problem, the dual problem is considerred. Let the Lagrangian for problem (2) be [3]

$$\begin{aligned} L(x,p) &= \sum_{s\in S} U_s(x_s) - \sum_{l\in L} p_l \left(\sum_{s\in S(l)} x_s - c_l \right) \\ &= \sum_{s\in S} \left(U_s(x_s) - x_s \sum_{l\in L(s)} p_l \right) + \sum_{l\in L} p_l c_l \end{aligned}$$

where p is the Lagrange multiplier associated with the inequality constraints of problem (2). p is a vector of prices p_l, one for each link l, where p_l is interpreted as the price per unit bandwidth at link $l \in L$. Because the first term in the Lagrangian is separable in x_s, so we have

$$\max_{x_s>0} \sum_{s\in S} \left(U_s(x_s) - x_s \sum_{l\in L(s)} p_l \right) = \sum_{s\in S} \max_{x_s>0} \left(U_s(x_s) - x_s \sum_{l\in L(s)} p_l \right)$$

the objective function of the dual problem is [3]:

$$(3) \quad D(p) = \max_{x_s>0} L(x,p) = \sum_{s\in S} B_s(p^s) + \sum_{l\in L} p_l c_l$$

$$\text{where} \quad \begin{aligned} B_s(p^s) &= \max_{x_s>0}(U_s(x_s) - x_s p^s) \\ p^s &= \sum_{l\in L(s)} p_l \end{aligned}$$

Applying the Karush-Kuhn-Tucker theorem to find x_s which maximizes the Lagrangian $L(x,p)$, the solution is [3]:

$$(4) \quad x_s(p^s) = U_s'^{-1}(p^s)$$

where $U^{'-1}$ is the inverse of $U_s^{'}$. For x_s from (4) to be the unique maximizer that solves problem (2), p must be a Lagrange multiplier that satisfies the complementary slackness condition [5, prop. 3.3.4]. In practice, we use as p^s the loss event rate of TCP which satisfies with approximation the complementary slackness condition.

3. Mixing TCP-Friendliness with Rate pricing

We present in this section a method for finding a congestion control algorithm suitable for multimedia streaming applications by combining the TCP-friendly model with the Rate pricing model. More specifically, we first **a) derive the utility function of the system which is maximized by the solution of the TCP-friendly equation (1)**, then we **b) modify this utility function we have obtained to be more media specific (or media-friendly)** and then we **c) compute backwards using relation (4) the solution to the new optimization system**. In the rest of our calculus we will use equation (1) for characterization of TCP-friendliness and not the equation proposed by [7] because it is very difficult to invert the latter. To get an ideea of the dificulties involved in inverting TCP's equation from [7] please see [8].

In order to derive the utility function of the optimization system for which the TCP-friendly equation is a solution, we equalize the TCP-friendly equation (1) with the equation of the optimization system's solution, i.e. equation (4),

$$x_s(p) = \frac{1.5 * \sqrt{\frac{2}{3}}}{RTT * \sqrt{p}} = U_s^{'-1}(p^s)$$

By inverting the function from the right-hand side of the equation, we get

$$U_s^{'}(x_s) = \frac{1.5^2 * \frac{2}{3}}{RTT^2 * x_s^2}$$

and then

$$U_s(x_s) = \int \frac{1.5^2 * \frac{2}{3}}{RTT^2 * x_s^2} dx = -\left(\frac{1.5 * \sqrt{\frac{2}{3}}}{RTT}\right)^2 * \frac{1}{x_s} + k \quad \text{,k is a constant} \tag{5}$$

is TCP's utility function. By maximizing the utility function presented above we obtain *weighted minimum potential delay fairness* [4].

In the second step of our method, we modify TCP's original utility function obtained in (5) to be more media specific. We consider two versions of

media specific utility function based on (5):

$$U_s(x_s) = -\frac{b}{b_{avg}} * \left(\frac{1.5 * \sqrt{\frac{2}{3}}}{RTT} \right)^2 * \frac{1}{x_s} \tag{6}$$

$$U_s(x_s) = -\left(\frac{1.5 * \sqrt{\frac{2}{3}}}{RTT} \right)^2 * \frac{\sqrt{x}}{x_s} \tag{7}$$

where b is the multimedia stream's bitrate in the last second and b_{avg} is the multimedia stream's average bitrate.

If we solve using equation (4) the new optimization systems corresponding to the two utility functions depicted above, we get the following solutions:

$$x_s(p) = \frac{b}{b_{avg}} * \frac{1.5 * \sqrt{\frac{2}{3}}}{RTT * \sqrt{p}} \tag{8}$$

and

$$x_s(p) = \sqrt[3]{\frac{1.5^2 * \frac{2}{3}}{4 * RTT^2 * p^2}} \tag{9}$$

The utility function from (6) is media-friendly because it takes into account the bitrate demands of the stream (i.e. when the instant bitrate is above the average bitrate the application has a higher utility of bandwidth x_s) and the utility function from (7) is also useful for multimedia streaming applications because it tries to reduce fluctuations on the transmision rate x_s.

4. Conclusions and Future work

In this paper we have developed a method for obtaining rate controls that are TCP-friendly and in the same time media-friendly. We exemplified this method by two such rate control algorithms (equations (8) and (9)) which maximize an application-specific utility function. As future work we intend to test the two rate controls we developed here in varying network environments and to find more appropriate utility-functions for multimedia specific applications and use them to develop improved TCP-friendly and media-friendly rate controls.

References

[1] Chiu, D. M., Jain, R., *Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks*, Computer Networks and ISDN Systems 17, North-Holland, 1989.
[2] Kelly, F., Maulloo, A., Tan, D., *Rate control in communication networks: shadow prices, proportional fairness and stability*, Journal of the Operational Research Society 49 (1998), pp. 237-252.
[3] Low, S., Lapsley, D. E., *Optimization Flow Control I: Basic Algorithm and Convergence*, IEEE/ACM Transactions on Networking, Vol. 7, No. 6, Dec. 1999.
[4] Srikant, R., *The Mathematics of Internet Congestion Control*, Birkhauser, 2004.
[5] Bertsekas, D., *Nonlinear Programming*, Athena Scientific, 1995.
[6] Floyd, S., Fall, K., *Promoting the Use of End-to-End Congestion Control in the Internet*, IEEE/ACM Transactions on Networking, August 1999.
[7] Padhye, J., Firoiu, V., Towsley, D., Kurose, J., *Modeling TCP Throughput: A Simple Model and its Empirical Validation*, ACM SIGCOMM 1998, Vancouver.
[8] Sterca, A., *Finding TCP's utility function*, Technical Report, 2007 (available at http://cs.ubbcluj.ro/ forest/phd/Technical%20Reports).
[9] Jacobson, V., *Congestion Avoidance and Control*, SIGCOMM Symposium on Communications Architectures and Protocols, pp. 314-329, 1988.

Babes-Bolyai University, Faculty of Mathematics and Computer Science, Department of Computer Science, Cluj-Napoca
E-mail address: forest@cs.ubbcluj.ro