A Preliminary Formal Specification of Virtual Organization Creation with RAISE Specification Language

Mohammad Reza Nami
FacultyofMathematicsand ComputerScience DelftUniversityofTechnology, Netherlands,
nami@ce.et.tudelft.nl

Mohsen Sharifi
FacultyofComputerEngineering, IranUniversityofScienceand Technology, Iran
msharifi@iust.ac.ir

Abbas Malekpour
FacultyofComputerScience, UniversityofRostock, Germany
abbas.malekpour@uni-rostock.de

Abstract

Recently, several flavors of Formal Methods (FMs) have been gaining industrial acceptance and production quality software tools have begun emerging. Domain Engineering (DE) has been introduced as one of outstanding ideas in software development. It serves form al methods that provide a rigorous, mathematical based framework (domain model) for specifying, defining, and verifying systems in the software development. The increasing demands for extended products and services along with advances in IT industry have motivated researchers to create Virtual Organizations (VOs) in order to better respond to business opportunities and produce qualitative services and products. The purpose of this paper is to describe Virtual Organization and Domain engineering with introducing formal methods especially RAISE as a formal method that has been used successfully on diverse applications. Then, a primary formal model is presented using RAISE method and its language (RSL) formula for the improvement of VO creation process. Formal model presented in this paper is provable to implement. It improves reusability and reliability in such environments.

Keywords: *Software Engineering, Formal Methods, RAISE Method, Collaborative Networks, Virtual Organization.*

1. Introduction

Formal methods [11] provide a rigorous, mathematical based framework for specifying, defining, and verifying systems. These methods provide the basis for the precise definition of consistency, completeness, specification, implementation, and correctness. Formal method is the applied mathematics of computer systems engineering. The mathematics of formal methods includes predicate calculus (first order logic), recursive function theory, lambda calculus, programming language semantics, and discrete mathematics (e.g., number theory, abstract algebra). To these mathematical base, formal methods adds notions from programming languages such as data types, module structure, and generics [20]. On the other hand, a formal method in software development is a method that provides a formal language for describing a software artifact such as specifications, designs, source code, such that formal proofs are possible, in principle, about properties of the artifact so expressed. Domain engineering [7] [3] uses this method for developing reliable software for critical systems such as Air Traffic Control (ATC). New software engineering [7] for the development of safety critical systems includes three phases consisting of domain engineering, requirement engineering, and design and implementation. Due to the importance of efficient creation of dynamic VOs and lack of method, we present a formal model for developing provable, reliable, and reusable software. Rigorous Approach to Industrial Software Engineering (RAISE) [16] has been used in this paper as a strong FM that emphasizes the use of formal (mathematical) techniques in the development of software including requirements analysis and formulation, specification, design and development. RAISE offers the RAISE Specification Language (RSL) and the RAISE method. The lack of reliable software development approach for safety critical systems encouraged to serve FMs in software development. New software engineering idea was

0-7695-2867-8/07 $25.00 © 2007 IEEE
DOI 10.1109/SERA.2007.65

introduced. FMs and DEs have been used successfully in financial domain, railway domain, and ATC [4], [23], [17]. See also Craigen et al [15] and [11] for a description of 12 case studies in FMs. No one has presented domain formal model or formula in VOs. Therefore, this research can be novel. Today, many large scale software developments based on the domain/requirements/software design paradigm outlined has been systematically applied to the experimental development of software designs for the computing support of a number of diverse infrastructure components.

Beside, evolution of the Internet and rapid changes in customer demands for extended services and products have motivated organizations toward a new cooperation schema including geographically and legally organizations that collaborate to achieve the goal. This cooperation is supported by computer networks. Researchers have introduced concepts such as virtual company, virtual corporation, and Virtual Enterprise (VE) in the early 1990s, but definitions and concepts of VO/VE are still completed and grown [8]. Advances in computer networks also affected marketing and business systems so that traditional business systems have been metamorphosed. Virtual Organizations have been introduced as a new organizational schema including a temporary set of geographically organizations collaborating, sharing skills and resources to fulfill customer requests in a networked environment. Efficient creation of dynamic VOs requires a proper environment that the members of new VOs are selected in it according to their capabilities and trust [13] among them. For this, Virtual organization Breeding Environment (VBE) was introduced in [9]. The main goal of VBE is to improve the preparedness of its member organizations for efficiently creating VOs. To achieve this goal, VBE can even find some external organizations to join the VBE and select them as partners to establish a new VO [1].

This paper is organized as follows. In section 2, an overview of the RAISE formal method and the RSL including their capabilities and their advantages are discussed. Section 3 presents VOs and some issues concerning on VOs. In section 4, domain engineering of our domain including an abstract formal model of VO creation (formula) is presented. Finally, conclusions and further researches are presented.

2. An Overview of Formal Methods

This section presents an overview of formal methods with focusing on the RAISE formal method and the RSL.

2.1 Definitions

A formal method in software development is a method that provides a formal model to specify and model a system. From the formal methods point of view, DE describes and develops formal specifications at the beginning of the lifecycle and then automatically derives the source code for the system. Thereby, a formal domain model is developed. This model includes domain abstracts that describe specifications in formal form by formal specification languages such as Z, VDM, and RSL and informal form. The aim is to develop domain theorem by domain specification language [6]. Domain engineering covers all the activities for building software core assets [2]. These activities include identifying one or more domains, capturing the variation within a domain (*domain analysis*), constructing an adaptable design (*domain design*), and defining the mechanisms for translating requirements into systems created from reusable components (*domain implementation*). The products (or software assets) of these activities are domain model(s), design model(s), domain-specific languages, code generators, and code components. In new software engineering definition [7], DE is performed before requirement engineering and is based on FMs for enhancing software reliability [5].

2.2 RAISE Method

RAISE is a formal method that provides facilities for the industrial use of Formal Methods in the development of software systems. The aim of RAISE is to enable the construction of more reliable software with fewer errors, more easily maintainable software, and better documented software. In particular, RAISE supports development all the way to final compilable code. The entire development from specification through to implementation will be formally recorded when using RAISE, and this is the first prerequisite for optimal maintenance activities. Main properties of RAISE include abstraction, verification, stepwise refinement paradigm for software construction through a series of steps, where each step represents a refinement of the previous one. Editing, performing justifications (proof rules), translating into imperative languages, and document support are the RAISE tools for editing and manipulating a variety of entities that are relevant during a development process: modules,

relations between modules, theories, justifications and hints. In comparison with other FMs, RAISE has a distinguished property. The RAISE handles concurrency.

2.3 RSL Language

RSL is a wide-spectrum specification language. It is inspired by and unifies features of VDM [24], CSP [19], and ACT-ONE [10] specification languages. It is case sensitive. Some of advantages of RSL include carrying through the whole development in RSL, supporting decomposition and reuse, and integrating concurrency and serializability.RSL supports modularization with parameterization at the structuring level. The basic building block in RSL is the class expression and is often referred to as an *RSL module* . All declarations are made within a class expression. RSL specifications can be constructed in a hierarchical manner since modules can refer to the entities of other modules in defining their own entities. The modules can be parameterized thus supporting generalized specification. This is an important mechanism in creating reusable modules [22]. RSL also supports the following styles [22]:

- Model-oriented specifications as used in Z and VDM.
- Algebraic specifications as used in Larch and OBJ.
- Imperative specifications similar to the style used in imperative programming languages such as C and Pascal.
- Implicit definitions using pre- and post-conditions
- Applicative specifications similar to the style used in applicative programming languages such as Lisp and Miranda
- Concurrent specifications as used in CSP
- Explicit definitions

Unlike Z language [12] that is basically a specification language and is good for producing specifications, RSL can be used for both specification and development. RSL is more expressive than Z by virtue of its facilities for under specification and concurrency. Therefore, it is easier to translate a Z specification into an RSL specification than vice versa. RSL has been derived from VDM with added facilities such as concurrency, axiomatic and imperative styles. Modularity is also an integrated feature of the RSL language and RAISE method whereas it is not supported as part of standard VDM.

3. Virtual Organisation: An Introduction

There are three concepts used in this context: Virtual Organization (VO), Virtual Enterprise (VE), and Virtual organization Breeding Environment (VBE). In this section, a brief introduction of VO is presented.

3.1 Virtual Organization Definition

A VO is defined as a temporary coalition of reconfigurable, independent, networked, which geographically dispersed organizations including high level trust and competencies that collaborate and share their resources and competencies in order to fulfill the customer request. A VE is a subset of VO. R. Camacho and et al [8] define VO as a"new organizational scheme that requires high level agility in order to achieve a competitive advantage when a business opportunity is identified". L. M. Camarinha-Matos, H. Afsarmanesh and M. Ollus [9] explain the co-operation of independent organizations is supported by computer networks. Partners in a VO should collaborate in order to achieve business opportunities. Trust among them and operation according to a common agreement are essential things for collaborating. Networks or breeding environments are an appropriate context for effective creation of dynamic VOs. [1] has called this context as VBE and defined it as "an association of organizations and their related supporting institutes, adhering to a base long term cooperation agreement, and adoption of common operating principles and infrastructures, with the main goal of increasing both their chances and their preparedness towards collaboration in potential VOs". VBE can be local and global. Local VBE initiates dynamic VOs from organizations located in one geographical region while global VBE incorporates involved organizations from geographically distributed regions to effectively create VOs. A VBE should identify and obtain new business opportunities, know the competencies and capabilities of its members, then select an appropriate set of partners for creation of new VO. Of course, it is possible to VBE administrator searches and recruits new organizations as member for new VO. VBE also evaluates and coordinates created VO during its life cycle. Each VBE has also a life cycle. VOs are initiated within the VBEs. On the other hand, the aim of a VBE is to improve preparedness of the partners.

3.2 Virtual Organization Issues

Since VO is fixing as a master component of dynamic collaborative networks, there are different issues and challenges in VO creation, management, design, and implementation. We discuss some of them as follows.

3.2.1 VO Life cycle

As mentioned, a VO framework is created for responding to a business opportunity and terminated after completion. VO life cycle is formed from three stages. These stages include VO creation, VO operation, and VO dissolution. VO Creation identifies business opportunities, examines the partner competencies, selects partners from within or outside the VBE (network), forms the best partnership in terms of the competencies, creates the necessary databases, and registers new members. In VO operation, activities of the partners are integrated, competencies and common knowledge are managed, cooperation's are organized, and collaboration is improved. In VO dissolution, each VO has a limited lifetime. When a VO fulfills the business opportunity, it is dissolved. Of course, the partner competencies and relationships between the partners are kept in the network (VBE) knowledge.

3.2.2 VO Planning and Launching

VO planning activities include receiving and analyzing business opportunities, selecting proper partners, determining high level Work Breakdown Structure (WBS), and setting up VO. R. Camacho and et al [8] present a reference model for VO planning and launching. This model integrates the elements involved in VO creation in VO creation, modeling, and knowledge management dimensions. VO modeling is also created in four views: Resource, organization, functional, and Knowledge.

3.2.3 VO Management

As a VO is composed of different members located at dispersed sites, different issues can affect the VO. Therefore, the VO management must be examined in different aspects. We can categorize them in human issues and technical issues. For example, different experiences and cultures play the key role in collaboration. Beside, communication between the partners, trust among them, VO planning, and security are important challenges from technical point of view. Trust management [18], competency management [14], and security management [21] are main issues in VO management.

3.3 VBE Functionality

Efficient creation of dynamic VOs requires a proper environment that the members of new VOs are selected in it according to their capabilities and trust among them. The main goal of VBE is to improve the preparedness of its member organizations for efficiently creating VOs. To achieve this goal, VBE can even find some external organizations to join the VBE and select them as partners to establish a new VO. Each VBE serves a specific domain and attempts to select the best members to achieve its specific aims in the domain. Meanwhile, VBE Members can be different organizations such as business entities, ministries, legal service providers, and environmental organizations. These organizations should be registered at the VBE, accept the general VBE rules and policies, and have access to common information and tools for operation in a VO. A VBE member can have different roles in different VOs established that has been discussed in [1].

4. Presenting a New Formal Formula VO Creation

This section presents a primary abstract formal model for VO creation domain, representing formula with the RAISE method and its language. This formula is innovation for the implementation of a provable process and improving VO creation. As mentioned before, the aim of DE is to develop an abstract formal model of domain with domain axioms that are represented in specification languages such as RSL.

In general, a VO creation (VOc) can be defined with use of mapping organizations (Org) into their information (VOInfo) including the VO member selection (Sel), VO member registration (Reg), forming partnership (Par), and creating database (DB). Meanwhile, the VO member selection (Sel) is concluded from the VO member competency (Cmp) and the VO member trust (Tru) according to the past performance measurements of members saved in the database (DBInfo) for selecting in new VOs. This definition has been written in RSL language in the following (Bold words are reserved words in RSL).

$$\text{VOc} = \text{Org} \xrightarrow{m} \text{VOInfo}$$

$$\text{VOc} = \text{Org} \xrightarrow{m} (\text{Sel} \times \text{Reg} \times \text{Par} \times \text{DBInfo} \xrightarrow{m} \text{Performance Measurement})$$

$$\text{DB: Cmp} \times \text{Tru} \xrightarrow{m} \text{DBInfo}$$

An axiom of this domain is as"The member selected for a new VO must has eligible capabilities and if the eligible member cannot be found within the VBE, it must be selected from outside". Is-Sel defines this

axiom. Output of this function shows that member has been selected or has not been selected.

Is-Sel: Org $\longrightarrow$ **bool**.
o,o1: Org, t: Tru, c:Cmp
Is-Sel (o) $\equiv$
$\forall$ o: Org, $\exists$ t,c $\in$ **dom** DB(m(DBInfo))
if $\exists$ db: DB **then** Findoutside(o).

Findoutside: Org $\longrightarrow$ DB
Findoutside(o1) $\equiv$ **if** $\exists$ t1,c1 $\in$ **dom** DB(m(DBInfo1)) **wr**
am: c1 $\times$ t1 $\xrightarrow{m}$ DBInfo $\wedge$ o1 $\xrightarrow{m}$ VOInfo
post o1 $\notin$ **dom** VOc* $\wedge$ am= am* $\bigcup$ t1,c1 $\longrightarrow$ partnership() **end**

In above function, it is specified that if there is not eligible organization (with respect to competency and trust) within the database, a new eligible organization must be selected from outside. New organization as new VO member also joins other the VO members. *Post* means post condition after joining.

Hence, new VO member should not already be within the VBE. *am** means old"am". After adding new member, the set of new am-set includes this member. Partnership function forms the best partnership in terms of the competency (c1) and the trust (t1), creates the necessary databases, and registers new member.

To describe this formal model, work flow is needed. Abstract state of VO creation domain includes identifying business opportunities, evaluating the partner competencies, selecting partners from within or outside the VBE (network), forming the best partnership in terms of the competencies, creating the necessary databases, and registering new members. In next paper, we will present formal specifications to define above work flow and formal verification of axioms defined for this domain with RSL and its proof rules.

5. Conclusion and Further Work

The aim of Domain Engineering is to specify and develop an abstract domain model and domain analysis using formal specification languages. It is performed before requirement engineering and software design in the development of safety critical systems. Formal methods are the basis of domain engineering to develop reliably software for such systems because they are based on mathematics and logic. Therefore, they are provable. RAISE is a comprehensive development formal method which we used in this paper to develop domain model with presenting formula. In this paper, we presented a primary formal model (formula) of VO creation using RAISE method and its language (RSL). This formal model can be a provable fundamental process for the design and the implementation of reusable and reliable members for VO creation. Nowadays, research on serving formal methods in distributed systems is growing. Formal Methods for Open Object-based Distributed Systems (FMOODS) is one of good ideas that serves advantages of formal methods to develop provable and reliable software and distributed processing such as selecting eligible new member in distributed environments. Future researches include expanding these formulas and proceeding formal verification of axioms defined for this domain with RSL and its proof rules.

6. References

[1] H. Afsarmanesh and L. M. Camarinha-Matos. A framework for management of virtual organization breeding environment. In *PROVE'05* , page September, 2005.

[2] D. Bjorner. Where do Software Architectures come from ? Systematic Development from Domains and Requirements. A Re-assessment of Software Engineering? In *SouthAfrican JournalofComputerScience* , volume 22, pages 3–13, 1999.

[3] D. Bjorner. Domain Engineering: A Software Engineering Discipline in Need of Research. In *Lecture NotesinComputerScience,Springer-Verlag* , volume 1963, pages 1–17, November 2000.

[4] D. Bjorner. Formal Software Techniques in Railway Systems. In *In Eckehard Schnieder, editor, 9th IFAC SymposiumonControlinTrans portationSystems,Technical University, Braunschweig, Ge rmany. VDI/VDEGesellschaft Mess- und Automatisieringstec hnik, VDIGesellschaft fr Fahrzeug-und Verkehrstechnik. Invited plenum lecture.* , pages 1–12, June 2000.

[5] D. Bjorner. A Cloverleaf of Software Engineering. In *Third IEEE International Conference on Software Engineering and Formal Methods (SEFM 2005), Koblenz, Germany*, pages 75–85, September 2005.

[6] D. Bjorner. Software Engineering 2: Specification of Systems and Languages. In *Springer*, October 2006.

[7] D. Bjorner. Software Engineering 3: Domains, Requirements, and Software Design. In *Springer*, October 2006.

[8] R. Camacho and et al. An integrative approach for VO planning and launching. In *PROVE'05* , page September, 2005.

[9] L. M. Camarinha-Matos, H. Afsarmanesh, and M. Ollus. Virtual Organizations: systems and practices. In *Springer Science*, 2005.

[10] I. Classen. Revised ACT ONE: categorical constructions for an algebraic specification language. In *Springer-Verlag NewYork,Inc.* , pages 124– 141, 1989.

[11] D. Craigen, S., Gerhart, and T.J. Ralston. An International Survey of Industrial Applications of Formal Methods: Volume 1 Purpose, Approach, Analysis, and Conclusions. In *NationalTechnicalInformationService, 5285PortRoyalRoad,Springfield,VA22161,USA* , volume 1 and 2, March 1993.

[12] J. Davies and J. Woodcock. Using Z: SPecification, Refinement, and Proof. In *PrenticeHall* , 1996.

[13] T. Dimitrakos and et al. Towards a Trust and Contract Management Framework for Dynamic Virtual Organisations. In *Proceedingofthe eChallenges2004,Vienna,Austria* , October 2004.

[14] F. Graser et al. Towards performance measurement in virtual organizations. In *PROVE'05* , September 2005.

[15] S. Gerhart, D. Craigen, and T. Ralston. Observations on industrial practice using formal methods. In *Proceedings of the 15th international conference on Software Engineering, Baltimore, Maryland, United States* , volume 36, pages 24–33, May 1993.

[16] The RAISE Method Group. The RAISE Development Method. In *PrenticeHall* , 1995.

[17] A. Hall. Using Formal Methods to Develop an ATC Information System. In *IEEE Software* , volume 13, pages 66–76, March 1996.

[18] J. Haller. A stochastic approach for trust management. In *22nd International Conference on DataEngineeringWorkshops(ICDEW'06),Atlanta, GA,USA* , April 2006.

[19] T. Hoare. Communicating Sequential Processes. In *1stedition,PrenticeHall* , June 1985.

[20] C. Michael Holloway. Epistemology, Software Engineering, and Formal Methods. In *PresentedatThe RoleofComputersinLaRC* , June 1994.

[21] Jaroslaw Magiera and Adam Pawlak. Security frameworks for Virtual Organizations. In *Virtual Organizations,Springer* , pages 133–148, 2005.

[22] J. Storbank Pedersen. RAISE Frequently Asked Questions.In *http://spdweb.terma.com/Projects/RAISE/faq.html*, June 2004.

[23] M. Penicka and D. Bjorner. From railway resource planning to train operation. In *Building the Information Society, IFIP 18th World Computer Congress, Topical Sessions, Toulouse, France*, volume August, pages 629–636, July 2004.

[24] W. J. Toetenel S. Prehn. VDM '91 - Formal Software Development. In *4th International SymposiumofVDMEurope,Noordwijkerhout,The Netherlands, Proceedings, Volume 1:* Conference Contributions. Lecture Notes in Computer Science*551 Springer*, pages 15–22, October 1991.