

# Knowledge Based Automated Software Engineering



# Knowledge Based Automated Software Engineering

Edited by

Ivan Stanev and Katalina Grigorova

**CAMBRIDGE  
SCHOLARS**

---

P U B L I S H I N G

Knowledge Based Automated Software Engineering,  
Edited by Ivan Stanev and Katalina Grigorova

This book first published 2012

Cambridge Scholars Publishing

12 Back Chapman Street, Newcastle upon Tyne, NE6 2XX, UK

British Library Cataloguing in Publication Data  
A catalogue record for this book is available from the British Library

Copyright © 2012 by Ivan Stanev and Katalina Grigorova and contributors

All rights for this book reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the copyright owner.

ISBN (10): 1-4438-3771-7, ISBN (13): 978-1-4438-3771-2

# TABLE OF CONTENTS

Preface .....	vii
Chapter One.....	1
KBASE Unified Process	
Ivan Stanev and Katalina Grigorova	
Chapter Two .....	21
Towards a Unified Approach for Modelling and Management of Workflow Processes	
Dimitar Blagoev and George Totkov	
Chapter Three .....	35
Modelling and Carrying on E-learning Processes with a Workflow Management Engine	
Hristo Indzhov, Rositza Doneva, Dimitar Blagoev and George Totkov	
Chapter Four .....	53
Intelligent Business Process Repository	
Katalina Grigorova and Ivaylo Kamenarov	
Chapter Five .....	67
Method for Automated Programming of Robots	
Ivan Stanev	
Chapter Six .....	87
Implementing Search Strategies in Winspider I: Introduction to Control Network Programming and Search	
Kostadin Kratchanov, Emilia Golemanova, Tzanko Golemanov and Yasemin Gökçen	

Chapter Seven.....	115
Implementing Search Strategies in Winspider II: Declarative, Procedural, and Hybrid Approaches	
Kostadin Kratchanov, Emilia Golemanova, Tzanko Golemanov and Yasemin Gökçen	
Chapter Eight.....	137
A Decision Support Method in Informatics Teaching for Beginners	
Neli Maneva and Plamenka Hristova	
Chapter Nine.....	153
Genetic Algorithms as a Toolkit for Automated Technological Design in CAD/CAM Environment	
Ivo Atanasov and Desislava Atanasova	

# PREFACE

This book presents the authors' views on the Knowledge Based Automated Software Engineering (KBASE). It involves the subject domain scopes, the implemented research methods, tools and applications.

## The authors

The authors are members of teams from the following universities:

- Plovdiv University, Bulgaria, with Prof. George Totkov leading;
- Sofia University, Bulgaria, with Prof. Neli Maneva leading;
- Yaşar University, Turkey, with Prof. Kostadin Kratchanov leading;
- University of Ruse, Bulgaria, with Ivan Stanev and Prof. Katalina Grigorova leading.

Distinctive for the authors is their long-term research activities in the field of Software Engineering, Artificial Intelligence, Data Bases, and Computer Science. The teams worked out KBASE products within the domains of Software Engineering (SE), Manufacturing Engineering (ME), Education (E), and Computer Science (CS).

## Coverage

The KBASE products presented in the book are included in chapters as listed below:

- **Chapter 1.** *The SE method* KBASE Unified Process, *the SE applications* Adaptive Document Display, and Information Objects Manager, as well as *the ME application* Intelligent Product Manuals.
- **Chapter 2.** *The CS method* Unified Approach for Workflow Execution and *the SE tool* Workflow Management Engine (WME).
- **Chapter 3.** *The E application* eLearning Workflow Engine for Moodle.

- **Chapter 4.** *The SE method* Business Process Generation and *the SE tool* Business process repository.
- **Chapter 5.** *The SE method* Automated Programming of Robots, and *the SE tool* Module for Automated Programming of Robots (MAP).
- **Chapter 6.** *The CS method* Control Network Programming.
- **Chapter 7.** *The CS tool* Control Network Programming (WinSpider).
- **Chapter 8.** *The E method* Decision Support in Informatics Teaching for Beginners.
- **Chapter 9.** *The ME* Automated Technological Design Tool (ATDT).

### Audience

The methods, tools, and applications described in the book are addressed to the needs of scientists, practitioners and students in Software Engineering, Computer Science, Knowledge Representation, Artificial Intelligence, Manufacturing Engineering and Education.

### Acknowledgments

We are grateful to the European Commission (programs INCO Copernicus, and ICT), the Bulgarian Government, and Sofia Technical University for the grants provision.

Our thanks go to Maria Koleva for her valuable assistance and tolerance; to Velichka Stateva for her important role in checking the manuscript; to Prof. Chavdar Vesirov, Prof. Krastimir Popov, Valentina Vuleva, Rumen Kozhuharov, Savka Kalinova, Julia Zlateva, Daniela Tsaneva, Rosen Kozlev, Stanislava Ninova-Kozleva, Yavor Trapkov, Adriana Borodjieva, Svetlin Stoyanov, Svetlin Petrov, and Irena Valova from the University of Ruse for their contributions in the implementation of some KBASE tools and applications.

Ivan Stanev  
Katalina Grigorova  
April 2012

# CHAPTER ONE

## KBASE UNIFIED PROCESS

IVAN STANEV AND KATALINA GRIGOROVA

**Abstract.** The scope of Knowledge Based Automated Software Engineering (KBASE) domain area is defined. Rational Unified Process (RUP) is extended to KBASE Unified Process (KBASE-UP) with a set of new KBASE objects. Three business processes (for Ontology Generation, Information System Generation, and Domain Area Problem Solving) are added to RUP. KBASE Technological Framework, based on RUP Service Oriented Architecture (SOA) Enterprise Framework is composed as SOA-based software. Some important techniques (non-formal task specification, formal software verification and modification, automated program generation, and runtime application self-monitoring) are incorporated in the KBASE Technological Framework. Five applications realised under this Framework are described for illustration of some KBASE advantages.

**Keywords:** Automated Programming, Knowledge Based System, Software Engineering, Technological Framework, Unified Process, Ontology, Non-formal Task Specification, Formal Software Verification, Runtime Self-monitoring.

### Introduction

The process of developing big information systems (IS) is less effective and more resource consuming than the software developers expect. Most widely disseminated software engineering methods and tools applied through the life cycle of this process are characterised with a low level of process automation, insufficient component reusability, and bad final product flexibility.

The efficiency of the software development process can be improved using hi-tech Information Technology (IT) instruments such as: (1) non-

formal (insufficient, and incomplete) business model specifications, (2) automated verification and modification of non-formal specifications based on predefined standardised knowledge bases (KB) both for domain and IT areas, (3) direct automated generation of final software product from verified business model, and (4) incorporation of a set of components for real time monitoring and tuning into the generated software.

The designed new methods and tools, which aim to achieve industrial importance, must support a combination of widespread IT industrial standards (such as UML, BPMN, JEE, .Net, XML, WSDL, RUP, etc.), and software architectures (Multi-Layer, Service Oriented, GRID, etc.), as well as the above mentioned instruments.

This paper describes: (1) the extension of RUP with new KBASE objects and processes, (2) KBASE Technological Framework (TFR), (3) four applications illustrating some important KBASE techniques.

## KBASE Unified Process

Four features, standardisation, capsulation, flexibility, and automation, are important for the realisation of the KBASE process.

The Rational Unified Process (Kruchten 2001) is one of the worldwide disseminated industrial methods for description and management of IT projects through their full lifecycle. It is the only **industrial standard** in its area.

The SOA version of RUP (IBM 2009) based on business processes (BP) and services (tasks, as given below) enriches the well-defined RUP standardisation with three features important for the KBASE: **capsulation** of its components (realised as reusable tasks and processes), **flexibility** of the computation process (based on the built-in mechanism for rapid reengineering of the business processes), and **automation** of the IT product development (achieved by the SOA process server and integration servers).



The above mentioned features make RUP for SOA a good basis for the establishment of KBASE Unified Process.

The KBASE-UP concept (including objects, components, and processes) is defined below and presented in Fig. 1-1.


## KBASE objects

Three important categories, tasks, products, and roles (Heijde 2006), are the ground for the RUP definition. These three categories are used for extension of RUP to KBASE-UP.


The RUP objects are enriched with the below mentioned new KBASE objects:

- **Products** (presented in Fig. 1-1. as )
  - **Non-formal BP Specification** is an insufficient, incomplete, or fuzzy (Zadeh 1996) business process specification;
  - **Formal BP Specification** is a determined BP specification enriched with new (extracted from KBASE Ontology) data, facts, relationships, and algorithms during the non-formal BP specification processing;
  - **Ontology** is a knowledge within a domain area (DA), formally presented as a set of primitives and the relationships between them;
  - **Non-formal Ontology Specification** is an insufficient, incomplete, or fuzzy ontology specification;
  - **Problem Specification** is a determined specification of the problem to be solved described by KBASE users;
  - **Problem Result** is an output data set received by KBASE users during and at the end of the problem solving.
- **Tasks** (presented in Fig. 1-1. as )
  - **BP Editor** is a software component used by KBASE IT and DA experts for specification of the business processes and their tasks;
  - **BP Specification Processor** is a software component for non-formal BP specification conversion into a formal one, based on the content available in the ontology KB;
  - **IS Generator** is a software component which generates a new software product in Compiler regime, using the BPs' and BP tasks' formal specifications, and the Ontology KB;
  - **Ontology Editor** is a software component used by KBASE IT and DA experts for specification of the ontology and its primitives;
  - **Ontology Specification Processor** is a software component for conversion of non-formal to formal ontology;
  - **Ontology Generator** is a software component which generates a new software product in Compiler regime, using the ontology formal specifications, and the Ontology KB content;
  - **Problem Specification Editor** is a software component used by KBASE IT and DA users for specification of the problem to be solved;
  - **Information System Interpreter** is a software component which solves the relevant problem, using the preliminary generated IS

suitable for it, as well as the problem specification defined by the KBASE users.

- **Roles** (presented in Fig. 1-1. as )
  - **Domain Area Knowledge Engineer** is a DA specialist, or business analyst, who designs new data objects, ontology primitives, ontology structures, Graphical User Interfaces (GUI), messages, dialogue sequences, matrices <roles x rights> (MRR), etc.;
  - **IT Knowledge Engineer** is an IT specialist (designer, implementer, tester, etc.) who implements new data objects, tasks, ontology primitives (OP), GUIs, messages, or dialogue sequences;
  - **Domain Area User** is a domain area specialist, who introduces the specification of the problem to be solved into the generated IS as well as the input data, and analyses the calculated problem results;
  - **IT User** is an IT specialist (business analyst or system administrator) who manages the security and history of the IS computation process, analyses the data generated during the IS performance and manages the process of fine IS tuning.

### KBASE components

The KBASE objects are organised in four KBASE components (presented in Fig. 1-1. as ):

- **Knowledge processor** is in charge of BP conversion and BP tasks non-formal specifications into formal BP specifications, as well as the ontology and ontology primitives' non-formal specification into formal ontology specifications. On the occasion that the conversion process is performed incompletely and contradictorily, the knowledge processor may address the KBASE users for help, providing relevant description and explanation of the results achieved to the moment.
- **Product Generator** is in charge of generating new IS (IS Generator) based on the formal BP and BP tasks specifications. It is also in charge of generating new DA ontology (Ontology Generator) based on the formal ontology specifications and ontology primitive repository.

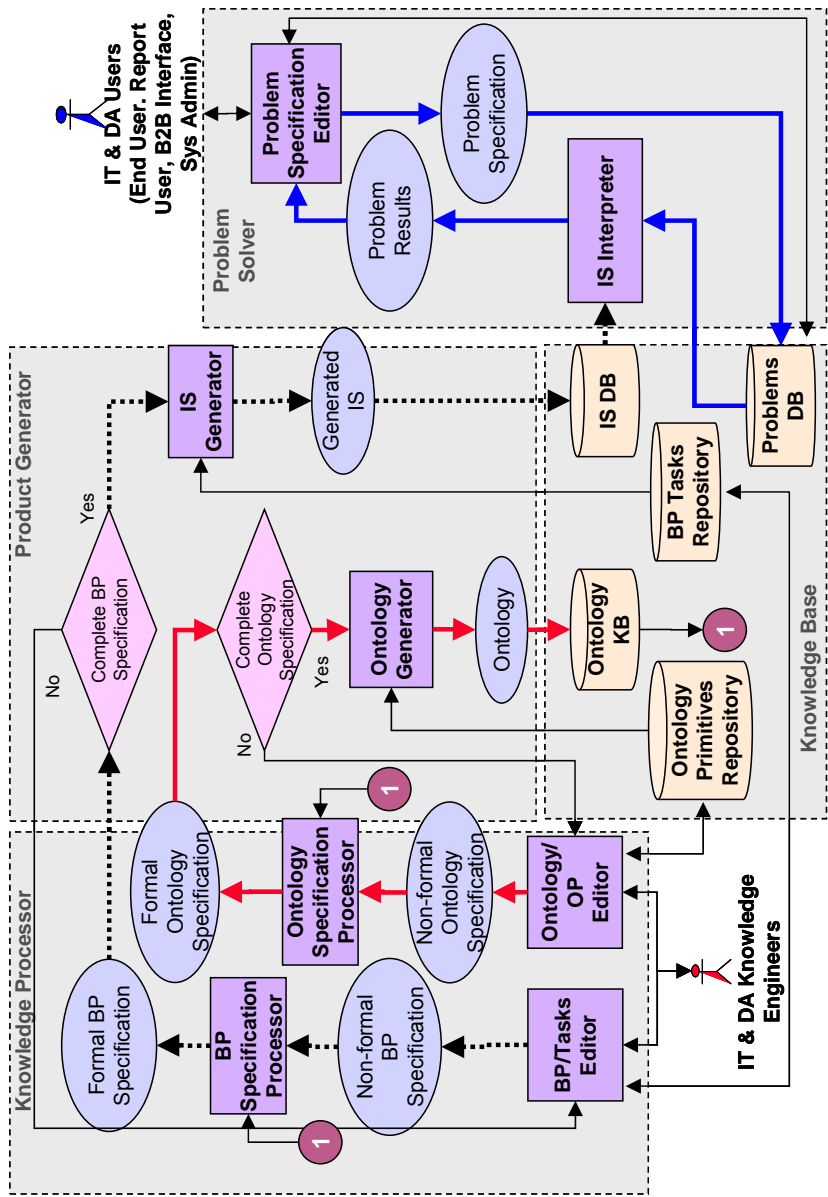



Fig. 1-1. KBASE Concept

- **Problem Solver** takes charge of the management of the problem solving process, as well as the provision of the interface between KBASE and users.
- **Knowledge Base** stores the knowledge needed for IS generation, DA ontology generation, problem solving, as well as problems' input data and output results. This knowledge is organised within the below mentioned components (presented in Fig. 1-1. as ):
  - **BP Task Repository** is a storage location for reusable IT and DA tasks used for business process establishment;
  - **Ontology Primitives Repository** is a storage location for reusable IT and DA primitives used for DA ontology establishment;
  - **Ontology Knowledge Base** is a storage location for different full or partial versions of DA ontology;
  - **Information Systems DB** stores different versions of ISs as generated by KBASE;
  - **Problems DB** is a storage location for different input data sets and output results required for problems solving.

### KBASE processes

The end-to-end functioning of the KBASE can be described with the business processes (bp01 – bp03) as listed below. The notations s0i are steps within the processes.

- **bp01: Ontology Generation** (s01) DA expert informally describes his/her DA knowledge; (s02) the created specification is analysed by the Ontology Specification Processor and further transformed into formal specification; (s03) the formal specification is checked for consistency and completeness. If it is not complete, it is sent for additional knowledge entry by the DA expert (return to s01, and repeat process). If it is complete (s04) it is sent to the Ontology Generator that generates the ontology of the information system. (s05) The generated ontology is stored in the Ontology KB.
- **bp02: Business Process Generation** (s01) The IT expert informally describes business processes that have to be realised in the generated IS; (s02) the created specification is processed by the BP Specification Processor and is transformed into a formal specification; (s03) the formal specification is checked for consistency and completeness; if it is not complete it is sent for additional specification of BP by the IT expert (return to s01, and repeat

process); if it is complete, (s04) it is sent to the IS Generator for code generation; (s05) the generated system is stored in the IS DB.

- **bp03: Problem solving** (s01) The DA user describes a problem that he/she needs to solve; (s02) the problem description is stored in the Problems DB; (s03) the specified problem together with the code of IS which solves this problem (generated by bp02, and saved in IS DB) are sent to the IS Interpreter in the Problem Solver; (s04) Problem results are calculated by the IS Interpreter and sent to the Problem Specification Editor for analysis by the DA user.

## KBASE Technological Framework

The KBASE-UP TFr (including technological objects, technological components and technological processes) is presented in Fig. 1-2. and is defined below.

### KBASE TFr objects

KBASE TFr objects are classified according to the **object hierarchy** in two groups: **Terminal Objects (TO)**, including all primary (indivisible) objects; and **Non-terminal Objects (NO)**, which are constructed by TOs and other NOs through UML class associations.

KBASE objects are classified according to the **object type** also in two groups: **Data Objects (DO)**, including pure data structures of the solved task, DA ontology, and system objects of the generated BPs and ISs; **Control Objects (CO)**, including data and methods of the KBASE Tasks, Ontology Primitives, Business Processes, and Ontology.

The RUP TFr objects are enhanced with new KBASE TFr objects as mentioned below:

- **Message** is a (non-)terminal DO describing the KBASE input and output messages (including KBASE correspondence with B2B remote clients, and all KBASE users, as well as the correspondence between KBASE components).
- **Screen** is a (non-)terminal DO describing the set of graphical objects (boxes, pop-up menus, radio buttons, images, etc.) which are presented at the same time on the display screen.
- **Form** is a (non-)terminal DO describing a KBASE document (screen or tree of screens) for input and output of operational information.

- **Report** is a (non-)terminal DO describing a KBASE document (screen or tree of screens) for input and output of analytical information.
- **Role Record** is a (non-)terminal DO describing a KBASE role, including the role script, permitted and prohibited role operations, DOs permitted and prohibited for that role, etc.
- **History Record** is a (non-)terminal DO presenting a KBASE operational record in the system logs, which describes the behaviour (static and dynamic) of all data and control objects.
- **Role – Resources Matrix (RRM)** is a non-terminal DO describing the relationships between KBASE roles and KBASE resources.
- **Object Descriptor** is an unique (non-)terminal DO, which describes the object structure and its store location for DO and its instances; the object name, set of the object performance regime parameters, and set of object composition parameters for CO.
- **Task** is a terminal CO which is the technological projection of a KBASE task, realised as xml, java, C#, etc. code or as a runtime KBASE unit.
- **Business Process** is a (non-)terminal CO which is the technological projection of a KBASE business process, realised as Business Process Execution Language (BPEL) or similar standard.
- **Ontology Primitive** is a terminal CO which is the technological projection of a KBASE ontology primitive, realised as a set of parameters, and methods describing the object behaviour.
- **Ontology** is a (non-)terminal CO which is the technological projection of KBASE ontology, realised as a set of ontology primitives describing the object behaviour.
- **Information System** is a (non-)terminal CO which is the technological projection of a KBASE IS, realised as an executable code generated by the KBASE.

The KBASE Control TFr objects are more detailed descriptions of the above mentioned KBASE objects: task, business process, ontology, ontology primitive, etc.

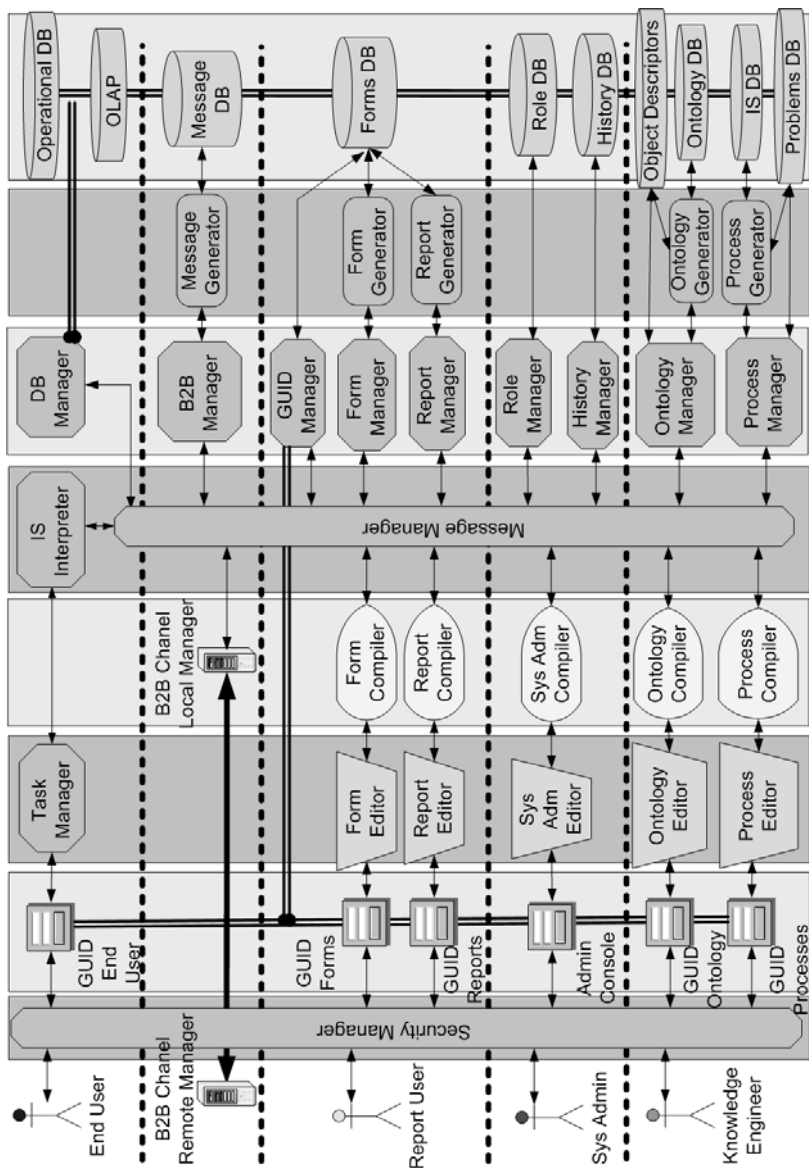








Fig. 1-2. KBASE Technological Framework

## KBASE TFr components

- **GUI Display** (presented in Fig. 1-2. as ) is a software component which conducts the dialogue between the KBASE TFr and the players of KBASE roles. It is typically realised as html, xml, or JSF/JSP product. Both distributed and centralized technologies for GUID are acceptable, depending on the realized task, and the used communication channels quality. The key requirements for GUID realization under KBASE TFr are the automation of its design, development, and testing process, as well as its rapid reengineering.
- **Editor** (presented in Fig. 1-2. as ) is a software component which ensures the design or modification of GUIs (Form Editor, Report Editor), the parameterisation of KBASE components (Sys Admin Editor), as well as the generation or modification of KBASE tasks, ontology primitives, processes, and ontology (Ontology Editor, Process Editor). These components are realised on the basis of text, GUI, or Natural Language (NL) specification techniques. The specification languages are based on the wide spread international standards (such as BPMN, UML, EPC, etc.). The simple specification techniques and availability of rich libraries with DA and KBASE predefined reusable objects are key requirements for the realization of Editors.
- **Compiler** (presented in Fig. 1-2. as ) is a software component which ensures the transformation of non-formal to formal user specification. The formal specification is usually based on international standards such as XML, WSDL, JEE, .Net, etc. The key requirement for the Compiler functionality is the capability of identifying the insufficient, incomplete, or fuzzy parts of user specifications and for producing a fully defined formal specification.
- **Manager** (presented in Fig. 1-2. as ) is a software component responsible for the organisation, management and fault tolerant performance of every particular KBASE TFr process. Its role is to ensure the resources required for the process performance, to organise communication with other managers, to control the prioritisation of the required services performance, as well as to identify and, if possible, to correct the own runtime errors. These managers are realised usually as Intelligent Agents organised in a Multi Agent System. Well-defined self-training and self-organisation capabilities are the main requirements for these managers.

- **Generator** (presented in Fig. 1-2. as ) is a software component responsible for the generation or modification of other software components (forms, reports, tasks, processes, ontology primitives, and ontology) through formal specifications. Radical reduction of development and testing activities performed by the KBASE experts after component generation is the main goal of its functionality.
- **DB** (presented in Fig. 1-2. as ) is a software component responsible for storing of KBASE knowledge and data. It is typically realised as relational or object DB. Both distributed and centralized technologies for the DB are acceptable depending on the realized task and the used communication channels quality.

### KBASE TFr processes

This section presents the process concepts and main process scenarios. Alternative KBASE scenarios can be produced by KBASE experts and users depending on the requirements and the problems to be solved. Five key KBASE technological processes are in focus as follows:

- **tp01 SOA Interpretation Process**

The KBASE SOA interpretation process is realised as a SOA RUP TFr extension. The SOA Interpretation process components (presented in Fig. 1-2.) are listed in format <KBASE Component> (<SOA RUP equivalent>): Security Manager (Part from Portal), Task Manager (Application Server), Process Manager (Process Server), Message Manager (Enterprise Service Buss - ESB), DB Manager (DB Server), IS DB (Process and Task repositories), as well as IS Interpreter (Application Server).

The process steps follow the SOA standard interpretation and are not described in detail.

- **tp02 Knowledge Engineering Process**

This process describes the generation of new ontology and IS for solving of a particular class of problems.

The main Knowledge Engineering process scenario includes: (s01) DA and IT Knowledge Engineers select from the Ontology DB the available DOs, tasks, ontology primitives required for the problem to be solved; (s02) DA and IT Knowledge Engineers specify the new DOs, forms, reports, ontology primitives, tasks, messages, permitted and prohibited message sequences, RRM, and history records, which are not found in the Ontology DB; (s03) Ontology and Process Compilers analyse newly specified objects and, if being non-formally specified, they require new

specifications; (s04) Ontology and Process Generators generate new objects and incorporate them in new versions of Ontology and IS; (s05) the Ontology and Process managers control the process and record the generated objects, ontology and IS in the Ontology and IS DBs.

- **tp03 Domain Customisation Process**

This process describes new instance customisation of the IS generated in tp02.

The main Domain Customisation process scenario includes the following steps: (s01) Knowledge Engineers modify the required DOs using the Ontology Editor; (s02) Knowledge Engineers and End Users modify messages, message sequences and forms of the customised IS; (s03) Knowledge Engineers and Report Users modify reports of the customised IS; the Process Manager generates the modified IS and records it in the IS DB.

- **tp04 System Customisation Process**

This process describes the system tuning of the instance of a particular IS, as customised in tp03.

The main System Customisation process scenario includes the following steps: (s01) Sys Admin defines the roles and records them in the Role DB; (s02) Sys Admin activates the required history processes, and records them in the History DB; (s03) Sys Admin defines RRM and records it in the Role DB; (s04) Sys Admin tunes the performance control parameters, and records them in the IS DB.

- **tp05 Runtime Operation Process**

This process describes the problem solving which uses the instance of a particular IS as customised in tp04.

The main Runtime Operation process scenario includes the following steps: (s01) End User specifies a set of input data for the problem to be solved; (s02) IS interpreter generates the output data set based on the selected IS and the input data set; (s03) End User analyses the output data; (s04) If the output data is acceptable it is recorded in the Problems DB, if not, the process is returned to step s01 for modification of the input data set.

## **KBASE Applications**

The advantages of KBASE TFr are illustrated in brief with the description of six applications, realized on the basis of the existing KBASE techniques and components. The KBASE features are viewed in four groups in order to obtain precise description and comparison of the KBASE applications. These groups involve the KBASE specification

techniques, IT technologies, realized KBASE components, and realized KBASE technological processes.

Table 1-1. presents the KBASE applications (columns), characterized by the KBASE features (rows).

**Table 1-1. KBASE applications described by KBASE features**

KBASE Feature Type	KBASE Feature Name	KBASE Applications				
		MAP	BP Gen	IPM	BIT. ADD	IO Man
Specification technique	Formal language	yes	yes		yes	
	Graphical user interface		yes	yes		yes
	Natural language	yes				
IT hi-tech used	Non-formal specification	yes		yes		
	Code generation	yes	yes	yes		yes
	Self-verification	yes	yes			
	Self-monitoring				yes	
	Self-tuning			yes		
	Knowledge processor	yes		yes		
KBASE Component	Product generator	yes	yes	yes	yes	yes
	Problem solver	yes	yes	yes		
	Knowledge base	yes	yes	yes	yes	yes
	SOA interpretation process					yes
Technological process	Knowledge engineering	yes	yes	yes	yes	
	Domain customisation process	yes	yes	yes		yes
	System customisation process	yes	yes		yes	yes
	Runtime operation process	yes		yes	yes	

A detailed description of these applications is given below:

- **Module for Automated Programming of Robots (MAP<sup>1</sup>)**
  - **MAP Description**

MAP (see Chapter 5 for more details) generates machine programs for robot control through Natural Language (NL) Specifications. The specification text can be a fuzzy, incomplete, or insufficient description of robot business processes (including descriptions of fuzzy information objects and fuzzy robot commands). A formal language Net (Stanev 2001) is used for ontology specification (such as knowledge for NL interpretation, description of fuzzy command interpreter, work space maps, etc.). A linguistic processor for Bulgarian language (Stanev 2002) is used for generation of a fuzzy robot program from an NL text. The Specification Validator checks the completeness and consistency of the

---

<sup>1</sup> The work presented here has been partially funded by the project Bg.TU-Sofia.SRC.741178. System for Robocar Control through Bulgarian Language Specification.

fuzzy robot program. The Product Generator creates the determined version of the fuzzy robot program. The Problem Solver supported by the Robot Knowledge Base is responsible for the interpretation of the determined robot program. All KBASE technological processes are supported by MAP, excluding SOA interpretation process because of the programming language Net specifics.

➤ **MAP Advantages**

The accent is placed on two achieved goals: (1) the time for realisation of a specific Robot control assembly program (approximate volume of 300KB for Motorola 6800 processor) is reduced from 120 to 1 man/day (in the case of preliminary created DA ontology); (2) all actions for updates' preparation needed for MAP working in a new DA require from 2 to 10 man/days.

- **Business Process Generator (BP-Gen)**

➤ **BP-Gen Description**

BP-Gen provides functionality that allows the production of new BPs using preliminarily defined BPs stored in the BP repository. The new process generation is performed using the rules incorporated in the BP-Gen. Changing the rules or the identification of the subprocesses enables the generation of several instances of the new process.

The representation of a BP repository may be viewed in a sense as an on-line database (Grigorova 2012). The direct access to it allows the system analysts to have fast and varied ways of business process analysis.

➤ **BP-Gen Advantages**

The main BP-Gen advantage relates to one insufficiently discussed aspect of the utilizing business process instances and patterns stored in business process repositories. If a process designer has a tool for a fast and convenient way of process construction, he can analyse the variants of a business process.

- **Intelligent Product Manual (IPM<sup>2</sup>)**

➤ **IPM Description**

The IPM (Stanev 2000) is a software component which creates Product Manuals (PM) for industrial products. The created PM includes relevant software modules and data for user guide, expert system, training system, and document display of the product. IPM generates PM software modules and data from product information objects, DA ontology, and DA task library. The product information objects are established using tools such as

---

<sup>2</sup> The work presented herein has been partially funded by project EU.INCO Copernicus 96/0231. Intelligent Product Manuals.

ProEngineer, Macromedia, SAP, Oracle, and Kappa during the design process. The language Net and a GUI display are used for DA Ontology formal specification. An IPM software component is realised for verification of the generated PM code and data consistency and completeness. The IPM software architecture is designed including the components Automated Document Generator (ADG), Intelligent Training System (ITS), Hypermedia Adaptive Diagnostic Expert System (HADES), and Adaptive Document Display (ADD). The IPM DB includes data structures for products, documents, actors, and training process. The IPM DA ontology includes knowledge for product construction, product diagnostics, document standards, document templates, document generation, document visualisation, actor description, and actor training. An ADG process is designed for product manuals generation on the basis of the product structure, product data, document standards, and document templates. The process of ITS generation is constructed based on the product, document and actor data. The process of HADES generation and adaptation to the rapidly changed product characteristics, dynamically exploited environment and actor profiles is described. The process of ADD adaptation to actors depending on the document media, actor profiles and actor experience is presented. A technological framework for automated Intelligent Product Manuals generation is created, including development, deployment and delivery environments. The components: Knowledge Processor, Product Generator, Problem Solver and Knowledge Base are included in the TFr for the realisation of knowledge engineering, domain customisation, and runtime operation KBASE processes.

#### ➤ **IPM Advantages**

Three main IPM quality improvements are achieved based on the use of KBASE techniques: (1) PM presentation is adaptive to different working environments and different end user qualification; (2) PM content is adaptive to different presentation styles, document standards, and media; (3) PM content is automatically updated in real time conditions.

The impact of the new IPM technologies can be illustrated by some figures obtained during the prototyping of two Product Manual versions for the same Fork Lift Truck. Approximately, two man/years are required for the realisation of a non-automated and non-adaptive PM version. The time spent for the realisation of a more powerful, automated and adaptive PM version is reduced approximately to 40%.

- **Built-in-Test Adaptive Document Display (BIT.ADD<sup>3</sup>)**

- **BIT.ADD Description**

BIT.ADD is a software module which is the second version of the above mentioned IPM Adaptive Document Display. This module includes a set of Built-In-Test components (Component+ 2003) for improvement of BIT.ADD testability and reusability. Two new functionalities are added in BIT.ADD. The first functionality realises the automation of Contract tests process, during the integration of BIT.ADD as Commercial-of-the-shelf (COTS) component with hosting system. Contract tests are performed during the hosting system development process and include the verification of automated communication channel functionality. These tests involve syntax and (where possible) semantics validation of input and output BIT.ADD messages structure; generate for testing purposes right and wrong dialogue sequences between host system and BIT.ADD, as well as validate dialogue scenarios. The second functionality realises the automation of Quality of Service tests process during BIT.ADD real-time operation. These tests include monitoring of the pair <hosting system, BIT.ADD> behaviour, as well as BIT.ADD self-tuning.

- **BIT.ADD Advantages**

Three main BIT.ADD quality improvements are achieved using KBASE technology: (1) The built-in component Contract Test instruments make it more testable and reusable through minimisation of customisation and testing time for its integration in a new hosting system; (2) The built-in component Quality of Service test instruments make this component self-monitored and self-tuned.

- **Information Objects Manager (IOMan<sup>4</sup>)**

- **IOMan Description**

IOMan is a KBASE tool which generates software components for document management through information objects structure and behaviour formal specification. The IO descriptors are constructed in tables which include all IO fields, rules for formal and logical field control, as well as initial data values (where required). Graphical User

---

<sup>3</sup> The work presented herein has been partially funded by the project EU IST-1999-20162 Development and Applications of New Built-in-Test Software Components in European Industries

<sup>4</sup> The work presented herein has been partially funded by the project BG 2007/019-303.06.04. Development of a National System for Administering the Excise Duty Entirely by the Customs Administration - Development of Excise management system phase 2.1.

Interfaces are used for construction of registers (operational DBs), forms (all the information situated in one screen), documents and reports (constructed as sequences or trees of IOs and/or forms), as well as the MRR from the existing IO descriptors. A State Engine (SE) is described for every document and report in a separate table. The SE description includes document states, document operations, state and operation rules, as well as state and operation pre- and post-conditions. An IO descriptors Compiler creates an XML record for every specified table, sequence and/or tree of forms. Two business processes (BP) are created for the interpretation of XML records. The first BP ensures the interpretation of XMLs describing state engines as long life business processes. The second BP is responsible for the BP task performance management, for the full history record, access rights management, as well as printing and reporting processes.

#### ➤ **IOMan Advantages**

Two main improvements of the development process are achieved using IOMan. The first improvement is the serious reduction of the implementation and testing staff. For the functionality realised in the above mentioned project (8 state engines, each having 10-15 states, 30-40 arrows, and 60-80 IOs) the business analysts staff was increased by 4 persons, and the implementers staff (GUI engineers, code engineers, and testers) was decreased by 9 persons. The second improvement is the time decrease for product development by more than 40%.

## **Conclusions**

The improvements achieved by introducing KBASE technology and tools in the software development process can be summarized (Table 1-2.) in two big groups – quantity and quality features.

**Quantity improvements** are in two aspects: (1) decreasing the time for performance of different software process tasks (modelling, implementing, testing, and customisation); (2) reducing the product development staff.

**Quality improvements** are in three aspects: (1) improvement of product adaptation to DA (adaptation to different users, standards, media, etc.); (2) improvement of product stability (crash prevention and crash problems solving); (3) quality of service improvement including product response time reduction, decreased time for user task solving, increased real-time product and product documentation update and synchronisation, etc.

Another effect of introducing the KBASE technology and tools is the **change of IT development team profile**. The application of the DA ontology requires increasing the number of business analysts in the team. However, the introduction of automation in the implementation process leads to a seriously decreased number of implementers and testers in the team.

**Table 1-2. Improvements achieved in the realised KBASE applications**

KBASE Improvement Type	KBASE Improvement Name	KBASE Applications				
		MAP	BP Gen	IPM	BIT. ADD	IO Man
Quantity improvement	Decrease specification time	yes	yes			
	Decrease DA customisation time	yes	yes	yes		
	Decrease implementation time	yes		yes	yes	yes
	Decrease COTS component integration				yes	
	Decrease testing time	yes	yes	yes	yes	yes
	Decrease COTS component testing time				yes	
	IT staff reduction	yes	yes			yes
Quality improvement	Adaptive to different domain area	yes	yes	yes	yes	yes
	Adaptive to different end user	yes	yes	yes		
	Adaptive presentation to different		yes	yes		
	Adaptive presentation to different media	yes		yes		
	Real time code synchronisation			yes		
	Real time document synchronisation			yes		
	Crashes prevention				yes	yes
	Real time performance improvement		yes		yes	yes

## References

- (BPMN 2008) Object Management Group (OMG), Business Process Modeling Notation. V1.1. Technical report, January 2008.
- (Component+ 2003) EU IST-1999-20162 Development and Applications of New Built-in-Test Software Components in European Industries. Software Architecture.2003
- (Grigorova 2012) Grigorova, K., I. Kamenarov. Intelligent Business Process Repository (this volume).
- (Heijde 2006) Heijde, P. van der. Introduction RUP and RUP/SOA. IBM Corporation. 2006.Pp. 27.
- (IBM 2009) IBM Corporation. RUP for SOMA Roadmaps. Rational Method Composer v7.5.0.1. 2009.
- (Kruchten 2001) Kruchten, P. The Rational Process An Introduction. Addison-Wesley. 2001. Pp. 298.
- (Stanev 2000) Stanev, I., Ch. Vezirov, R. Kozlev, R. Kozhuharov, S. Kalinova. Intelligent Product Manual - Definitions, Structure, and Application in Agricultural Engineering. Proceeding of the XVI

International Conference on "Material Flow, Machines and Devices in Industry" - ICMFMDI'2000, 07 -08 December 2000, Belgrade, Yugoslavia, pp. 1-153 ÷ 1-156.

(Stanev 2001) Stanev, I. Formal Programming Language Net. Part I – Conception of the Language. In proceedings of the CompSysTech'2001. Sofia. Bulgaria. June. 2001. Pp.. I.16-1 – I.16-5.

(Stanev 2002) Stanev, I. A Bulgarian Linguistic Processor Based on the Formal Model Control Networks - General Concepts. In proceedings of the CompSysTech'2002. Sofia. Bulgaria. June. 2002. Pp.. III.7-1 – III.7-5.

(Zadeh 1996) Zadeh, L. Fuzzy Logic = Computing with Words. Ieee Transactions On Fuzzy Systems, Vol. 4, No. 2, May 1996.



## CHAPTER TWO

# TOWARDS A UNIFIED APPROACH FOR MODELLING AND MANAGEMENT OF WORKFLOW PROCESSES

DIMITAR BLAGOEV AND GEORGE TOTKOV

**Abstract.** Business process and workflow management systems are gradually becoming less specialized and more open to different application areas, allowing many types of activities to be described, analysed and even executed and monitored using standardized notations. Currently, many different workflow models, engines and execution frameworks exist, each having its own unique set of features, which makes any simultaneous work on different modelling systems (like BPMN, XPDL, BPEL, Petri nets) tedious and impractical, forcing the users to choose one over the other. This paper presents a workflow modelling and execution system that supports custom runtime controllers, allowing the execution of different models inside one environment. The introduced system also utilizes event-driven messaging communication between concurrently running workflows and centralized workflow storage and execution. This enables seamless sharing of models between members and interoperability between workflows modelled using different notations. The presented system is currently used for researching applicability of workflows in E-Learning and computational linguistics.

**Keywords:** Business Process Management, Workflow Management Engine, E-Learning Systems, Petri Nets, XML Process Definition Language

## Introduction

The business process modelling (BPM) systems continue to draw growing interest in both the business management and software modelling domains (Dufresne 2003). The ability to model, analyse, execute and manage a process with visual constructs is appealing as it provides a more comprehensive understanding of the business logic. These software systems often provide programming interfaces (web services, programming APIs) that enables the developed business applications to extend the functionality of the BPM core. Where they differ is how they describe and execute the models. Since the BPM introduction in (Williams 1967) over the course of more than forty years a number of standards and established modelling languages have emerged. Some of them are:

- Business Process Execution Language (BPEL) (also known as Web Services Business Process Execution Language WS-BPEL) developed by OASIS (Alves 2007);
- XML Process Definition Language (XPDL) – a format standardized by the Workflow Management Coalition (WfMC) (WfMC 2008);
- Java Process Definition Language (JPDM) – a process language build on top of the JBoss jBPM framework (RHM 2007);
- Business Process Modelling Notation (BPMN) - graphical representation for specifying business processes in a business process model, developed by Business Process Management Initiative (BPMI) and currently maintained by the Object Management Group (OMG) (OMG 2008);
- YAWL (Yet Another Workflow Language) – a BPM/Workflow system written in Java and based on Wil van der Aalst's collection of workflow patterns (Aalst 2003, Russell 2007).

One of the main feature that is common to all of these systems is the ability to describe the business processes as workflows. A huge variety of solutions for both modelling and executing workflow models exists. Currently there are more than 40 open source java products (see [http://www.manageability.org/blog/stuff/workflow\\_in\\_java](http://www.manageability.org/blog/stuff/workflow_in_java), <http://java-source.net/open-source/workflow-engines>) alone that offer workflow modelling and/or execution features. Some of them comply with the BPEL standard (Apache ODE, ActiveBPEL, Beexee, PXE), others with the XPDL standard (Open Business Engine, Enhydra Shark, jawflow, Joget Workflow), the IBM's WSFL standard, or the Petri-net based YAWL