World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:9, No:12, 2015

A Common Automated Programming Platform for Knowledge Based Software Engineering

Ivan Stanev, Maria Koleva

***Abstract*—**Common Platform for Automated Programming (CPAP) is defined in details. Two versions of CPAP are described: Cloud based (including set of components for classic programming, and set of components for combined programming); and Knowledge Based Automated Software Engineering (KBASE) based (including set of components for automated programming, and set of components for ontology programming). Four KBASE products (Module for Automated Programming of Robots, Intelligent Product Manual, Intelligent Document Display, and Intelligent Form Generator) are analyzed and CPAP contributions to automated programming are presented.

***Keywords*—**Automated Programming, Cloud Computing, Knowledge Based Software Engineering, Service Oriented Architecture.

I. Introduction

A common technological framework for automated programming has been defined in [8] based on the analysis presented in [5] and [8]. The Common platform for automated programming (CPAP) is built as a combination of Cloud Computing (CC) principles, Service Oriented Architectures (SOA), Knowledge Based Automated Software Engineering (KBASE), and Method for Automated Programming of robots (MAP).

CPAP is composed of four types of components located in different CPAP layers. These four types are as follows:

(1) Components for classic programming (CCLP);
(2) Components for combined programming (CCP);
(3) Components for automated programming (CAP);
(4) Components for ontology programming (COP).

All major components in CPAP layers (L3 L6) and packages (P07-P17) will be summarized following the CPAP structure presented in [8].

II. CPAP Components for Cloud Computing

CPAP technological framework used to work in the cloud [3], [4] is shown in Fig. 1. Two types of components are presented - for classic and combined programming.

Components for classic programming allow the realization of classic programming techniques related to multi tier architectures; SOA [4] and CC [2] and are of little interest as regards programming automation. However, these components are the minimal body of technical means sine qua none standard software products would not operate. This group includes the following components:

- in layer L3: Package P07 - components C1 LDAP / AD Server, C2 Application Server, C3 RDB Server, C4 Search Engine, C5 eMail Server; in package P08 - components: C6 Identity Server, C7 Web Server, C8 Process Server, C9 Data Integration Server (case management);
- in layer L5: Package P10 - components C10 Training Manager, C11 Help Manager, C12 Payment Manager; in package P11 - components C13 Issue Manager, C14 Wiki Manager, C15 Conference Manager, C16 Calendar Manager; in package P12 – component C17 System Monitor; in package P13 - components: C18 Register Manager, C19 Contents Manager; in package P14 - components C20 Roles Manager, C21 Customer relationship Manager, C22 Human Resources Manager; in package P15 - components C23 Service Manager, C24 Business Rules Manager, C25 Codelist Manager; in package P16 - components C26 Developer graphic user interface display (GUID), C27 Portal, C28 System Administrator GUID, C29 End User GUID, C30 Reports GUID;
- in layer L6: Package P17 - components: C31 Authorization Integrator, C32 Legacy Systems Integrator, C33 Partner Systems Integrator, C34 Remote Systems Integrator.

Components for combined programming represent techniques for classic programming enriched with automation elements, e.g. dynamic reconfiguration of data structures or computing process management based on business rules or business process specifications, etc. Components of this type are the following:

- in layer L4: Package P09 – components C35 Process definition BPMN (Business Process Model and Notation), C36 Service definition UML (Unified Modeling Language), C37 RDB Model;
- in layer L5: Package P12 - component C38 Business Activity Monitor; in package P13 - component C39 Intelligent Documents Manager; in package P14 - component C40 Organization Manager, in package P15 – components C41 Process Manager;
- in layer L6: Package P17 - components C42 Enterprise Resource Planning, C43 Enterprise Service Bus.

This work is supported by the National Scientific Research Fund under the contract ДФНИ - И02/13.

I. N. Stanev is with the Informatics and Information Technologies Department, University of Ruse "Angel Kanchev", Ruse 7000, Bulgaria (tel.: 359-882117345; e-mail: instanev@gmail.com).

M. P. Koleva is with Computer Informatics Department, University of Sofia "St. Kliment Ohridski", Sofia 1000, Bulgaria (e-mail: marie.koleva@gmail.com).

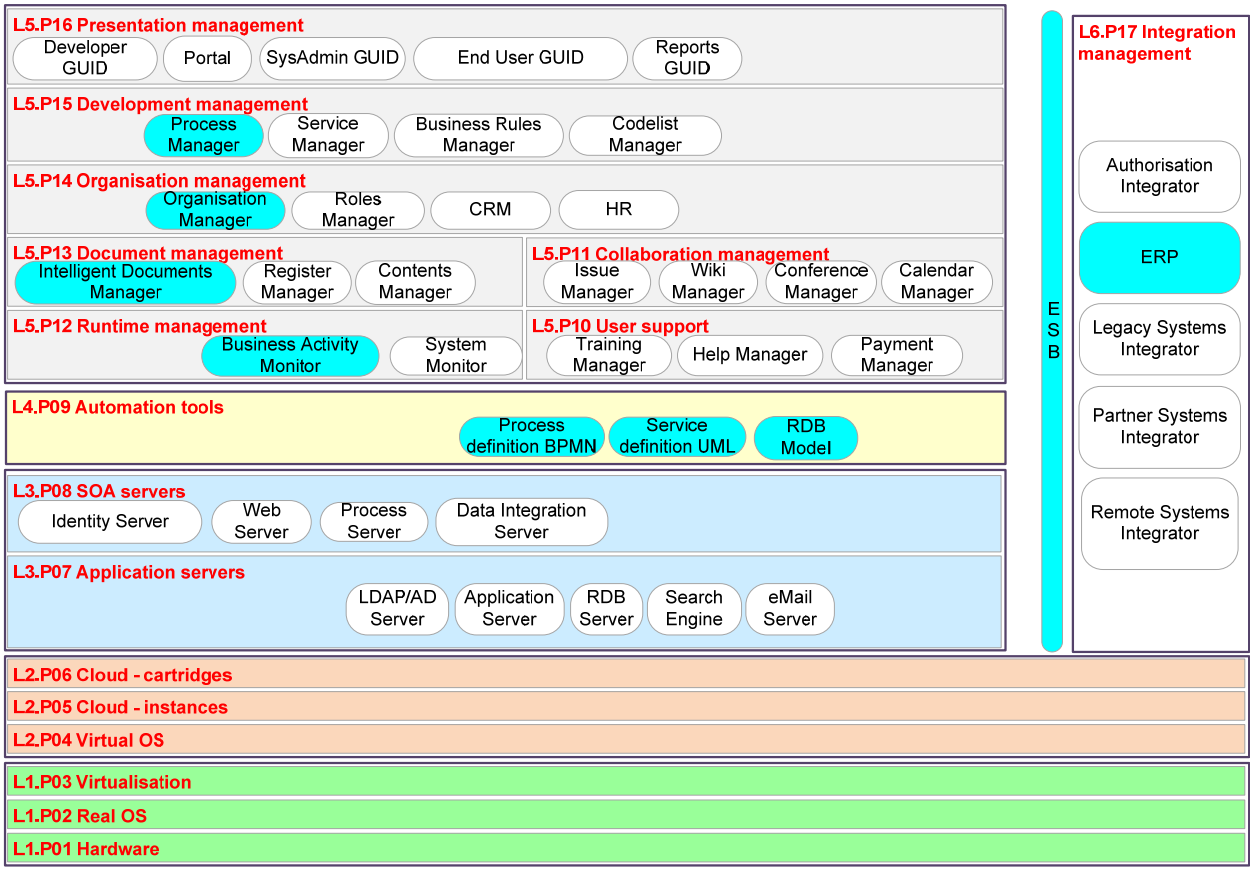

Fig. 1 CPAP technological framework used to work in the cloud

III. CPAP Components for Automated Programming

CPAP technological framework for automated programming (Fig. 2) add to the components presented in section 2 above another two types of components [1], [9], [11] – components for automated programming and components for ontology programming.

Components for automated programming represent automation tools based on the interpretation of formal models (e.g. direct generation of software products from UML, BPMN, EPC, fourth generation languages, formal methods, etc.). This type of components include:

- in layer L3: Package P07 - components: C44 Document Server, C45 Runtime Monitor, C46 ODB Server;
- in layer L4: Package P09 - components: C47 ODB Model, C48 Case definitions, C49 Interface Model, C50 Reports Model;
- in layer L5: Package P12 - component C51 Runtime Manager; in package P14 - component C52 Reports Manager; in package P15 - component C53 Forms Manager.

Components for ontology programming provide for automation through knowledge interpretation, learning and self-learning (e.g. generate software based on ontology descriptions, fuzzy interpreters of incomplete and inaccurate specifications, code generators working with natural language specifications, etc.). This type of components include:

- in layer L3: Package P07 - component C54 FullText Indexing Server; in package P08 - component C55 Ontology Server;
- in layer L4: Package P09 - component C56 Ontology Model;
- in layer L5: Package P15 - component C57 Knowledge base Manager; in package P16 - component C58 Ontology GUID;
- in layer L6: Package P17 - component C59 Semantic ESB.

IV. CPAP Contributions to Industrial Programming

In order to assess CPAP contributions to industrial programming an analysis based on quantitative and qualitative criteria is proposed in Table I. This analysis covers the following software products: Module for automated programming of robots [6], [7], [9], Intelligent Product Manual (IPM) [10], Intelligent Document Display (IDD) [1], Intelligent Form Generator (IFG) [12].

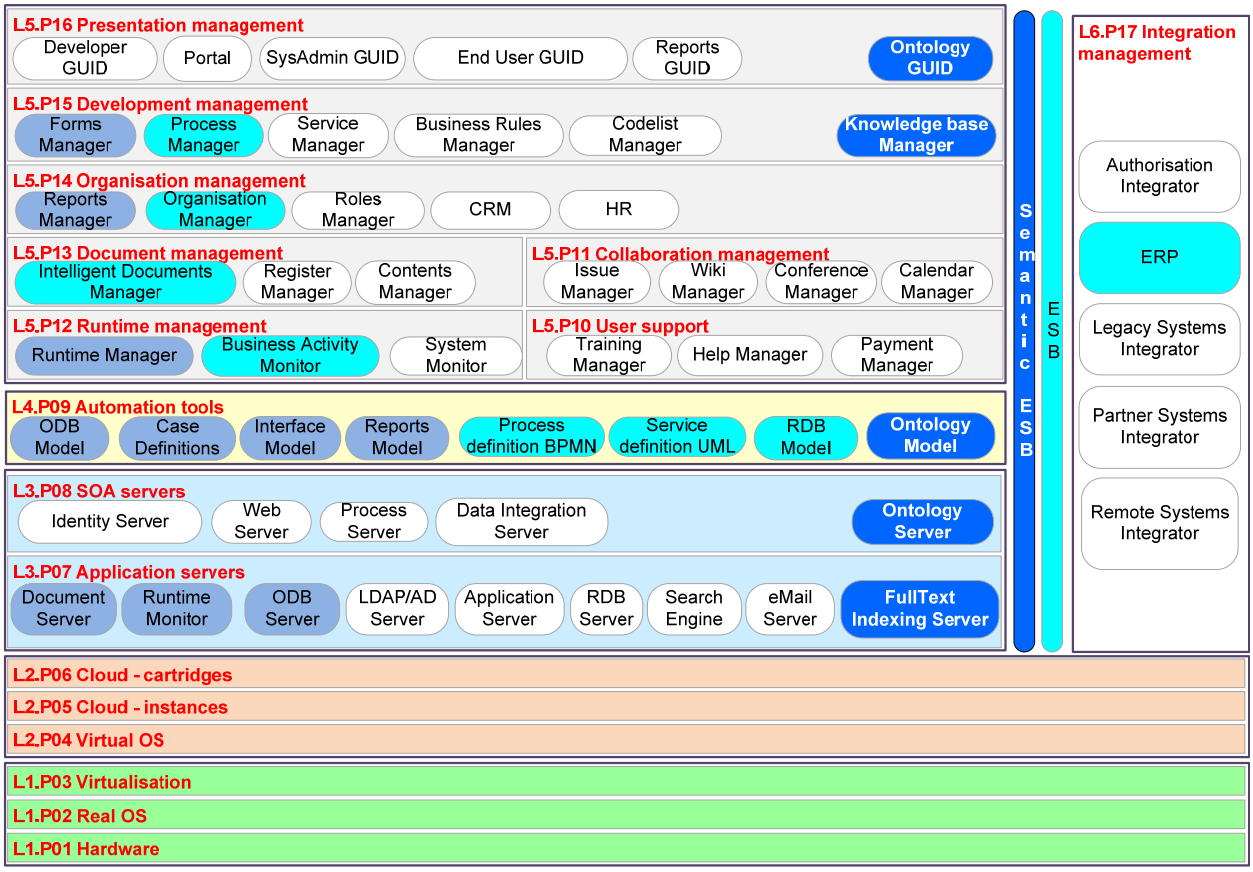

Fig. 2 CPAP technological framework for automated programming

TABLE I
CPAP CONTRIBUTIONS TO INDUSTRIAL PROGRAMMING

		KBASE applications			
Type	*Contribution*	*Module for automated programming of robots*	*Intelligent Product Manual*	*Intelligent Document Display*	*Intelligent Form Generator*
quantitative	product specification time reduced	√			
quantitative	programming time reduced	√	√	√	√
quantitative	COTS components integration time reduced			√	
quantitative	testing time reduced	√	√	√	√
quantitative	COTS components testing time reduced			√	
quantitative	IT team reduced	√			√
qualitative	adaptive to various domain areas	√	√	√	√
qualitative	adaptive to different end users	√	√		
qualitative	adaptive presentation to diverse standards		√		
qualitative	adaptive presentation to different media	√	√		
qualitative	real time code synchronization		√		√
qualitative	real time documents synchronization		√		
qualitative	prevent emergency system failure			√	√
qualitative	real time performance improvement			√	√

The following conclusions could be drawn as a result of the analysis of automated programming techniques used in the KBASE applications above:

(1) software development and testing time is considerably reduced;
(2) Only the first design phase (initial knowledge acquisition) requires increased resources, whereas the second design phase (reuse of accumulated knowledge) requires significantly reduced resources;
(3) COTS components integration and testing time is well reduced;
(4) software products quality is substantially increased, including capabilities for adaptation to new domain areas and different end users, restructuring to efficiently function on various media, improvement of version control process, etc.

V. Conclusion

The CPAP technological framework provides opportunities for cheaper and shorter development cycle as well as improvement of the quality of the software products since CPAP combines the advantages of different technologies for automated programming.

Satisfactory economic return on efforts needed for the introduction of automated programming tools is observed.

References

[1] EU IST-1999-20162 Development and Applications of New Built-in-Test Software Components in European Industries. Software Architecture. 2003

[2] Liu, F., et al. NIST Cloud Computing Reference Architecture. Gaithersburg: National Institute of Standards and Technology Special Publication 500-292. US Department of Commerce. Pp. 35 2011.

[3] Mell, P., T. Grance. The NIST Definition of Cloud Computing. Gaithersburg: National Institute of Standards and Technology NIST Special Publication 800-145 US Department of Commerce. Pp. 7 2011.

[4] Organization for the Advancement of Structured Information Standards (OASIS). Reference Model for Service Oriented Architecture. OAZIS. Pp. 31. 2006.

[5] Piprani, B., D. Sheppard, A. Barbir. Comparative Analysis of SOA and Cloud Computing Architectures Using Fact Based Modeling. Springer-Verlag Berlin Heidelberg: OTM 2013 Workshops Volume 8186 of the series Lecture Notes in Computer Science. Pp. 524–533. 2013.

[6] Stanev I. A Bulgarian Linguistic Processor Based on the Formal Model Control Networks - General Concepts. In proceedings of the CompSysTech'2002. Sofia. 2002. Pp. III.7-1 – III.7-5.

[7] Stanev I. Formal Programming Language Net. Part I – Conception of the Language. In proceedings of the CompSysTech'2001. Sofia. 2001. Pp. I.16-1 – I.16-5.

[8] Stanev I. M. Koleva, KBASE Technological Framework – Requirements. ICSII 2015: 17th International Conference on Semantic Interoperability and Integration. Rome. 2015 (submitted for publication).

[9] Stanev I. Method for Automated Programming of Robots. In Knowledge Based Automated Software Engineering. Cambridge Scholars Press. Cambridge. Pp.67 – 85. 2012.

[10] Stanev I., et al. 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, 2000. Belgrad. Pp. 1-153, 1-156.

[11] Stanev, I., K. Grigorova. KBASE Unified Process. Knowledge Based Automated Software Engineering. Cambridge Scholars Publishing. Cambridge. Pp. 1 – 19. 2012.

[12] EU OPAC Program K10-31-1/ 07-09-2010, Sub-project Д-26/30.05.2012 Realization of Priority Municipality Electronic Administrative Services. Software Architecture (Проект Реализиране на приоритетни електронни административни услуги на общински администрации. Софтуерна архитектура.) Ministry of Transport Information Technologies and Communications. Sofia. 2013.