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;

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;
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

<table>
<thead>
<tr>
<th>Type</th>
<th>Contribution</th>
<th>Module for automated programming</th>
<th>Intelligent Product Manual</th>
<th>Intelligent Document Display</th>
<th>Intelligent Form Generator</th>
</tr>
</thead>
<tbody>
<tr>
<td>quantitative</td>
<td>product specification time reduced</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>quantitative</td>
<td>programming time reduced</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>quantitative</td>
<td>COTS components integration time reduced</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>quantitative</td>
<td>testing time reduced</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>quantitative</td>
<td>COTS components testing time reduced</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>qualitative</td>
<td>IT team reduced</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>qualitative</td>
<td>adaptive to various domain areas</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>qualitative</td>
<td>adaptive to different end users</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>qualitative</td>
<td>adaptive presentation to diverse standards</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>qualitative</td>
<td>real time code synchronization</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>qualitative</td>
<td>real time documents synchronization</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>qualitative</td>
<td>prevent emergency system failure</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>qualitative</td>
<td>real time performance improvement</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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


