Integrating Workflow-Based Mobile Agents with Cloud Business Process Management Systems

Antons Mislevics and Janis Grundspenkis
Department of Systems Theory and Design,
Riga Technical University,
Meza ¼, Riga, LV-1048, Latvia
antons.mislevics@rtu.lv, janis.grundspenkis@cs.rtu.lv

ABSTRACT

While researching integration aspects of cloud Business Process Management Systems (BPMS) with on-premises applications, authors have identified various scenarios that cannot be effectively addressed using existing integration solutions and patterns. Mobile Agents (MAs) were proposed as the technology addressing these integration challenges [1]. During the research, existing MA platforms were analyzed what led to conclusion that none of them is fully addressing specific requirements of end-users in BPM scenarios. As a result, a new approach for designing and executing mobile agents was proposed. This approach is based on workflows and enables end-users having no programming skills to design MAs what is critical in BPM scenarios [2]. In addition existing researches cover different scenarios of how intelligent agents can be used in BPM space, however these do not address specific aspects of MA integration with cloud BPMS, what is critical for proposed solution. During analysis of this subject, authors have identified required integration components, and are proposing an algorithm that allows to determine how these components should be implemented based on technical specification of cloud BPMS. AgentWF prototype provides a sample implementation of all components, and demonstrates how MA system should be integrated with cloud BPMS using proposed approach. Proposed approach is extendable and may be reused for integrating MAs with cloud systems in other domains as well.

KEYWORDS


1 INTRODUCTION

Business Process Management Systems (BPMS) became more popular in recent years [64]. Market analysts have observed this trend especially in service industries, where human productivity and effectiveness are critical to process performance. Analyzing future of BPMS researchers and market analysts agree that cloud computing environments will be widely used to adopt BPMS applications [10, 30, 31]. As any successful BPMS implementation relies on solid integration with existing line of business (LOB) applications and organizational data sources, moving BPMS to the cloud puts the focus on “cloud to on-premises” integration questions. While researching integration aspects of cloud Business Process Management Systems (BPMS) with on-premises applications, authors have identified various scenarios that cannot be effectively addressed using existing integration solutions and
patterns. Mobile Agents (MAs) were proposed as the technology addressing these integration challenges [1]. During the research, existing MA platforms were analyzed what led to conclusion that none of them is fully addressing specific requirements of end-users in BPM scenarios - also some existing MA platforms are providing graphical user interface, it is expected that user creating MA has at least some understanding of programming concepts. In majority of systems programmer’s involvement is also required to handle MA migrations. As a result, a new approach for designing and executing mobile agents was proposed. This approach is based on workflows and enables end-users having no programming skills to design MAs what is critical in BPM scenarios [2]. In addition existing researches cover different scenarios of how agent systems can be used in BPM area, however these do not address specific aspects of integration with cloud BPMS, what is critical for proposed solution. During analysis of this subject, authors have identified four required integration components, and are proposing an algorithm that allows to determine how these components should be implemented based on technical specification of cloud-BPMS.

The paper is organized as follows. The next section provides an overview of BPM concept and implementation aspects. Typical scenarios of using intelligent agents in BPM systems are presented in Section 3. Section 4 covers main concepts of MA systems, provides an overview of existing MA implementations and introduces newly proposed workflow-based MA development and execution approach. Architecture of proposed MA solution is presented in Section 5. This section also provides in-detail overview of proposed integration approach and analysis algorithm used to define integration components. Finally, a prototype used to validate proposed architecture and integration analysis algorithm is presented.

2 BUSINESS PROCESS MANAGEMENT SYSTEMS

Business Process Management Systems (BPMS) are used to control, analyze and manage business processes in organizations. BPMS help to reduce the amount of administrative effort and focus on the processes which add value. A modern BPMS should provide tools for defining and executing business processes, performance meter counters, management tools and support real time changes in processes [21]. The system should also contain a set of tools to integrate with external systems through the variety of different protocols. Besides it should integrate with other BPMS, and support cross-organization business processes [22]. In addition, a modern BPMS should adapt to changes in the environment [3]. The most common architecture for implementing BPMS is Workflow-Oriented BPMS (WBPM) [3]. WBPM is primary focusing on a workflow itself. BPMS of this kind are also called centralized, because these systems have a single process which ensures execution of the entire business process. Workflow is the automation of a business process [4]. A typical workflow consists of the following elements [5]:

- Messages – ensure communication between employees (e-mails, files, paper documents, etc.);
- Activities – tasks, which must be completed by employees after
receiving a message from the system;
- Business rules – describe logics of business processes;
- Flowcharts – specify how processes flow in the organization.

Workflow building blocks in WBPMS are activities. These are used to create flowcharts and define the flow of work in the organization. Activities may be simple, as task activities, or complex, as calls to external web services (providing possibilities to integrate with other enterprise systems or BPMS) [6, 7].

Figure 1 shows common architecture of a centralized BPMS.

![Figure 1. Centralized business process management system (adopted from [3])](image)

WBPMS are straightforward to implement, and allow organizations to gain the following advantages [4]:
- The system helps to define and formalize business processes. It makes clear to everyone what is happening in the organization and what role each individual plays;
- It is easier to analyze and optimize processes when they are well defined;
- Complicated business processes are broken down to simpler parts. It is possible to analyze and improve each part separately;
- It helps to monitor daily operations;
- It integrates different applications (which may be even hosted on different platforms) in order to ensure the single business process;
- It provides personal workspace for each employee (typically, a personal workspace contains a list of actual tasks, and supporting information – documents, reports, etc.);
- The system separates business logics of the process from actual work items. Each employee should not know each business process in detail. He should focus on his own tasks only.

These advantages are obvious and well describe why BPMS are becoming more popular. But taking a closer look at organizations’ business processes (especially at more complex ones), opens up the whole set of issues, with
which WBPM cannot deal so well [3, 4, 6, 7, 8, 9]:
1. The system is based upon the single central workflow server. It may be inappropriate in scenarios, where company has multiple branches in different offices, and each branch is managing its own business processes;
2. Insufficient automation: a system describes the logics of the business process – a sequence of steps that should be performed to achieve the goal, but all tasks are processed by employees;
3. The system is not flexible enough: the whole process with all branches should be defined;
4. No resource management: the system is not controlling whether all resources required to complete the task are available. It assumes that this was taken into consideration while planning the process. That’s why the system is not adaptive enough;
5. No knowledge of process semantics: the system has lack of information about the business process itself. This information cannot be used to make situation appropriate decisions;
6. Lack of well-defined protocols to exchange information among systems, and integrate different BPM systems to support cross-company workflows;
7. Not trivial error handling: an error in each step results in an error in the process. Developers should predict possible errors and implement error handlers. Unhandled exceptions will result in aborting the whole process. It gets even more complicated, when developers have to deal with parallelism and complex business logics;
8. Users cannot adjust execution of the workflow, thus in a non-standard situation a user cannot perform some action, which was not specified while defining the workflow;
9. Typically centralized systems provide limited performance, scalability and reliability;
10. Complicated versioning and upgrade process.

The majority of these limitations are obvious and are caused by specifics of workflow oriented systems and design of centralized BPMS. To improve flexibility of the system and make business processes more adaptable to changes in external environment, several alternative BPMS implementation approaches are proposed. One of them is Subject Oriented Business Process Management (S-BPM). This describes simplified approach to analyze and execute business processes that is radically different from widely adopted workflow oriented approach [10, 11, 12]. Other modern concept is Social BPM. It extends formal workflow-oriented BPMS approach with capabilities of social technologies – collaboration, social networks and instant messaging [10, 13, 14]. Dynamic BPM is another modern approach that is worth noting. It combines various methods and technologies to make business processes more adaptable, and simplify implementing modifications in business processes [10]. Various authors are proposing intelligent agents as a technology that may improve flexibility of BPMS, make processes more adaptable to changes in external environment, and also ensure resources management and automation capabilities.
Usage of intelligent agents in BPM scenarios is covered in the next section. Analyzing future of BPMS researchers agree that cloud computing environments will be used to adopt BPMS applications [30, 31]. Market analysts observe the same trend and pay special attention to cloud BPM offerings, expecting that these will achieve mainstream adoption within two to five years [10]. As any successful BPMS implementation requires establishing solid integration of business processes with existing line of business (LOB) applications and organizational data sources, moving BPM to the cloud puts the focus on “cloud to on-premises” integration questions. Similar questions are addressed in many studies around integrating cloud Software-as-a-Service (SaaS) applications with on-premises systems and data sources [32, 33, 34, 35, 36, 37, 38]. Nevertheless, integrating BPM cloud applications with on-premises LOB applications has specific scenarios and requirements that cannot be effectively addressed by using existing technologies and patterns [1], which are:

- Performing complex computations close to data sources in internal network (dealing with large amounts of data);
- Performing complex transformations and computations with data stored in on-premises application in internal network (for security and privacy reasons);
- Implementing rapid changes in integrations (to adapt business process to changes in the environment);
- Accessing legacy systems and specific devices that are deployed on-premises and have no web services or database interface.

These integration scenarios may be addressed by using mobile agents (MA) [1].

3 INTELLIGENT AGENTS IN BUSINESS PROCESS MANAGEMENT

There are two scenarios, how agents can be used in business process management: agent-supported BPM and agent-driven BPM. In the first scenario intelligent agents support business processes, which are operated by the BPMS. In the second – agents are driving the process, encapsulating the whole logic.

3.1 Agent-supported business process management

In agent-supported BPM scenario agents are working with BPMS to support execution of business processes. Typical roles of agents in this scenario are [4]:

- User interface: the agent helps user to complete tasks assigned during a business process (filtering e-mails, auto-responding to incoming e-mails, reminding about upcoming events, etc.);
- Autonomous work item processing: the agent completes tasks without user interaction;
- Interface to external systems: the agent ensures communication between BPM system and external applications.

Figure 2 shows agents of different types in action. Agent A1 is a user interface, agents A2 and A3 autonomously process work items, but agents A6 and A7 represent interfaces to external system. To enable this scenario, BPMS must
provide interfaces for integration with external systems. Using these interfaces, agents may connect to BPMS to access data and perform operations.

![Diagram](image)

**Figure 2.** Agent-supported business process management (adopted from [3])

### 3.2 Agent-driven business process management

In agent-driven BPM approach agents are controlling a flow of the process. Each agent represents a specific part of a business process. Collaboration of agents ensures the complete process [17]. To achieve this, entire business process must be split into separate functions and operations. Then an intelligent agent for each function and operation is implemented. Each agent is responsible for one or multiple workflow steps, and communicating with other agents decides which step should be executed next [4]. Agents may be different in implementation and reasoning complexity. For example, the simplest agent may delegate some task to an employee, wait when the task is completed, analyse outcome and decide which workflow steps must be executed next. This idea is depicted in Figure 3.

![Diagram](image)

**Figure 3.** Multiagent business process management (adopted from [3])
To enable this approach, BPMS must be implemented in specific way, named Agent-BPMS architecture [23, 24, 25, 26, 27, 28, 29]. Agent-BPMS is a distributed system that has no central workflow processing engine. Distributed architecture allows to integrate processes operated in different BPMS [4]. The BPM system becomes dynamic and expandable. Separate agents can be modified, added or removed without affecting operation of the entire system. Agent-driven BPM approach allows to achieve process automation, as autonomous agents can perform tasks without human interaction. Besides that, agents may represent resources, ensuring advanced resource management scenarios. They may determine changes in environment and adapt execution of the whole business process. Agents may learn over time, make better decisions and thus achieve their goals more effectively. They are focused not on executing all steps of the process, but on achieving some business goal. Agents may autonomously search for better way to achieve the goal, thus automatically improving effectiveness of the whole business process [7]. Figure 4 shows the idea of agent-driven BPM approach.

![Figure 4. Agent-driven business process management (adopted from [3])](image)

### 3.3 Agents and cloud business process management systems

In cloud-BPMS scenarios developers cannot affect architecture and implementation of BPMS system, thus cannot implement full agent-driven BPM scenario. However, cloud BPMS usually provide various integration interfaces [36], which may be used to implement agent-supported BPM scenarios. It also allows to combine agent-supported and agent-driven BPM approaches, to ensure that on some stage business process is executed in agent-driven mode (for example, to ensure that sub-process is adapting to changes in external environment). Implementing this scenario requires to combine agent-supported and agent-driven BPM approaches. On specific stage of the workflow a task for an agent is generated. The agent receives a task (as in agent-supported BPM scenario) and collaborating with other agents, initiates execution of sub-process (as in agent-driven BPM scenario). When execution
of sub-process is completed, agent completes initially assigned task, and the execution of the workflow is continued in BPMS. Figure 5 shows example of this scenario.

![Diagram](image.png)

**Figure 5.** Combining agent-supported and agent-driven business process management scenarios

The system addressing integration challenges of cloud-BPMS with on-premises enterprise systems and devices proposed by authors [1] is based on mobile agents (MA) and is implemented using agent-supported BPM approach. It also has some aspects of combined approach, as implementation of MAs is workflow-based [2]. This means, that MA may implement sub-process that is executed using agent-driven approach. Workflow-based MA approach is described in next section. Architecture of proposed MA system and cloud-BPMS integration aspects are covered in section 4.

4 WORKFLOW-BASED MOBILE AGENTS

4.1 Concepts of Mobile Agents Systems

Mobile agents (MA) are self-contained and identifiable computer programs, that can move within the network and act on behalf of the user or another entity. The following three capabilities are a foundation for MAs: mobile code, mobile computation, and mobile state [39]. Mobile computation involves moving a computation from one system to another. Mobile code is the ability to move code from one system to another what allows systems to be extremely flexible. Mobile state is an evolution of state capture, which allows the execution state of a process to be captured. State capture is very useful for long-running processes, because in the event of failure, the execution of a process can be restarted from the last saved state. Mobile state allows the movement of the execution state of an agent to another system for continued execution. Not all MA systems provide support for state mobility [39, 40]. The term strong
mobility is used to describe systems that can capture and move execution state with the agent. In agents that support strong mobility it is guaranteed that all variables will have identical values after moving to another host. Weakly MA systems usually support the capture of most of a program’s data, but restart the program from a predefined program point and thus require some programmer involvement at each migration. The advantage of strong mobility is that the result of migrating is well defined and easier to understand.

There are two main scenarios that must be addressed by MA systems [41]: creating an agent and transferring an agent. To create an agent, MA system creates an instance of the agent class on a default host, or a host the client specifies. The agent class specifies the implementation of the agent. During the execution MA may request the source agent system for a transfer to another host or agent system. To start this process host or agency on a sender’s side needs to initiate agent’s transfer: suspend the agent, identify transferable agent's state, serialize the agent class and state, encode it for the chosen transport protocol, provide authentication info to the server, and transfer the agent. To receive an agent destination host needs to authenticate client, decode the agent, deserialize the agent class and state, instantiate the agent, restore the agent state and resume agent execution. Serialization is the process of storing an agent in a serialized form, sufficient to reconstruct the agent. Deserialization is the inverted process.

4.2 Existing Implementations

There are three broad approaches to mobility implementation [39, 40], named remote evaluation, code on demand, and mobile agents. In remote evaluation scenario an application invokes services on another node by specifying the code, as well as the input data, necessary to invoke the service. The code and input data are sent to the remote node, and the remote node then executes the code and sends the output data back to the client. In code on demand approach code fragments are requested as they are needed, and dynamically compiled, verified and linked into a running system. In mobile agents approach rather than simply moving code and input data (as in remote evaluation scenario), MAs view a computation as a single entity and support the migration of a complete program to another node. MA infrastructures can be implemented as extensions to code-on-demand systems [42, 43, 44, 45, 46], as extensions to remote evaluation systems, or as new programming languages [47, 48]. In general, three targets for MA system design and implementation exist: using or creating a specialized language (language features provide the requirements of MA systems), as operating system (OS) services or extensions (what allows to take advantage of existing OS features), or as specialized application software [49].

There are many MA platforms available, with over 100 such systems implemented in the last five years. However, according to researchers the following may be considered as the most popular: Telescript, NOMADS, SafeTCL, D'Agents, JavaSeal, Mole, Aglets, Lime, Messenger, JADE, Voyager, TACOMA, Grasshopper, SPRINGS, MAPNET and EtherYatri.NET [39, 50, 51]. Analyzing existing systems researchers conclude that weak mobility is by far the
predominant approach [39]. Java is still the most popular agent implementation language, mainly due to both the popularity of the language and its support for dynamic loading and advanced security features [39, 50]. Microsoft .NET is more recent technology with the Common Language Runtime, which is also based on a virtual execution environment and supports the dynamic loading of programs. .NET is becoming more popular in recent MA implementations, however in the overall picture Java still prevails [39, 51]. The main advantage of building MA system on an existing language is that the existing technology can be leveraged. This definitely comes at the price of some conceptual complexity, since the system designer must deal with inherent limitations of the language.

Analyzing agent development experience in existing implementations, it should be noted that also some platforms are providing graphical user interface, it is expected that user creating MA has at least some understanding of programming concepts. Details of saving agent’s state and restarting execution are also not obvious for end-users with no technical knowledge. In systems implementing weak mobility approach programmers involvement is also required to handle MA migrations. To address these issues authors are proposing the approach for designing and executing mobile agents, which is based on workflows [2].

4.3 Workflow-Based Approach

Workflows are considered a standard mechanism used to develop long running processes. Recently, workflow platforms become widely available to developers as part of software development frameworks (for example, Microsoft is providing Workflow Foundation in .NET Framework [52]). In addition to features already available in specific software development frameworks, workflow platforms provide support for state persistence, resuming execution from persistence point and graphical definition tools. These tools enable users without programming knowledge to define complex workflow behaviors. Authors believe that with minimal extensions workflow platforms built on top of popular software development frameworks may become a solid foundation for building user-friendly MA systems.

MA design and initiation process is organized in three steps (Figure 6):

1. Business analyst defines agent behavior using special application Agent Designer (Figure 7). Application allows defining sequential agent behaviors as steps of a workflow. As a result Agent Definition package is created (.awfd file);
2. Agent Definition package is then uploaded to BPMS;
3. BPMS initiates execution of an agent. At this point Agent Instance package is created (.awfx file)

\[ \text{Agent Designer} \rightarrow 1. \text{Create} \rightarrow \text{Agent Definition} (* .awfd) \rightarrow 2. \text{Load to Agent Instance} (* .awfx) \rightarrow \text{BPM system} \rightarrow 3. \text{Start execution} \rightarrow \text{Agent Instance} (* .awfx) \]
from previously defined Agent Definition package. Figure 7 shows Agent Designer application developed as part of AgentWF prototype.

Figure 7. AgentWF Agent Designer application

5 PROPOSED SOLUTION

5.1 Overall Architecture
Overall architecture of proposed MA solution MA solution is shown on Figure 8. Organization is using a cloud BPM through an existing internet connection. Organization’s internal environment consists of employees, various devices, business systems, and data storages. To integrate this internal environment with cloud BPMS a MA infrastructure should be deployed in organization. On client side this requires installing agent execution environment software on devices. In addition a highly available communication proxy (agency) should be established. This may be deployed as part of cloud BPM system, part of organization’s internal infrastructure, or a separate cloud service. The communication proxy should provide agent queue that receives agents and routes them to appropriate destination host (BPM, or agent host). The system is implemented using agent-supported BPM approach. Agency connects to BPM system through BPM protocol handler implemented specifically for this system. Execution of MA is initiated in BPMS. MAs are traveling from BPMS
to appropriate agent host and back through agency. While transporting agents are packaged to special container (agent package). When the package reaches destination host it becomes an agent and runs in the execution environment.

Proposed architecture is based on the following standards and open specifications: OPC [53], XML [54], XML encryption [55], XAML [56, 57], JSON [58], X509 certificates (signing and encryption) [59, 60]. It was developed for integrating on-premises systems with cloud-BPM systems [1]. However, with minimal adjustments proposed approach may be reused in other domains as well. MA system includes the following components: (Figure 9):

- Agencies – establish communication between systems and agent hosts;
- Agent Hosts and Executors – execute MAs on devices;
- Protocol handlers – used to connect to initiating system from agencies;
- Mobile agents – agents implementing strong mobility;
- Agent designer – application used to create MAs in graphical environment;
- Monitoring tools – applications used to monitor agencies and agent hosts;

![Diagram](image-url)
- BPM extensions – provide integration capabilities for specific BPM system (on BPM side).

Relations between these components are show on Figure 9. The system is designed the way that all communications to download and upload MAs are established from client side only: agency connects to initiating system, agent host connects to agency. Communications between modules may be established through widely adopted protocols (for example, http or https). This allows deploying the system in highly dynamic environments, or ones having strict security requirements.

![Figure 9. Components of the system (adopted from [1])](image)

### 5.2 Integration with cloud business process management system

The following four components are required to integrate MA system with cloud-BPMS:

1. **Protocol Handler** – ensures communication with specific BPMS;
2. **Agent Definitions Store** – stores agent definitions;
3. **Agent Instance Store** – stores agent instances;
4. **Workflow Actions** – allows to use MA specific activities while defining workflow models (for example, initiate MA execution, or handle MA execution result).

Figure 10 shows these components and their relations with agency and agent designer application. **Protocol Handler** is mandatory component that is required to integrate MA system with BPMS. To implement Protocol Adapter developers use integration interfaces provided by BPMS. If BPMS is not providing such interfaces, implementing Protocol Adapter is not possible. All communication between agency and BPMS happens through Protocol Adapter.
**Agent Definitions and Instances Stores** are mandatory components that should be implemented as extensions to BPMS. In this case workflow models and definitions for MAs used in these are stored in single location (cloud-BPMS). This also allows to avoid developing separate mechanism to transfer MA execution results back to BPMS, as this is a part of MA instance stored in BPMS. Agent definitions are stored as MA package files (.awfd). Agent instances may be stored as MA package file (.awfx) or as unpacked collection of files (this simplifies analyzing MA execution results, as raw files are available and handling MA package is not required).

If storing Agent Definitions and Instances in BPMS is not possible, separate storage must be implemented. This should address the following requirements:
- It allows to store MA package files (.awfd and .awfx);
- It allows to publish MA definitions developed in Agent Designer application;
- It provides communication protocols through which agency may download MA definition packages and upload MA instances.

In this scenario it is also required to implement components that ensure information exchange between agency and Agent Definition and Instances Stores. In addition, it is also required to implement mechanism that transfers MA execution result into BPMS.

**Agent Workflow Actions** allow using MA specific activities while defining workflow models. Implementing the following activities is recommended:
- MA initialization, specifying agent definition and initiation parameters;
- Starting MA execution;
- Handling MA execution results.

If BPMS does not provide mechanisms to extend workflow models with custom activities, MA integration should be implemented using existing workflow activities. For example, integration may be achieved by using standard workflow tasks:
- To initiate MA execution new task is created, including agent initiation parameters in task description;
- When MA execution is completed, task status is changed to completed, and agent execution

---

**Figure 10. Integration with cloud-BPMS**
results are included in metadata of the task.

Figure 11 shows proposed algorithm that allows to determine how BPMS integration components should be implemented based on technical specification of cloud BPMS.

![Algorithm Diagram]

**Figure 11.** Algorithm used to analyse cloud-BPMS integration options

## 6 VALIDATION OF PROPOSED ARCHITECTURE AND INTEGRATION ANALYSIS ALGORITHM

To validate proposed architecture MA system prototype called AgentWF was implemented (see Figure 7) [1, 2]. It is built on Microsoft .NET Framework 4.0 technology. This allows to use standard security capabilities (like process isolation and code access security) to ensure secure execution of MAs. This also allows extending solution later using standard .NET mechanisms. Agent executors are built using add-in framework [61]. This allows to ensure
security by executing MA in a separate process with limited permissions. Add-in framework also provides native mechanism for versioning agent executors what will become very important over time when the format of agent package will evolve. Workflow design and execution is performed using Workflow Foundation 4.0 platform [52]. State persistence services provided by the platform allow implementing strong agent mobility. The system is extendable. Developers can implement custom Workflow Foundation 4.0 activities that may be used as additional actions while designing agent behaviors.

To validate proposed cloud-BPMS integration approach (Figure 10) integration with SharePoint Online (Office 365) system was implemented [62]. Using proposed analysis algorithm (Figure 11) and SharePoint Online technical documentation [63] the following implementation approach was selected:

- Protocol Handler connects to BPMS using SharePoint Online web services interfaces;
- Agent definitions are stored as agent package files in SharePoint documents library;
- Agent instances are stored in SharePoint documents library as upackaged files (as SharePoint Online does not provide mechanisms to handle OPC packages);
- Custom agent workflow actions are implemented as sandbox SharePoint Designer Workflow Actions, what allows these to be used defining SharePoint Online workflow models.

Figure 12 shows agent definition and instance stores in SharePoint Online. Example of using custom workflow actions are shown on Figure 13.

![Image](image_url)

**Figure 12.** Agent instance and definitions stores
7 CONCLUSIONS

Business Process Management Systems (BPMS) became more popular in recent years, especially in service industries, where human productivity and effectiveness are critical to process performance [64]. Nowadays the most common architecture for implementing BPMS is Workflow-Oriented BPMS (WBPMS) [3], that is primarily focusing on workflow itself, and has single process that ensures execution of entire business process. However, this architecture has a set of limitations, the majority of which are caused by specifics of workflow oriented systems and design of centralized BPMS. To improve flexibility of the system and make business processes more adaptable to changes in external environment, several alternative BPMS implementation approaches are proposed: among these, Subject Oriented BPM [10, 11, 12], Social BPM [10, 13, 14] and Dynamic BPM [10]. Various authors are proposing intelligent agents as a technology that may improve flexibility of BPMS, make processes more adaptable to changes in external environment, and also ensure resources management and automation capabilities [4, 15, 16, 17, 18, 19, 20].

Analyzing future of BPMS researchers and market analysts agree that cloud computing environments will be used to adopt BPMS applications [10, 30, 31]. As any successful BPMS implementation requires establishing solid integration with existing line of business (LOB) applications and organizational data sources, moving BPM to the cloud puts the focus on “cloud to on-premises” integration questions. Similar questions are addressed in many studies around integrating cloud Software-as-a-Service (SaaS) applications with on-premises systems and data sources [32, 33, 34, 35, 36, 37, 38]. Nevertheless, integrating BPM cloud applications with on-premises LOB applications has specific scenarios and requirements that cannot be effectively addressed by using existing technologies and patterns [1]. Authors propose mobile agents as the...
technology that may address these integration challenges [1].

There are two scenarios, how intelligent agents can be used with business process management systems: agent-supported BPM and agent-driven BPM. In the first scenario intelligent agents support business processes, which are operated by the BPMS. In the second – agents are driving the process, encapsulating the whole logic. To enable agent-driven BPM approach, BPMS must be implemented in specific way, named Agent-BPMS architecture [23, 24, 25, 26, 27, 28, 29]. In cloud-BPMS scenarios developers cannot affect architecture and implementation of BPMS system, thus cannot implement full agent-driven BPM scenario. However, cloud BPMS usually provide various integration interfaces [36], which may be used to implement agent-supported BPM scenarios. It also allows to combine agent-supported and agent-driven BPM approaches, to ensure that on some stage business process is executed in agent-driven mode.

MA system addressing integration challenges of cloud-BPMS with on-premises enterprise systems and devices proposed by authors [1] is implemented using agent-supported BPM approach. It also has some aspects of combined approach, as implementation of MAs is workflow-based [2]. This means, that MA may implement sub-process that is executed using agent-driven approach.

The following four components are required to integrate MA system with cloud-BPMS: Protocol Handler, Agent Definitions Store, Agent Instance Store and Workflow Actions. Authors are proposing algorithm that allows to determine how these integration components should be implemented based on technical specification of cloud BPMS. Proposed architecture was validated by implementing prototype called AgentWF [1, 2]. It also allowed to validate proposed cloud-BPMS integration approach by implementing integration with SharePoint Online (Office 365) system [62]. During implementation proposed analysis algorithm and SharePoint Online technical documentation [63] were used. This allowed to select implementation approach for all required components.

ACKNOWLEDGMENT

This work has been supported by the European Social Fund within the project “Support for the implementation of doctoral studies at Riga Technical University”.

REFERENCES


12. Metasonic: http://www.metasonic.de/


33. Scheibler T., Mietzner R., Leymann F., EAI as a Service – Combining the Power of...
Executable EAI Patterns and SaaS, IEEE Communications Society, 2008.
38. Ottinger J., Software as a Service Integration via Mule [Internet]. Available from: http://mule.mulesource.org/
43. Binder, W., Design and implementation of the J-SEAL2 mobile agent kernel, The 2001 Symposium on Applications and the Internet, 2001
44. Lange, D., Oshima, M., Programming and deploying Java Mobile Agents with aglets, Addison Wesley, 1998
55. XML Encryption Syntax and Processing: http://www.w3.org/TR/xmlenc/
58. JSON: http://json.org/