

# BPMS-RA: a novel Reference Architecture for Business Process Management Systems

**Citation for published version (APA):**

Pourmirza, S., Peters, S., Dijkman, R., & Grefen, P. (2019). BPMS-RA: a novel Reference Architecture for Business Process Management Systems. *ACM Transactions on Internet Technology*, 19(1), Article 13. <https://doi.org/10.1145/3232677>

**Document license:**

TAVERNE

**DOI:**

[10.1145/3232677](https://doi.org/10.1145/3232677)

**Document status and date:**

Published: 01/02/2019

**Document Version:**

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

**Please check the document version of this publication:**

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

**General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

[www.tue.nl/taverne](http://www.tue.nl/taverne)

**Take down policy**

If you believe that this document breaches copyright please contact us at:

[openaccess@tue.nl](mailto:openaccess@tue.nl)

providing details and we will investigate your claim.

# BPMS-RA: A Novel Reference Architecture for Business Process Management Systems

SHAYA POURMIRZA, SANDER PETERS, REMCO DIJKMAN, and PAUL GREFEN,  
Eindhoven University of Technology

---

A growing number of business process management systems is under development both in academia and in practice. These systems typically are based on modern system engineering principles, such as service-oriented architecture. At the same time, the advent of big data analytics has changed the scope of these systems, including functionality such as data mining. However, existing reference architectures for business process management systems date back 20 years and, consequently, are not up-to-date with these modern developments. To fill the gap, this article proposes an up-to-date reference architecture, called BPMS-RA, for modern business process management systems. BPMS-RA is based on analysis of recent literature and of existing commercial implementations. This reference architecture aims to provide a guideline template for the development of modern-day business process management systems by specifying functions and interfaces that need to be provided by these systems as well as a set of quality criteria that they need to meet.

CCS Concepts: • **Computer systems organization** → *Special purpose systems*;

Additional Key Words and Phrases: Business process management systems, reference architecture, workflow

## ACM Reference format:

Shaya Pourmirza, Sander Peters, Remco Dijkman, and Paul Grefen. 2019. BPMS-RA: A Novel Reference Architecture for Business Process Management Systems. *ACM Trans. Internet Technol.* 19, 1, Article 13 (February 2019), 23 pages.

<https://doi.org/10.1145/3232677>

---

## 1 INTRODUCTION

*Business Process Management (BPM)* is a discipline that aims at overseeing the activities performed in an organization to ensure the quality of outcomes and to discover improvement opportunities (Dumas et al. 2013). Business process management systems (BPMSs) are information systems that interpret business processes to ensure that the activities specified therein are properly executed and monitored (Baumgrass et al. 2015a).

The architecture of BPMSs has been a subject of research since the 1990s. However, apart from the long-established Workflow Reference Model (Hollingsworth 1995) and the Mercurius reference architecture (Grefen and Remmerts de Vries 1998), there is a general lack of modern reference architectures for BPMSs. In particular, the existing reference architectures were developed before the shift to Service Oriented Architectures and before the rise of Business Process Analytics and

---

Authors' addresses: S. Pourmirza, S. Peters, R. Dijkman, and P. Grefen, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands; emails: {s.pourmirza, s.p.f.peters, r.m.dijkman, p.w.p.j.grefen}@tue.nl.

Current Address: S. Pourmirza, Building Blocks, Sint Josephstraat 139, 5017 GG Tilburg, The Netherlands; email: shaya@building-blocks.nl.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2019 Association for Computing Machinery.

1533-5399/2019/02-ART13 \$15.00

<https://doi.org/10.1145/3232677>

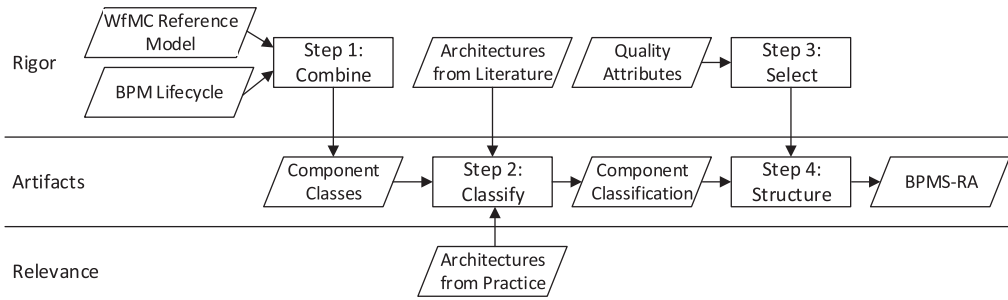


Fig. 1. BPMS-RA’s design approach with produced artifacts.

Process Mining. Therefore, a revision of existing reference architectures is due that considers at least these functions. This article fills that gap by proposing a reference architecture for BPMSs, called *BPMS-RA*, which emerges from both research in the BPM community and from existing BPMS architectures from practice. An implementation of the proposed architecture has already been developed in the GET Service platform (GET Service Consortium 2013; Baumgrass et al. 2015b), as discussed in the evaluation section of this article.

We define a BPMS reference architecture as a predefined guideline for the architecture of a BPM system, where the structures, elements, and relationships among the elements provide a template for concrete architectures (Bachmann et al. 2011). This template must be defined in such a way that concrete architectures can be instantiated from a reference architecture by implementing and modifying it according to the specific context of that concrete architecture. Accordingly, the development process of a reference architecture is categorized into two groups (Grefen 2015). On the one hand, the design principles that are provided by a reference architecture can be mined from best practices in a specific domain, in which case the resulting reference architecture is practice driven. On the other hand, a reference architecture can be designed before the existence of the practical best practices and inspired by existing research, which results in a research-driven reference architecture. As an example, the Workflow reference model was designed according to the practice-driven approach whereas the Mercurius reference architecture was designed according to the research-driven approach.

Using a design science methodology (Hevner et al. 2004), we combine both the research- and practice-driven approaches to design the *BPMS-RA*. Figure 1 illustrates our approach. It illustrates that the designed artifact is the *BPMS Reference Architecture*, which is designed in four steps. The literature that is used to shape the artifact represents the “rigor” dimension of the methodology, while the existing concrete architectures that are used represent the “relevance” dimension.

In the first step, we introduce a *BPMS component classification* by combining (i) the phases that constitute the BPM life-cycle (van der Aalst et al. 2007) and (ii) the components that are identified in the Workflow reference model (Hollingsworth 1995). We employ the BPM life-cycle because it provides a deeper insight into the phases, activities, and—consequently—the functions that need to be supported by *BPMS-RA*. Many reference architectures have already originated from the life-cycles of their target systems (e.g., eSRA (Norta et al. 2014)). Similarly, we use the Workflow reference model because it has been used as a blueprint template in developing many BPMS architectures (e.g., SWfMS (Lin et al. 2009)). In the second step, we use component classification to categorize functions of existing BPMS architectures both from research and industry. This leads to an overview of the functions that are provided by these existing architectures. In the third and fourth step, we select and apply a set of architecture quality attributes to arrange the functions that the concrete architectures provide into a set of components that constitute the *BPMS Reference Architecture*.

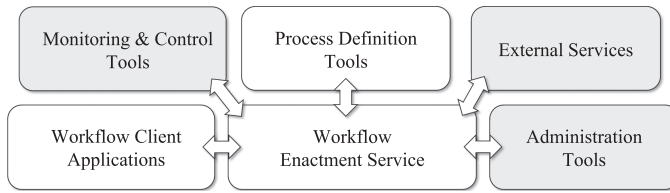


Fig. 2. BPMS component classification framework.

The remainder of this article is structured according to the research approach that is outlined in Figure 1. Section 2 presents the component classes (Step 1). Section 3 presents the classification of functions from existing architectures according to these component classes (Step 2). Section 4 discusses the selection of quality attributes that must be met in BPMS-RA (Step 3). Sections 5 and 6 discuss the design of the BPMS-RA at two levels of detail (Step 4). Section 7 considers the relevance of BPMS-RA by comparing it with three concrete architectures from practice. This section also covers the impact that the quality attributes had on the design of BPMS-RA. We present our conclusions in Section 8.

## 2 BPMS COMPONENT CLASSIFICATION FRAMEWORK

This section presents the component classification framework that we use to categorize the components from existing BPMS architectures. This framework consists of six *BPMS component classes* deduced by integrating (i) the phases distinguished in the BPM life-cycle and (ii) the components presented in the Workflow reference model.

Figure 2 shows the component classes, in which the old components from the Workflow reference model are depicted by white and the modified components are depicted by gray. First, we explain the component classes in this figure; then, we describe the rules that we used to deduce this framework.

The *Process Definition Tools* component class is used to design business process definitions in digitally processable formats that include all of the required information regarding business processes in order to realize business goals. The *Workflow Enactment Service* component class, which includes one (or in some cases multiple) so-called process engine(s), provides a runtime environment to operationalize designed process models by generating executable instances of them. The *Workflow Client Applications* component class enables the interaction of BPMSs' end users with target BPMSs. The *External Services* component class enables the interoperation of running process instances with external services. The *Administration Tools* component class provides user/role-based functions for target BPMSs. Finally, the *Monitoring & Control Tools* component class provides the ability to track and control the status of process instances during their executions.

We now describe the approach that we used to deduce the BPMS component classes. To this end, we used the mapping between the components of the Workflow reference model and the phases of the BPM life-cycle as shown in Figure 3.

In order to establish whether the Workflow reference model's components can be used as component classes in the framework, we evaluated them based on two rules.

- (1) If a component is mapped to one or more consecutive phases of the BPM life-cycle, we will consider this component as a component class in our classification.
- (2) If a component is mapped to two or more separated phases of the BPM life-cycle, we will decompose it into subcomponents until rule (1) is satisfied.

The arrows in Figure 3 illustrate the applied rules on the mapping between the components and phases. Each color represents a component class in Figure 2.

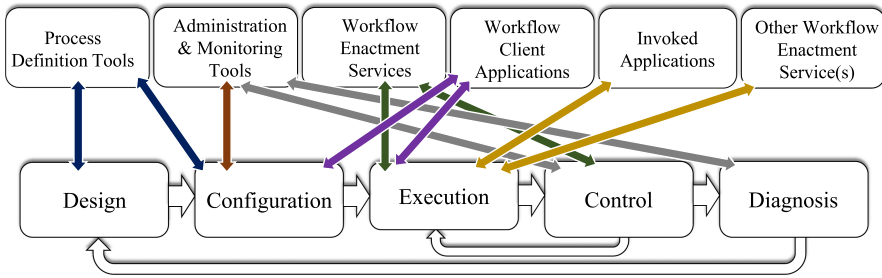


Fig. 3. Mapping between workflow reference model and BPM life-cycle.

Since two distinct phases relate to the Administration & Monitoring Tools component, we decomposed this component into two subcomponents: Administration Tools and Monitoring & Control Tools. The first subcomponent is mapped to the configuration phases while the second component is mapped to the control and diagnosis phases.

Additionally, since the other Workflow Enactment Services component and the Invoked Applications component relate to the same phase, we have merged these two components of the Workflow reference model into one component class in our framework. In the Workflow reference model, the former component allows multiple workflow systems to pass work items between one another. The latter component facilitates the invocation of all potential applications that might exist in a heterogeneous environment. However, currently, these interfaces are not usually considered separately.

### 3 BPMS COMPONENT CLASSIFICATION

This section presents the results of using the proposed classification framework from the previous section to categorize a set of existing BPMS architectures. In total, we captured 438 components from 41 primary studies in the academic literature, using a systematic literature review as elaborated in Pourmirza et al. (2017). Moreover, we selected 33 existing industry-strength BPMSs. However, since a reliable source for all existing industry-strength BPMSs is not available, we did not use a structured approach to retrieve these systems; we used a simple Google search instead. This presents a limitation because it means that the list of industry-strength BPMSs may not be complete; therefore, the functionality that is identified based on industry-strength BPMSs may not be complete.

The classification of the functionality from the existing systems is done according to a protocol in order to avoid subjectiveness as much as possible. Two of the authors read the available literature on the existing architectures (Pourmirza et al. 2017), extracted the functionality, classified that functionality, and discussed the classification in order to reach a joint classification. Subsequently, the classification was checked by the other two authors. Differences were discussed with the entire team and processed.

In the remainder of this section, we discuss the decomposition granularity and functionality that is provided by the existing architectures. We discuss the functionality per class of the classification framework from Section 2.

#### 3.1 Decomposition Granularity of Components

We discuss the levels of detail according to which the existing BPMS architectures are described, according to the concepts of *Commercial off-the-shelf (COTS) components* and *Software Suites*. COTS components refer to software modules that can be readily acquired in the market and,

Table 1. Distribution of Components Based on Classification

Component Class	Level of Details	# of Components	Component Class	Level of Details	# of Components
Workflow Enactment Service	<i>L0</i>	1	Workflow Client Applications	<i>L0</i>	0
	<i>L1</i>	44		<i>L1</i>	9
	<i>L2</i>	145		<i>L2</i>	11
	<i>L3</i>	47		<i>L3</i>	7
	<i>L4</i>	15		<i>L4</i>	0
	Total	252		Total	27
Process Definition Tools	<i>L0</i>	0	Administration Tools	<i>L0</i>	0
	<i>L1</i>	20		<i>L1</i>	5
	<i>L2</i>	51		<i>L2</i>	15
	<i>L3</i>	11		<i>L3</i>	3
	<i>L4</i>	0		<i>L4</i>	0
	Total	82		Total	23
Monitoring & Control Tools	<i>L0</i>	0	External Services	<i>L0</i>	0
	<i>L1</i>	15		<i>L1</i>	7
	<i>L2</i>	15		<i>L2</i>	10
	<i>L3</i>	3		<i>L3</i>	4
	<i>L4</i>	0		<i>L4</i>	0
	Total	33		Total	21

subsequently, can be integrated into a software system (Land et al. 2008). A set of COTS components that are bundled and that can thus be acquired in one package (Sobel et al. 2005) is usually called a software suite. Accordingly, an architecture at the *L1* level of detail presents a set of components that can be obtained as a software suite. An architecture at the *L2* level of detail presents a set of components that can be obtained as a COTS component. Subsequently, an architecture at *L3* level of detail illustrates the functional subcomponents of a BPMS and at *L4* level of detail it presents the refinement of subcomponents. Moreover, *L0* refers to a system presented as a black box. Based on these levels of detail, Table 1 provides the detailed results for the classification of the captured components from the research.

Based on this table, we argue that, by far, the greatest number of components in the selected BPMS architectures are positioned at the *L1* and *L2* levels of details. In the same way, the BPMS-RA will be mainly designed according to these two levels.

Figure 4 depicts the categorization of the functions that are deduced from the captured components on the basis of BPMS component classification. The numbers on each functionality correspond to the frequency of that functionality among the components.

In addition to the 41 BPMS-related primary studies in the literature, we have analyzed 33 industry-strength systems as presented in Table 4. However, since the detailed architectures of these systems are not easily accessible, we analyzed them only at the level of the BPMS component classes (i.e., not at the level of provided functions for each component class). Figure 5 illustrates the number of existing COTS components within the selected industry-strength systems that support the BPMS component classes. As expected, by far, the greatest number of COTS components are positioned at the Workflow Enactment Services and Process Definition Tools component classes. However, surprisingly, the External Services component class has received the lowest amount of attention in the selected systems.

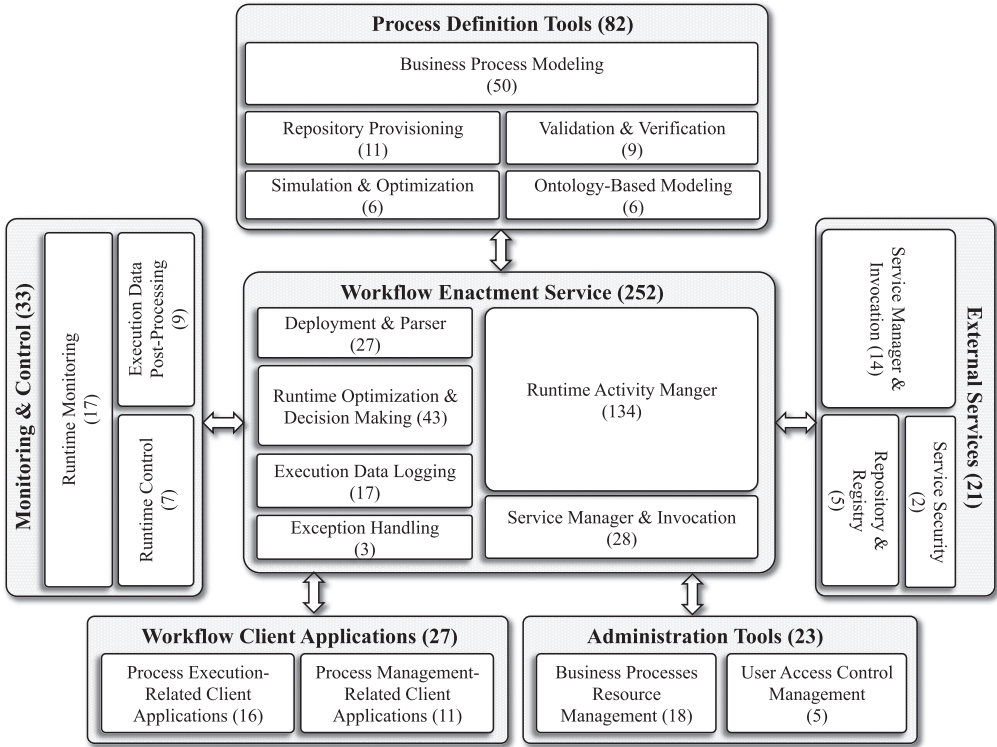


Fig. 4. Distribution of identified functions among BPMS component classes.

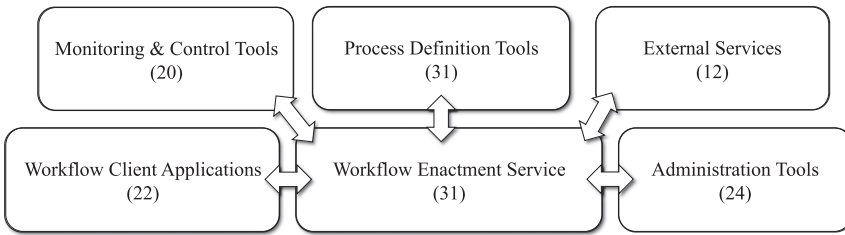


Fig. 5. Component support by industry-strength systems.

### 3.2 Functions of Process Definition Tools

We identified 82 components that are positioned in the Process Definition Tools component class. We specify five groups of functions for the Process Definition Tools component: (i) business process modeling, (ii) business process repository provisioning, (iii) business process validation and verification, (iv) business process simulation and optimization, and (v) offering ontology-based knowledge management for business processes.

The first functionality, modeling business processes, has been mentioned by 50 components in the selected architectures. A business process model contains a set of activities and a set of relationships among the activities. These relationships can be derived from a set of constraints and business rules that relate activities to each other.

The second functionality, business process repository, has been pointed out by 11 components in the selected architectures. The business process repository provisioning functionality enables the storage and retrieval of business process models.

The third functionality, validating and verifying business processes, has been suggested by 9 components in the selected architectures. These components evaluate designed business processes mainly in terms of syntax validity but also in terms of semantic validity.

The fourth functionality, simulation and optimization of business processes, has been proposed by 6 components in the selected architectures. The simulation and optimization functionality can help process designers and managers in making decisions by detecting bottlenecks and weaknesses in the designed processes before they are actually operationalized.

Finally, the role of ontologies and knowledge management in modeling business processes has received attention across the primary studies since 6 components in the selected architectures have been devoted to these issues. Ontologies are defined as a formal and shared representation model of knowledge in a specific domain (Gruber 1995). These components provide process designers and domain experts with ontological models that can be used in the modeling procedure of business processes. Using this functionality promotes common understanding and the reusability of developed business process models.

### 3.3 Functions of Workflow Enactment Services

We identified 252 components that are positioned in the Workflow Enactment Service component class. We specify six groups of functions for the Workflow Enactment Services component class: (i) business process deployment and parser, (ii) runtime activity manager, (iii) runtime optimization and decision making, (iv) logging execution data, (v) exception handling, and (vi) service manager and invocation.

The business process deployment and parser functionality, mentioned by 27 components, is the first action that has to be performed by the target component. Once a business process model is deployed to a process engine, it will be parsed by the engine. Subsequently, all activities therein will be detected by the process engine. Then, this engine can instantiate so-called process instances from the deployed process models and schedule the activities therein. Henceforth, the process engine controls the *states* of the generated processes' instances as well as the state of the inner activities.

The second functionality, runtime activity manager, has the highest frequency with 134 components among the components in the target component class. Having instantiated a process instance, the process engine is responsible for controlling the states of activity instances in addition to the process instance (Baumgrass et al. 2015c).

The third functionality, runtime optimization and decision making, has been pointed out by 43 components. This functionality provides process engines with optimization and decision-making capabilities by using business intelligence and analytics methods.

The fourth functionality, logging of execution data, has been mentioned by 17 components in the selected architectures. All information produced during the execution of process instances must be recorded by process engines.

The fifth functionality, exception handling, has been suggested by only 3 components in the selected architectures. However, we can argue that most of the BPMSs implicitly considered this functionality, but they did not explicitly depict a functional component for this issue owing to the amount of details of their target architectures.

Finally, the sixth functionality, service manager and invocation, has been recognized by 28 components in the engine components of the selected architectures. This functionality provides a



process engine with an ability to invoke external services and consume them by interpreting their response values.

### 3.4 Functions of External Services

We identified 21 components that are positioned in the External Services component class. We specify three groups of functions for this class: (i) service manager and invocation, (ii) service repository and registry, and (iii) service security and trust issue.

The first functionality, service manager and invocation, addressed by the 14 components within the External Services component class, is the same functionality as mentioned in Section 3.3. It aims at providing a service invocation environment by connecting, mediating, and managing interactions between services and BPMSs. Specifically, it enables service invocation across heterogeneous software platforms and other BPMSs. The main reason that this functionality is common across these two platforms can be explained by the fact that in many architectures there is no explicit component for handling external services (as it has been considered as part of their process engine) while in others, explicit components referring to the external services have been suggested. Consequently, this functionality has appeared in both classes.

The second functionality, service repository and registry, has been offered by 5 components within the External Services component class. This functionality is responsible for providing a catalog of available and known services, which can be also used by process designers to pick out target services in designing process models.

The third functionality, service security and trust, has been addressed by only 2 components. A BPMS can use external services only if they are trusted; this functionality provides a trust mechanism by offering features such as message encryption, signature verification, authentication, and access assessment.

### 3.5 Functions of Workflow Client Applications

We identified 27 components that are positioned in the Client Application component class. We specify two groups of functions for this component class which are: (i) process execution-related client applications, and (ii) management-related client applications.

The first functionality, process execution-related client applications, mentioned by 16 components, is used to allow end users to perform a set of activities that are available to them. The second functionality, management-related client applications, suggested by 11 components, is mainly employed by managers to gain an insight into the execution of their process, which may include a set of dashboards to show the performance of the processes. It should be noted that, in some of the primary studies, this functionality has also been considered as part of the Monitoring Tools class.

### 3.6 Functions of Administration Tools

We identified 23 components that are classified into the Administration Tools component class. We specify two groups of functions for this component class: (i) business process resource management and (ii) user access control management.

The first functionality, business process resource management, provided by 18 components, is used to specify resources that can perform activities in business processes. Note that some of the selected architectures have suggested advanced automated techniques for allocating resources to activities in a process model (e.g., Senkul and Toroslu (2005)).

The second functionality, user access control management, mentioned by 5 components, aims at validating and verifying resources before allowing them to perform specific tasks with BPM systems.

Table 2. Quality Attributes

Design-Time Quality Attributes					
Simplicity	Modifiability	Integrability	Portability	Completeness	Feasibility
Runtime Quality Attributes					
High Automation	Security	Interoperability	Usability	Performance	Availability

### 3.7 Functions Monitoring & Control Tools

We identified 33 components that are classified into the Monitoring & Control Tools component class. We distinguish three groups of functions for this class: (i) runtime monitoring, (ii) execution data post-processing, and (iii) runtime control.

The first functionality, runtime monitoring, addressed by 17 components, aims at precisely tracking and accurately recording statuses of process instances. This information is of great importance in providing insights into current statuses of running process instances.

The second functionality, execution data post-processing, identified by 9 components, provides qualitative and quantitative information about the execution of business processes. For example, it can produce valuable insights regarding the duration, costs, and quality of previously executed process instances by using some Key Performance Indicator (KPI). In more sophisticated systems, various so-called *process mining* techniques have been employed that aim at extracting knowledge from the execution data.

The final functionality, runtime control, provided by 7 components, aims to control the executions of a currently running process instance by employing various business process intelligence techniques. Unlike the previous functionality, the runtime control functionality is classified as a-priori analysis techniques. As an example, in the architecture presented in Kashlev and Lu (2014), a component, called Runtime Behavior Analytics, has been proposed that employs runtime execution data to predict and control the upcoming activities for the currently running business processes.

## 4 BPMS ARCHITECTURE QUALITY ATTRIBUTES

To design a quality reference architecture, quality attributes should be considered. We consider the classification of quality attributes that is proposed by Bass et al. (2013) because it can be considered seminal work judging by the number of citations that it has received. The quality attributes are shown in Table 2. It should be noted that in the classification by Bass et al. (2013), other quality attributes are considered subclasses of the main attributes from Table 2. For example, scalability is an important attribute but is captured under “modifying system capacity” (i.e., modifiability).

In previous work (Angelov et al. 2012), we presented a classification of different types of reference architectures and their properties. These properties determine the quality attributes that are important. In particular, we can characterize BPMS-RA as a Type III reference architecture, which is defined as “a classical reference architecture, designed by an independent organization to facilitate the design of concrete architectures of multiple other organizations” (Angelov et al. 2012).

Below, we discuss each of the quality attributes in more detail. We discuss their importance against the background of BPMS-RA as a Type III reference architecture and explain how they influenced the design approach that we used to arrive at BPMS-RA in Sections 5 and 6.

### 4.1 Design-Time Quality Attributes

*Simplicity* is the degree to which a system has a straightforward and easy-to-understand design, implementation, and deployment (IEEE 2010). *Modifiability* is the degree to which a change can be made to a system and the degree to which the system can adapt to changes (IEEE 2010).

*Integrability* is the degree to which separately developed modules and components can correctly integrate. To achieve seamless integration among the interfaces of multiple components, their interface protocols should be compatible. *Portability* is the degree to which a system can be transferred from one platform to another (IEEE 2010). This requires an architecture to be technology agnostic. *Completeness* is the degree to which a reference architecture covers the required functions for concrete architectures and *feasibility* is the degree to which a reference architecture is implementable in a timely manner.

Each of these quality attributes should be taken into account in the design of BPMS-RA. Considering that BPMS-RA is a Type III reference architecture, modifiability, integrability, portability, completeness, and feasibility are especially important. The importance of integrability, portability, and completeness follows from the property of a Type III reference architecture as an architecture for multiple organizations. As multiple organizations may be implementing different components of the architecture, it is important that these components can integrate and that they are portable. We also consider completeness important in order to facilitate *any* organization that operates in the BPMS area. The importance of feasibility follows from the Type III reference architecture property that the architecture was developed by an independent organization. It mitigates the risk that an independent organization develops an infeasible reference architecture.

The design-time quality attributes determined the design approach that we used to develop BPMS-RA in Sections 5 and 6, as follows. Simplicity and modifiability are considered by designing BPMS-RA in a modular manner on two levels of abstraction. Simplicity can be induced by applying the principle of modularity on the basis of functional separation of concerns (Fielding 2000), thus making individual components considerably less complex and, subsequently, easier to understand and implement. Similarly, the modularity principle supports modifiability of individual loosely coupled functional components. Integrability is considered by identifying interfaces at which the various components interact. While we leave the detailed definition of these interfaces abstract at this point, they can be specified in detail in future work. Portability is considered by leaving choices with respect to technology abstract. Completeness and feasibility are considered by basing BPMS-RA on existing concrete architectures, thus ensuring that it is complete with respect to these architectures and that it is feasible to implement BPMS-RA. In this manner, the design-time quality attributes lead to the design principles of modularity, abstraction, and concrete architecture mapping that are the basis for the design of BPMS-RA in Sections 5 and 6.

## 4.2 Runtime Quality Attributes

*High automation* is the degree to which functions can be automated. In the case of BPMSs, this applies to the automation of business processes (van der Aalst 2013) that facilitate the automated selection of the right tasks to perform. *System security* is the degree to which a system resists illegal usage while still providing its services to rightful users (Bass et al. 2013). *Interoperability* is the degree to which multiple information systems can exchange and use information (IEEE 2010). *Usability* is the degree to which the end users of a system acquire enough knowledge and skills to comfortably perform, to insert inputs to, and interpret outputs from the system (IEEE 2010). *Performance* is the degree to which a system accomplishes its designated functions within given constraints, such as response time, computation power, and memory usage (IEEE 2010). *Availability* is the degree to which a system is operational and accessible when required (IEEE 2010).

Runtime quality attributes are the primary concern of concrete architectures because they can be implemented and tested only in concrete systems. Therefore, consideration of runtime attributes is not part of our design approach. However, we do provide “hooks” in the architecture, where the various runtime quality attributes can be considered by concrete architectures, as we will discuss in Section 7.2.

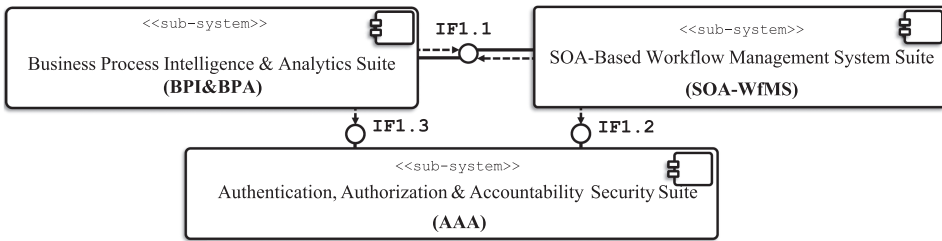


Fig. 6. BPMS-RA at the  $L1$  level.

## 5 BPMS-RA AT THE $L1$ LEVEL

We develop the BPMS-RA by modularizing the BPMS functions that were identified in Section 3. Modularization is done according to a protocol in order to avoid subjectiveness as much as possible. First, two of the authors conducted the modularization. In doing so, they respected the mapping onto the existing classification, described in Section 3, and left implementation choices abstract, thus observing the quality attributes as described in Section 4. Subsequently, the modularization was checked by a third author. Differences were discussed and the reference architecture was modified according to the discussion. A more detailed discussion on how the quality attributes influenced the modularization is given in Section 7.2.

The resulting first-level ( $L1$ ) modularization of the functionality is illustrated in Figure 6. The BPMS-RA at this level consists of three components. As  $L1$  is defined as the “tool suite” level, the modules at this level are derived from existing tool suites. In existing architectures, three types of tool suites can be recognized:

- (1) Workflow Management System (WfMS) suites (e.g., Imixs Workflow),
- (2) Service-Oriented Architecture (SOA) suites (e.g., Oracle SOA suite), and
- (3) Business Process Intelligence & Analytics (BPI&BPA) suites (e.g., SAS suite).

Since the core components of the WfMS suites and SOA suites contain common BPMS functionality, we merge these two suites. Accordingly, a component is suggested for the BPMS-RA at the  $L1$  level that is called the SOA-Based Workflow Management System Suite (*SOA-WfMS*). The commonalities between the functionality of *BPI&BPA* suites and the other two suites are limited to monitoring functionality and security functionality. For that reason, we modularize security functionality into a third component, called Authentication, Authorization & Accountability (*AAA*). The commonality with respect to monitoring functionality will be discussed and solved in the level 2 decomposition.

Interfaces are required to allow each of the three components to interact with the others. The first interface, IF1.1, facilitates data exchange between the *SOA-WfMS* and the *BPI&BPA*. The other two interfaces, IF1.2 and IF1.3, integrate the mentioned two components, respectively, with the (centralized) *AAA* component.

## 6 BPMS-RA AT THE $L2$ LEVEL

In this section, we present the BPMS-RA at the  $L2$  level of modularization. This architecture consist of a set of components that are at the same level as the refined COTS component. To define these components, we first evaluate whether we can find COTS components in the market that provide the same functions as those we derived from the captured components in Figure 4. Subsequently, we place these components into the components composing the BPMS-RA at the  $L1$  level.

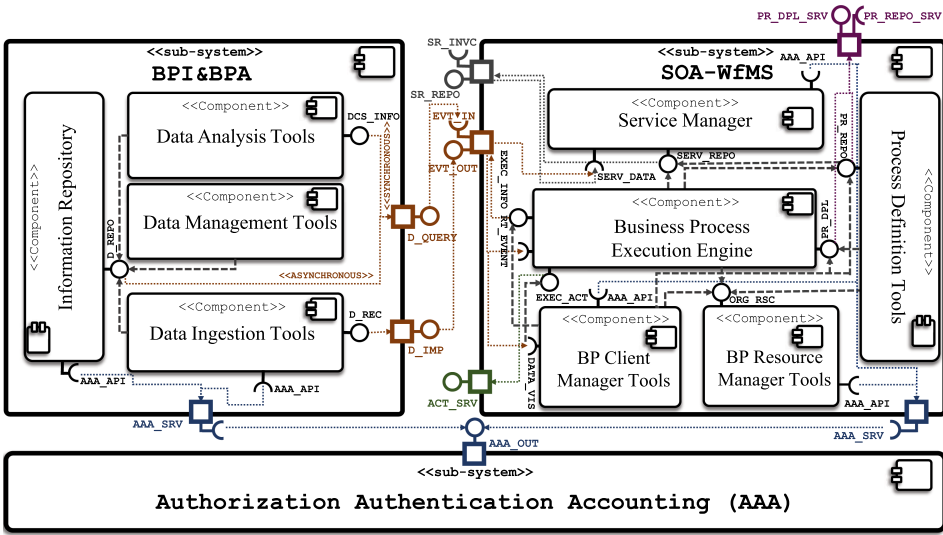


Fig. 7. BPMS-RA at the L2 level.

Note that we zoom into only the two main components of BPMS-RA at the L1 level, the SOA-WfMS component and BPI&BPA component, because the architectures that we studied did not provide a more detailed decomposition of their security components and functionality. Figure 7 illustrates the BPMS-RA at the L2 level.

The SOA-WfMS component is composed of five inner components: *Process Definition Tools*, *Business Process Execution Engine*, *Service Manager*, *BP Client Manager Tools* and *BP Resource Manager Tools*. The *Process Definition Tools* component provides a business process modeling environment, a model repository, a model validation and verification, and a simulation and optimization functions. The *Business Process Execution Engine* component provides a process deployment and parser, a runtime activity manager environment, a logging function, and an exception handling function. The *Service Manager* component provides a service manager and invocation and a service repository. The *BP Client Manager Tools* component supports process execution-related and management-related client applications. Finally, the *BP Resource Manager Tools* provides a process resource management functionality.

The BPI&BPA component comprises four inner components: *Data Ingestion Tools*, *Information Repository*, *Data Management Tools*, and *Data Analysis Tools*. The *Data Ingestion Tools* component facilitates the extraction of data from various heterogeneous data sources, transforms them into comprehensible formats, and loads them into another component of the BPI&BPA, the *Information Repository* component in the architecture. The *Data Management Tools* component enables the handling of imported information (e.g., by enriching and correlating the information). Last, the *Data Analysis* component provides both post-processing and runtime information processing functions that can be used to improve and control BPMS-RA-compliant systems by provisioning the decision-making capability.

In the rest of this section, we further explain the BPMS-RA at the L2 level.

### 6.1 Inner Components of the SOA-WfMS

**6.1.1 Derived Component from Process Definition Class.** There are some COTS components in the market that provide the required functions for the *Process Definition Tools* class. For example,

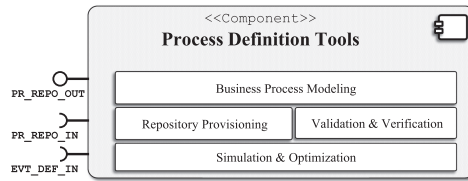


Fig. 8. Functions of process definition tools.

Signavio Process Editor is a stand-alone Web-based modeling environment for designing BPMN process models. Consequently, we suggest a component, called *Process Definition Tools*, for the BPMS-RA at the *L2* level. This component, as shown in Figure 8, must ideally support the first four identified functions. Moreover, the PR\_REPO\_OUT interface and PR\_REPO\_IN interfaces are defined for uploading and downloading business process models from the repository and the EVT\_DEF\_IN interface can be used mainly by process designers to model business processes according to the external events that may be received.

A BPMS-RA-compliant system must provide business process modeling functionality. To this end, it allows end users of the system to design business processes that include multiple types of activities and relationships among them.

A full-fledged Process Definition Tools component must contain its own business process repository. A well-known example of such a repository is the SAP reference model, which includes over 600 process models. Regarding academic initiatives, an advanced business process model repository, called APROMORE, has been proposed (La Rosa et al. 2011).

A BPMS-RA-compliant system must support validation and verification functionality. For example, ADONIS provides a feature for validating the syntactical correctness of business processes. Additionally, many academic initiatives, such as WoPed (Freytag 2005), offer syntactical and semantical validation of process models.

A complete Process Definition Tools component must include business process simulation and optimization functionality. Considering industrial solutions, many process modelers such as Bizagi Process Modeler and Tibco have a feature that supports process simulation. Considering academic initiatives, a notable example is CPN Tools (Jensen et al. 2007) for simulating and analyzing Petri Nets.

Although ontology-based modeling functionality has been deduced from academic studies, it has very limited support among industrial solutions. This functionality aims at providing learned knowledge for the process designer; therefore, it is shifted to the BPI&BPA component.

**6.1.2 Derived Component from Workflow Enactment Services Class.** All WfMS and SOA suites include COTS components that provide the main functions for the Workflow Enactment Services component class. For example, Camunda and Activiti contain the Camunda Process Engine and Activiti Process Engine, which are responsible for executing business process models. Consequently, we suggest a component, called *Business Process Execution Engine*, as shown in Figure 9, for the BPMS-RA at the *L2* level. The Business Process Execution Engine interacts with the Process Definition Component via the PR\_DPL\_OUT and PR\_DPL\_IN interfaces. A business process model can be deployed to the execution engine automatically using the PR\_REPO\_IN interface or manually using the PR\_REPO\_OUT interface. Also, the BP Execution Engine interacts with the BPI&BPA component by receiving runtime events from (via the RT\_EVT\_IN interface) and providing execution data to this component (via the EXEC\_INFO\_OUT interface). Finally, the business process execution engine interacts with the service manager and invocation functionality (shown in the detailed view of

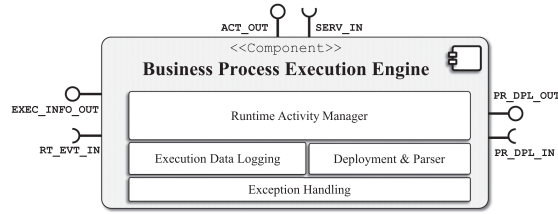


Fig. 9. Functions of the business process execution engine.



Fig. 10. Functions of the service manager.

Figure 10) concerning runtime data exchange (via the SERV\_IN interface) and with any external services (e.g., to perform an activity) with the target engine (via the ACT\_OUT interface).

A BPMS-RA-compliant system provides business process deployment and parser functionality that produces process instances from the deployed process model. A process instance can be at different states (i.e., initialized, running, terminated, completed, and suspended); therefore, the Business Process Execution Engine must support the transition among these states.

Having instantiated a process instance, the process engine must support runtime activity manager functionality, which can be seen as the core feature of a BPMS-RA-compliant system. This functionality enables a BPMS-RA-compliant system to categorize the activities in a process instance with different states (i.e., scheduled, (re-)assigned, accepted, rejected, skipped, enabled, running, and completed) and it enables the transitions among these states.

A full-fledged BPMS-RA-compliant system must be able to support logging execution data functionality by providing an interface in which execution logs can be exchanged. Therefore, the BPI&BPA component can exploit the execution data for provisioning more accurate information.

A BPMS-RA-compliant system must support exception handling functionality by identifying potential exceptions (using execution monitoring) and, subsequently, by exception recovery (using some techniques that are bundled in the BPI&BPA component). This functionality has been considered as an *explicit* functionality in BPM systems. However, according to the usability runtime quality attribute, it seems to be logical to highlight exception handling functionality in designing the BPMS-RA architecture.

The other two functionalities of the Workflow Enactment Services component class will be positioned at other components in BPMS-RA. Considering the architecture of the BPMS-RA at the LI, we shift runtime optimization and decision making to the BPI&BPA component. The reason for this shift is coupling all of the functions that use business intelligence and analytics techniques so that they can be carried out by the BPI&BPA component. Having shifted this functionality, we promote the principle of separation of concerns, thus, further boosting the modularity principle in our design. Consequently, the IF1.1 interface between the SOA-WfMS component and the BPI&BPA component, as shown in Figure 6, must provide the business process execution engine component with runtime information provisioned by the runtime optimization and decision-making functionality. Also, we merge the service manager and invocation functionality in this component class with the same functionality in the External Services component class. The main reason for

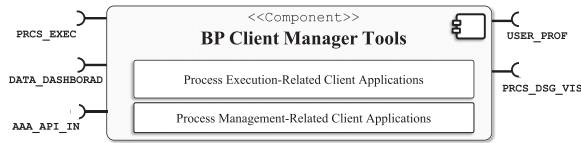


Fig. 11. Functions of client applications.

this design decision is the principle of separation of concerns, as the main concern for the process engines is to interpret the business process models and provide a runtime environment to operationalize them while the major concern of the captured components in the external services is to employ external services to feed required input data into the process engines.

**6.1.3 Derived Components from External Services Class.** Many WfMS and almost all SOA suites in the market include COTS components that provide the required functions for the External Services component class (e.g., Oracle SOA Suite). Accordingly, we suggest a component, called *Service Manager*, for the BPMS-RA at the *L2* level. This component, as shown in Figure 9, provides a service invocation environment for the consumer of this functionality (via the SERC\_DATA\_INC interface). Moreover, the Service Manager component ideally contains a repository and/or a registry for services that can be reachable via SRV\_REPO\_OUT and SRV\_REPO\_IN interfaces. Finally, in collaboration with the AAA component, the Service Manager component can certify the validity of services through the AAA\_API\_IN interface.

As discussed before, a BPMS-RA-compliant system must support the service manager and invocation functionality through the Service Manager component. For example, Hu and Grefen (2003) have proposed an architecture for the service mediating workflow management systems. Also, a notable example of a COTS component is the Oracle SOA Suite, which includes the Oracle Service Bus.

An ideal BPMS-RA-compliant service must include the service repository and registry functionalities. An example of an available COTS component that provides such a functionality is the Anypoint Service Registry that has been offered by MuleSoft. Another well-known example of this functionality is the WebSphere Service Registry and Repository from IBM.

Although the service security has been deduced from the selected architecture, considering the separation of concern, we shift it to the AAA component in the BPMS-RA at the *L1* level.

**6.1.4 Derived Component from Workflow Client Applications Class.** Almost all WfMS suites in the market include COTS components that support the functions for the Client Applications component class. For example, the Activiti BPM Platform has a dedicated component, called Activiti Explorer, providing end users with a Web-based interface. Also, Microsoft Outlook has been employed as a client application for many BPMs, such as the Together Workflow Server. Accordingly, we suggest a component, called *Business Process Client Manager Tools*, for the BPMS-RA at the *L2* level. This component, as shown in Figure 11, must support the process execution-related client applications and the process management-related client applications functions.

For this purpose, the PRCS\_EXEC and DATA\_DASHBOARD interfaces are designed to consume data from and visualize required features for process execution-related functionality (e.g., activity list) and management-related functionality (e.g., performance dashboard), respectively. However, an ideal client manager tool includes a set of pluggable applications, such as business processes modeler, that can be realized via the PRCS\_DSG\_VIS interface. Another example for these interfaces is an application for managing users' profiles that can be implemented through the USER\_PROF interface. End users can ideally customize their own client application based on their preferred tools according to their authorization level. Nevertheless, end users need to be granted access from the



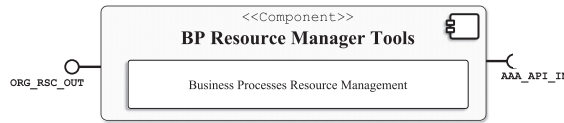


Fig. 12. Functions of the resource application.

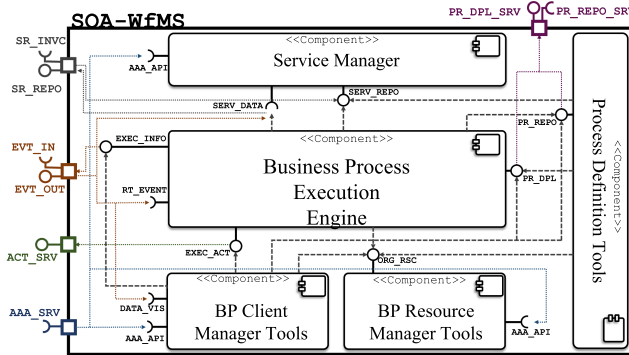


Fig. 13. The SOA-WfMS component at the L2 level.

AAA component; thus, the IF1.2 interface between the SOA-WfMS and the AAA component needs to be implemented in a way that the BP Client Manager Tools’ end users are authenticated and authorized via the AAA component. This functionality can be realized through the AAA\_API\_IN interface.

6.1.5 *Derived Component from Administration Tools Class.* A great number of WfMS and a lesser number of SOA suites include COTS components that support the required functions for the Administration Tools component class (e.g., Oracle Role Manager and Metastorm BPM User Management). Accordingly, we suggest a component, called *Business Process Resource Manager Tools*, for the BPMS-RA at the L2 level. We have used the term *resources* as it has been predominantly used by the BPM community. This component, as shown in Figure 12, manages all of the potential resources that can be employed to perform business processes by providing an interface (i.e., ORG\_RSC\_OUT) to the other components of a BPMS-RA-compliant system.

Note that, owing to the separation of the concern design principle, the user access control management functionality is handled by the AAA component; thus, the AAA\_API\_IN interface is foreseen to enable this interaction.

6.1.6 *SOA-WfMS Component at the L2 Level.* Altogether, these suggested components result in the SOA-WfMS component at the L2 level, as illustrated in Figure 13.

We employ black arrows if a component provides/uses an internal interface and colorful arrows if a component provides/uses an external interface through the designed ports, which are distinguished according to their concerns.

The concern of the purple port is business process models. This port contains a set of CRUD functions for an external process definition tool and a deployment functionality to directly deploy a business process model into the Execution Engine. The concern of the dark-gray port is services. This port enables external service managers to interact with the internal service repository of the target BPMS. The concern of the green port is execution-related activities. This port provides a set of functions to enable users to interact with running process instances. The concern of the orange port is events. This port establishes an interaction between the SOA-WfMS component

and the BPI&BPA component. Finally, the concern of the navy-blue port is security. This port consumes a set of authentication, authorization, and accounting services that are provided by the AAA component.

## 6.2 Inner Components of the BPI&BPA

Considering the BPI&BPA, we suggest that inner components stem from the Monitoring & Control component class to provide three main functions: (i) runtime monitoring, (ii) execution data post-processing, and (iii) runtime control.

Runtime monitoring functionality observes the execution of business process instances by producing a set of events (i.e., execution data). There are some COTS components in the market for monitoring these events, such as Oracle Business Activity Monitoring (BAM) and IBM Business Monitor.

The recorded execution data can be further exploited to provide organizations with better understanding as to how their process instances are actually executed. These kinds of techniques align well with the execution data post-processing functionality for which we found a great number of COTS components, such as ProM and Disco.

Runtime control functionality controls the behavior of currently running process instances by enabling decision making. Having exploited the past and current execution data, this functionality forecasts and further predicts the future events that may influence running instances. This functionality has been supported by many COTS components (e.g., SAS Business Intelligence & Analytics and IBM Cognos Analytics).

Accordingly, we design the inner architecture of the BPI&BPA component based on the introduced COTS components. The first common action among these COTS components is the ETL (Vassiliadis 2009) procedure, which extracts data from various heterogeneous data sources, transforms them into comprehensible formats, and loads them into an information repository. The BPI&BPA component must enable the import of all types of data (i.e., batches or streams). Since the ETL techniques are mainly designed to support the batch mode, a new method has been suggested, called *Data Ingestion* (Grover and Carey 2015), covering both types offered by many COTS components, such as Apache Chukwa and Gobblin. Consequently, the BPI&BPA component contains two components: (1) a *Data Ingestion Tools* component that imports and formats both batches and streams of data and (2) an *Information Repository* component that provides storage for the formatted data.

Additionally, the BPI&BPA must be able to manage imported data in such a way that when new data come into the information repository they must provide some added value to the already existing information. Many data management challenges are proposed in the literature. For example, the data enrichment challenge in Chaudhuri (2012) and the data retention challenge in Tene and Polonetsky (2012) have been suggested. There are some COTS components resolving these challenges, such as SAS Data Management Software, which provides the data enrichment functionality. Therefore, the BPI&BPA component must include a *Data Management Tools* component, which resolves data management challenges.

Finally, the imported data must be analyzed. The input for post-processing analysis techniques are often batches of execution data and for runtime control techniques can be both batches and streams of data. Therefore, the BPI&BPA component includes a *Data Analysis Tools* component supporting both sets of batch data analysis and stream data analysis techniques. Considering batch data analysis, one of the well-known methods is OLAP, for which many systems are available as COTS component (e.g., SAS OLAP Server and Oracle OLAP). Another example for supporting batch processing is Apache Hadoop, which provides a distributed computing framework to process large amounts of data in parallel. Considering stream data analysis, a well-known method is

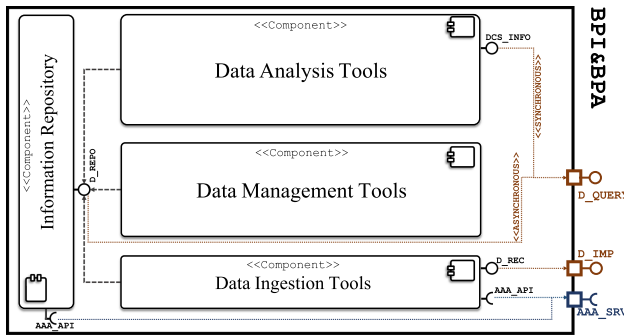


Fig. 14. The BPI&BPA component at the *L2* level.

Complex Event Processing (CEP), provided by existing COTS components, such as jBoss Drools Fusion, Oracle Complex Event Processing, and Esper.

Altogether, these suggested components constitute the BPI&BPA component at the *L2* level as seen in Figure 14, which illustrates the detailed view of this component.

This architecture contains three internal interfaces and three ports. The *D\_REC* interface, provided by the Data Ingestion Tools, receives data from various data sources. The *D\_IMP* port is connected to this interface, which can be linked to the *EVT\_OUT* interface to establish an interaction between the SOA-WfMS and the BPI&BPA component for transferring the execution data. The *D\_REPO* interface, provided by the Information Repository, offers a set of required CRUD functions used by the other components. The *D\_QUERY* port, connected to the *D\_REPO* interface, can be linked to the *EVT\_IN* interface to also establish an interaction between the SOA-WfMS and the BPI&BPA component for asynchronously transferring post-processed improvements. The *DCS\_REP* interface, provided by the Data Analysis Tools, is also connected to the *D\_QUERY*. However, the *DCS\_REP* interface synchronously invokes consumers using different mechanisms, such as triggers. Finally, the *AAA\_SRV* port is responsible for consuming a set of authentication, authorization, and accounting services that are provided by the AAA component.

## 7 EVALUATION AND DISCUSSION

This section provides a discussion on the architecture that is described in the previous two sections along two lines. First, it compares the reference architecture to three concrete BPM system architectures that are applied in practice. Second, it discusses how the quality attributes that were discussed in Section 4 have impacted the reference architecture.

### 7.1 Relation to Concrete Architectures

To evaluate the practical relevance of BPMS-RA, we compared its components to the components of three concrete BPMS architectures that are used in practice. Table 3 shows the results of this effort.

All of the BPMSs provide design, execution, and monitoring capabilities for BPMN 2.0 business processes. As shown in the table, the Process Definition Tools component of BPMS-RA is realized in Activiti by two components: Activiti Modeler, which is a web-based business process development environment; and Activiti Designer, which provides an Eclipse-based plug-in with the same purpose. Bizagi and GET Service support this component with the Bizagi Process Modeler and the Process Development Environment, respectively. The mapping of concrete components to the BP Execution Engine, BP Client Manager, and BP Resource Manager is straightforward. Regarding

Table 3. Mapping Concrete Architectures to BPMS-RA

BPMS-RA ( <i>L1</i> )	BPMS-RA ( <i>L2</i> )	Alfresco Activiti	Bizagi	GET Service
SOA-WfMS	Process Definition Tools	Activiti Modeler/Designer	Bizagi Process Modeler	Process Development Environment
	BP Execution Engine	Activiti Engine	Bizagi Engine	Orchestration Engine
	BP Client Manager Tools	Activiti Explorer	Bizagi Work Portal	Process Client
	BP Resource Manager Tools	Activiti Admin	Workload Management	Planner
	Service Manager	Activiti REST	Application Integration	Backend System/Service Registry
BPI&BPA	Data Analysis Tools	Activity Analytics App	Reports and Process Analytics	Event Correlator/Aggregator
	Data Management Tools	Data Model Component	EntityManager	Event Management
	Data Ingestion Tools	Alfresco ETL Connector	Out-of-the-box connectors	Event Channel/Subscription Store
	Information Repository	Activiti Datasource (ADS)	Operational Data Store (ODS)	Information Store
AAA	AAA	Identity Management	Work Portal Security	Community Passport Manager

the Service Manager, the three systems provide RESTful service integration capabilities that allow both internal and external services to be stored, managed, and exploited.

While all the components of the SOA-WfMS part of BPMS-RA are fully supported by the three systems, there is far less support for the BPI&BPA components. The Data Analysis component is supported by the event correlator and event aggregator components of GET Service. However, this component received less attention in Activiti and Bizagi. Both systems have an application for producing reports and for analyzing past executions of the business processes. However, they do not support the prediction of future executions based on past events. The Data Ingestion Tools component is supported by GET Service and Activiti, and in the new release of Bizagi there are some out-of-the box components to ingest data from specific systems such as SAP. The functions that are provided by the Data Management Tools component and the Information Repository component are linked with the functions that are provided by the relevant components as shown in Table 3. Consequently, it is possible to claim that these two components are supported by all three systems.

The functions of the AAA component are also covered by all three systems. Activiti by default supports basic authentication, while Bizagi provides different options for authentication, including default Windows authentication, Active Directory authentication, and federated authentication. GET Service provides an OAuth 2.0 authentication procedure.

## 7.2 Impact of Quality Attributes

The quality attributes have impacted the reference architecture as follows.

- Simplicity has been considered by creating a two-level decomposition of the architecture into a relatively small number of components that have clearly identifiable functionality.
- Modifiability has been considered by creating loosely coupled components that can be modified relatively independently of each other.
- Integrability has been considered by identifying the interfaces at which the components must interact. It can be further considered in future work by standardizing the interaction at these interfaces.
- Portability has been considered by abstracting from implementation choices. It can be further considered in future work by considering the way in which the various components must be deployed.
- Completeness has been considered by using a systematic literature review that aims to identify all existing BPMSs (Pourmirza et al. 2017).
- Feasibility has been achieved by considering BPMSs that have been implemented, showing that the reference architecture can indeed be implemented as discussed in detail in Section 7.1.
- High-automation has been considered in the communication between the BP Execution Engine (from SOA-WfMS) and Data Analysis Tools (from BPI&BPA). The latter component exploits the historical execution logs from the former component (through the EVT\_OUT interface) and feeds the former component with the predicted future execution paths automatically (through the EVT\_IN interface).
- Security has been considered by introducing a dedicated component that must ensure security.
- Interoperability is achieved by introducing dedicated components that are responsible for interoperability, in particular, the Service Manager component through its SR\_INVC interface and the Data Ingestion Tools through the D\_IMP interface.
- Usability quality must be considered in future work in the BP Client Manager Tools and Process Definition Tools, as these two components are used by the end user of a BPMS-RA compliant system. It must also be considered in the BP Execution Engine and, in particular, through the Exception Handling functionality, since the BPMS-RA must support exception detection and recovery.

In the design of BPMS-RA, we considered a trade-off between the following quality attributes.

- Simplicity versus modifiability: Although the former facilitates BPMS-RA-compliant systems to be realized and deployed easily, the latter prevents the systems getting stuck forever with the original deployment. Therefore, we paid more attention to modifiability than to simplicity.
- Simplicity versus completeness: A more complete system may result in a more complex one. We tried to mitigate this issue by modularizing the functionality, which we consider complete in light of the literature study that we conducted.
- Simplicity versus high automation and interoperability: A more automated and interoperable system may result in a more complex one. We tried to mitigate this issue by proposing the definition of standard interfaces.
- Feasibility versus modifiability, integrability, and portability: Although developing a more modifiable, integrable, and portable system requires more implementation efforts during the first development cycle, we believe that these quality attributes will eventually result in less implementation effort during the life-cycle of a BPMS-RA-compliant system. Therefore, we paid more attention to these quality attributes than to feasibility.

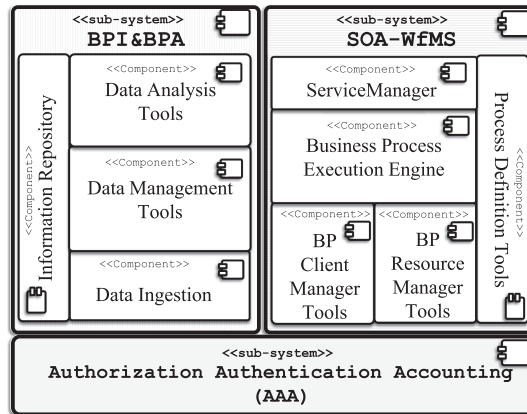


Fig. 15. Overview of BPMS-RA.

- Security versus integrability and interoperability: The security of a system may degrade as a system is integrated with a new components. In light of this trade-off, we did not consider security beyond the introduction of an AAA component that primarily implements authorization and authentication.
- Modularity versus performance: A less modular decomposition may result in a better performing system since all of the required resources are accessible with less latency. However, BPMS-RA has been designed primarily based on the principle of modular decomposition.

## 8 CONCLUSION

This article presents a reference architecture for Business Process Management Systems, called BPMS-RA. BPMS-RA provides a guideline for the development of concrete BPMSs. In addition, it offers a common understanding of the provided functionalities and interfaces of these systems. Finally, it can be employed as a standard for evaluating the completeness of existing BPMS architectures and systems.

BPMS-RA was developed based on concrete BPMSs from the research community (using a research-driven approach) and on concrete BPMSs from industry (using a practice-driven approach).

Figure 15 presents a condensed view of BPMS-RA. Shared borders between components in this figure represent interfaces between these components. At the highest level of abstraction, BPMS-RA consists of three main components: (1) the SOA-WfMS component, which offers functions such as business process modeling and execution; (2) the BPI&BPA component, which is responsible for monitoring and controlling BPMS-RA-compliant systems; and (3) the AAA component, which secures BPMSs by providing functions for authentication, authorization, and accounting. At lower levels of abstraction, BPMS-RA is refined into components and lists the functionality that is provided by these components.

By providing a reference architecture that is based on recent concrete architectures, the contribution of BPMS-RA is an update, after twenty years, of existing reference architectures (Hollingsworth 1995; Grefen and Remmerts de Vries 1998) with the latest developments from both research and practice. Thus, BPMS-RA takes functionality into account that is provided by traditional workflow systems as well as functionality provided by modern-day business process intelligence systems. When comparing the existing reference architectures to BPMS-RA, the most striking improvements of BPMS-RA are the integration of components for real-time business

process analysis and the shift to a service-oriented paradigm, which is related to the end of a distinction between client applications and other workflow enactment services.

BPMS-RA is meant to facilitate the design of concrete architectures by researchers and practitioners. As a facilitation reference architecture, it provides guidelines and inspiration for the design of concrete architectures (Angelov et al. 2012). In particular, it gives a complete overview of the functionality that is provided by modern-day BPMSs. This overview can serve as an inspiration for the functions that a researcher or practitioner may want to implement. The reference architecture also suggests a system structure that organizes these functions, including the interfaces that facilitate the interaction between the functions. This structure can serve as a guideline for concrete BPMSs. It must be noted that the interfaces themselves are not standardized in this article. In that respect, BPMS-RA is a facilitation rather than a standardization architecture (Angelov et al. 2012), which it has in common with the existing reference architectures (Hollingsworth 1995; Grefen and Remmerts de Vries 1998) on which it is based. However, the mere identification of interfaces is a guideline in itself and shows where standardization efforts are necessary. Indeed, the Workflow Reference Model also had that goal (Hollingsworth 1995) and has (indirectly) inspired standards such as the BPMN Interchange Format (Object Management Group 2011), which partly standardizes the exchange of process models between a process definition tool and an execution engine. Along the same lines, BPMS-RA is related to the XES standard (IEEE 2016), which partly standardizes the data exchange between an execution engine and data analysis tools.

## REFERENCES

- Samuil Angelov, Paul Grefen, and Danny Greefhorst. 2012. A framework for analysis and design of software reference architectures. *Information and Software Technology* 54, 4 (2012), 417–431.
- Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, and Judith Stafford. 2011. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley.
- Len Bass, Paul Clements, and Rick Kazman. 2013. *Software Architecture in Practice* (3rd ed.). Vol. 2. Addison-Wesley.
- Anne Baumgrass, Mirela Botezatu, Claudio Di Ciccio, Remco Dijkman, Paul Grefen, Marcin Hewelt, Jan Mendling, Andreas Meyer, Shaya Pourmirza, and Hagen Völzer. 2015a. Towards a methodology for the engineering of event-driven process applications. In *BPM Workshop*, Vol. 256. Springer, 501–514.
- Anne Baumgrass, Claudio Di Ciccio, Remco Dijkman, Marcin Hewelt, Jan Mendling, Andreas Meyer, Shaya Pourmirza, Mathias Weske, and Tsun Yin Wong. 2015b. GET controller and UNICORN: Event-driven process execution and monitoring in logistics. In *BPM Demo*, Vol. 1418. CEUR Proceedings, 75–79.
- Anne Baumgrass, Remco Dijkman, Paul Grefen, Shaya Pourmirza, Hagen Völzer, and Mathias Weske. 2015c. A software architecture for transportation planning and monitoring in a collaborative network. In *IFIP Advances in Information and Communication Technology – Risks and Resilience of Collaborative Networks*. Vol. 463. Springer, 277–284.
- Surajit Chaudhuri. 2012. What next? A half-dozen data management research goals for big data and the cloud. In *Symposium on Principles of Database Systems*. ACM, 1–4.
- Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo Reijers. 2013. *Fundamentals of Business Process Management*. Springer.
- Roy Thomas Fielding. 2000. *Architectural styles and the design of network-based software architectures*. Ph.D. Dissertation. University of California, Irvine, Irvine, CA.
- Thomas Freytag. 2005. WoPeD–workflow Petri net designer. In *Applications and Theory of Petri Nets*. <https://www.researchgate.net/publication/228697436>.
- GET Service Consortium. 2013. GET Service: Efficient Transportation Planning and Execution. Retrieved January 12, 2018 from <http://getservice-project.eu/>.
- Paul Grefen. 2015. *Business Information System Architecture (BISA)*. Eindhoven University of Technology.
- Paul Grefen and Remmert Remmerts de Vries. 1998. A reference architecture for workflow management systems. *Data & Knowledge Engineering* 27, 1 (1998), 31–57.
- Raman Grover and Michael J. Carey. 2015. Data ingestion in AsterixDB. In *EDBT*. Openproceedings.org, 605–616.
- Thomas R. Gruber. 1995. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies* 43, 5 (1995), 907–928.
- Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. 2004. Design science in information systems research. *MIS Quarterly* 28, 1 (2004), 75–105.

- David Hollingsworth. 1995. *The workflow reference model*. Workflow Management Coalition Technical Report TC00-1003 (1995).
- Jinmin Hu and Paul Grefen. 2003. Conceptual framework and architecture for service mediating workflow management. *Information and Software Technology* 45, 13 (2003), 929–939.
- IEEE. 2010. Systems and software engineering – vocabulary. *IEEE Standard*, 1–418.
- IEEE. 2016. *Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams*. Technical Report 1849-2016. IEEE.
- Kurt Jensen, Lars Michael Kristensen, and Lisa Wells. 2007. Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer* 9, 3–4 (2007), 213–254.
- Andrey Kashlev and Shiyong Lu. 2014. A system architecture for running big data workflows in the cloud. In *IEEE International Conference on Services Computing (SCC)*. IEEE, 51–58.
- Marcello La Rosa, Hajo Reijers, Wil van der Aalst, Remco Dijkman, Jan Mendling, Marlon Dumas, and Luciano García-Bañuelos. 2011. APROMORE: An advanced process model repository. *Expert Systems with Applications* 38, 6 (2011), 7029–7040.
- Rikard Land, Laurens Blankers, Michel Chaudron, and Ivica Crnković. 2008. COTS selection best practices in literature and in industry. In *High Confidence Software Reuse in Large Systems*. Springer, 100–111.
- Cui Lin, Shiyong Lu, Xubo Fei, Artem Chebotko, Darshan Pai, Zhaoqiang Lai, Farshad Fotouhi, and Jing Hua. 2009. A reference architecture for scientific workflow management systems and the VIEW SOA solution. *IEEE Transactions on Services Computing* 2, 1 (2009), 79–92.
- Alex Norta, Paul Grefen, and Nanjangud C. Narendra. 2014. A reference architecture for managing dynamic inter-organizational business processes. *Data & Knowledge Engineering* 91 (2014), 52–89.
- Object Management Group. 2011. *Business Process Model and Notation (BPMN) - version 2.0*. Technical Report formal/2011-01-03. Object Management Group.
- Shaya Pourmirza, Sander Peters, Remco Dijkman, and Paul Grefen. 2017. A systematic literature review on the architecture of business process management systems. *Information Systems* 66 (2017), 43–58.
- Pinar Senkul and Ismail H. Toroslu. 2005. An architecture for workflow scheduling under resource allocation constraints. *Information Systems* 30, 5 (July 2005), 399–422.
- William Sobel, Bruce McCorkendale, and Thomas Powledge. 2005. Systems and methods for centralized subscription and license management in a small networking environment. Patent No. 11/192038, Filed July 29th., 2005, Issued February 1st., 2007.
- Omer Tene and Jules Polonetsky. 2012. Big data for all: Privacy and user control in the age of analytics. *Northwestern Journal of Technology and Intellectual Property* 11 (2012), xxvii.
- Wil van der Aalst. 2013. Business process management: A comprehensive survey. *ISRN Software Engineering* 2013, Article 507984 (2013), 37 pages.
- Wil van der Aalst, Mariska Netjes, and Hajo Reijers. 2007. Supporting the full BPM life-cycle using process mining and intelligent Redesign. *Contemporary Issues in Database Design and Information Systems Development*, Keng Siau (Ed.). IGI Global, 100–132.
- Panos Vassiliadis. 2009. A survey of extract–transform–load technology. *International Journal of Data Warehousing and Mining* 5, 3 (2009), 1–27.

Received November 2016; revised June 2018; accepted June 2018