The BWS Open Business Enterprise System Architecture

Cristian IONIȚĂ
Academy of Economic Studies, Bucharest, Romania
crionita@ie.ase.ro

Business process management systems play a central role in supporting the business operations of medium and large organizations. This paper analyses the properties current business enterprise systems and proposes a new application type called Open Business Enterprise System. A new open system architecture called Business Workflow System is proposed. This architecture combines the instruments for flexible data management, business process management and integration into a flexible system able to manage modern business operations. The architecture was validated by implementing it into the DocuMentor platform used by major companies in Romania and US. These implementations offered the necessary data to create and refine an enterprise integration methodology called DM-CPI. The final section of the paper presents the concepts, stages and techniques employed by the methodology.
Keywords: *BWL, Workflow, BWS, Evaluation, Open Business Enterprise System, DM-CPI*

1 Introduction

All modern businesses depend on complex business processes in order to conduct their daily activities. These processes involve documents, people and internal or external information systems. Traditional economic information systems are task based. They support the user in performing specific tasks, but they fail to integrate all the aspects involved in a typical business process. In order to do implement the support for business process automation into an enterprise system we need to transform it into a process aware system. A process aware information system is a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models ([4]). These models are typically instantiated multiple times and every instance is handled in a predefined way (possibly with variations).

The present paper proposes a new kind of business information system that extends the process management facilities of existing process aware information systems with the features necessary for the end-to-end management of business operations: flexible data management, extensible process management environment and enterprise integration support. Section 2 examines the characteristics and shortcomings of current economic information systems and proposes a new kind of system named Open Business Enterprise System – OBES. The paper proposes an OBES architecture called Business Workflow System – BWS that permits the implementation of such system. Section 4 presents how the architecture was implemented into the DocuMentor platform and the methodology created for managing business integration processes based on the platform.

2 Open Business Enterprise Systems

Modern economic information systems - EIS are large-scale distributed systems aimed at automating business processes within organizations. The features that characterize these systems are the size, purpose and importance to the organization.

Large *size* is a defining feature of modern EIS. This is apparent both in terms of volume of data processed and the number of concurrent users and the size and complexity of component modules. EIS modules contain sizable business logic code and use multiple processing nodes interconnected by communication networks. Because the development and implementation of such systems requires a large effort from the organization, economic systems are used for long periods and must be interconnected with other applications that participate in the economic processes.

The *purpose* of economic systems is to meet specific business needs. Therefore, economic

systems encode processes, rules and manipulate data entities within the organization. The ultimate objective of these systems is to increase profits by streamlining the organizations business processes.

The high *importance* of EIS comes from the fact that business processes supported by these systems have become critical for the success of the organization in the context of increased computerization. Therefore systems must be robust enough to allow continuous operation and be able to handle the load when the volume of processed data increases. Continuous operation requires the existence of effective mechanisms for monitoring and management. To meet business growth, economic systems must be scalable, allowing expansion of processing capacity without requiring a system redesign.

All these characteristics make the development of economic systems very complex and risky task. The shortcomings of traditional EIS are:

- **high cost and rate of failure of EIS development projects;**

According to [5], the U.S. companies' annual budget for the development of such systems is about $ 250 billion. The average cost for a project ranges from 2300000 USD to 430000 USD based on company size and complexity. Only 16% of these projects were completed on time and were within budget. Another 31% were canceled due to quality problems and generated losses of about 81 billion USD. From the analyzed systems, 53% of projects exceeded budget by an average of 189% resulting in losses of 59 billion USD. Completed projects implemented only about 42% of the originally planned functionality. The main factors that contributed to these results are unnecessary complexity and the failure to implement a continuous updating strategy.

- *unnecessary complexity from the use of inappropriate tools and techniques;*

Complexity is a measure of the effort required for understanding and developing a system. Complexity is classified by [2] in essential complexity and accidental complexity. Essential complexity results directly from the modeled economic system, especially the necessary functionality and relationships between them. Reduction or elimination of this category of complexity is impossible without redefining the problem statement. Accidental complexity is generated by the methods, techniques and tools used to implement the problem. This is caused mainly by the need to satisfy the nonfunctional requirements of the system such as interoperability, scalability, availability, maintainability and security. Reducing accidental complexity involves considering these issues in the early stage of system analysis and design. The system architecture has a major impact on the effort required for further development and maintenance.

- *failure to implement a continuous updating strategy.*

An EIS need must be permanently updated in order to adapt to the changes in the business processes and the environment in which it operates in. This is an important issue specific to economic systems. According to their abilities to respond to changes in the economic environment, individual applications and economic information systems are classified in:

- adaptable applications that allow almost continuous synchronization with the organization environment due to the low cost required for implementing changes;
- static applications that have a very high change cost due to application design, implementation or technology used.

Static applications are used without significant changes until the replacement cost is covered by revenues obtained by the use of new applications tailored to the current context. The factors that trigger the degradation of adaptability for an EIS are:

- postponing the implementation of necessary changes in the application which generates over time a sharp increase in the cost of updating;
- creating redundancy in the code by avoiding the higher initial cost required for component sharing;
- making changes or extensions that are not fully or properly supported by the current

architecture of the application due to time or budget constraints.

To eliminate these deficiencies of traditional EIS this paper proposes a new type of system called *open business enterprise system - OBES*. An OBES provides holistic business process management tools, has well defined interfaces required for two-way integration with existing systems and act as a development platform for extension components.

Because economic activity in organizations is not 100% automated, the implementation of business processes requires the management of a combination of manual activities and automated activities that use functionality provided by the existing systems. This situation creates two categories of problems in classical systems:

- enterprise system integration: this is necessary to avoid the human effort required to access and synchronize information from disconnected systems;
- holistic business processes management within the organization: involves the allocation and tracking of tasks performed by human actors or the systems to meet the objectives. This requires the presence of a mechanism for defining, implementing and monitoring processes. This mechanism must allow formal process definition, easy process update and automatic execution.

An OBES provides solutions to both problems.

An open business enterprise system, **OBES = (OSC, OSP, OSS),** is an integrated set of subsystems that support operations, management and decision making, where:

- **OSC – Open System Client** is the interface that allows the final user to interact with the data and business processes managed by the system; it can be extended using modules that implement specific business operations or provide integration with legacy systems;
- **OSP – Open System Processing** is the service that implements system operations, manages process instances, exposes its functionality to the external systems using a well-defined interface, and supports extension modules for integration with other systems or implementation of specialized processing tasks;
- **OSS – Open System Storage** stores the data manipulated by the business processes in a user defined structure.

Open business enterprise systems have the following characteristics:

- **adaptability**: allows the implementation of new presentation and processing features without requiring modification of the core system architecture or code;
- **integrability**: allows integration with other systems at the user interface level by creating plug-ins; the service provides a mechanism for using the functionality implemented by other systems and expose its functionality through services; the extensible storage mechanism allows the integration of data;
- **process management**: offers formal mechanisms for describing business processes and implements a general mechanism for managing the execution of business processes, regardless of the business domain;
- **automation**: offers interfaces for constructing the automated data processing tools and allows automating repetitive tasks using scripting languages or other similar mechanisms.

The proposed system offers significant advantages compared to conventional economic systems:

- is adaptable because it offers a wide range of ways to update;
- allows reuse of existing static applications by including them in the business processes automatically or with human intervention;
- reduces the costs of integration with other systems from within or outside of the organization.

3 The Business Workflow System – BWS Architecture

An OBES architecture is a description of the structure of a system which comprises the components and the relationships between them. The proposed OBES architecture is

called Business Workflow System. The objective of the BWS architecture is to describe the components necessary for a system designed to fully automate business processes inside a modern enterprise.

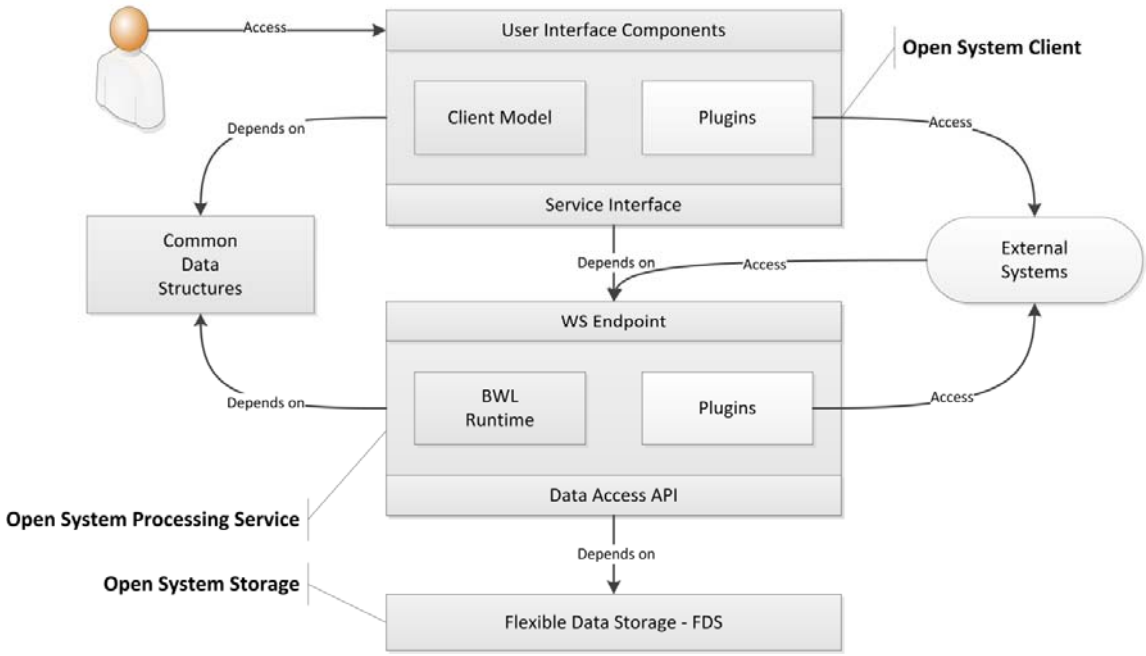

Fig. 1. BWS architecture overview

Figure 1 show the main components of the BWS architecture and the dependency relationship between them. The three major components are the one specified in the OBES definition: OSS, OSP and OSC.

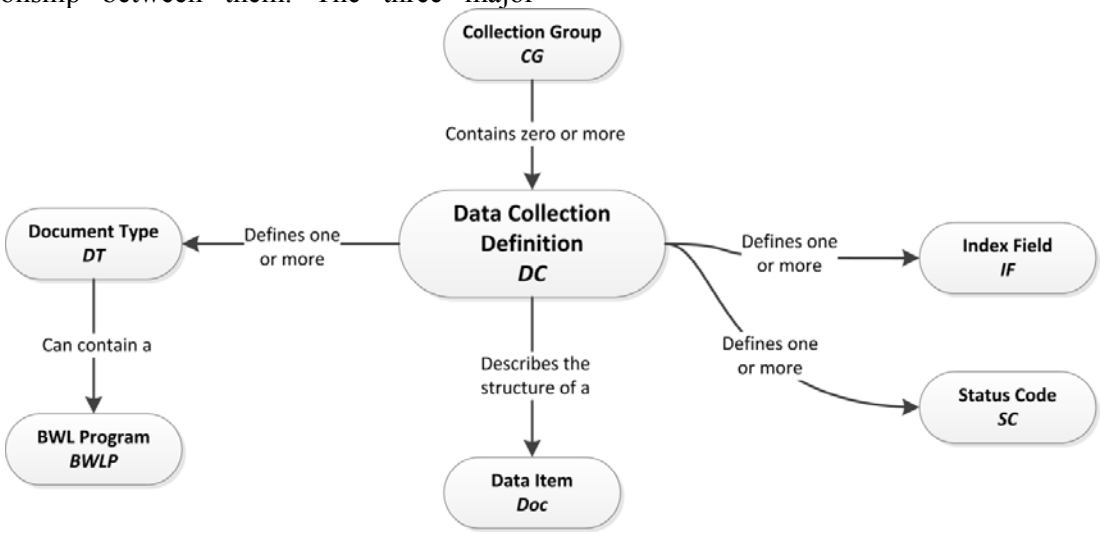

Fig. 2. Flexible data storage base concepts

The BWS implementation of the OSS is called Flexible Data Storage - FDS. The FDS allows the user to define the entities necessary to support business processes. Figure 2 shows the main concepts used to define the structure of information inside FDS.

The main concept is the Data Collection Definition – DC. A data collection $DC = (\{IF_{1-n}\}, \{DT_{1-m}\}, \{SC_{1-o}\})$ describes the structure

of the contained data items using the following structures:

- $\{IF_{1-n}\}$: a collection of *n* index field definitions; each index field definition specifies an attribute (name, data type, if required) that can be associated with each data item for retrieval purposes;
- $\{DT_{1-m}\}$: a collection of *m* document types (id, name) that defines the types of data items that can be added to the collection; each document type definition can contain an associated BWL program that is used to create process instances when a new data item is added to the system;
- $\{SC_{1-o}\}$: a collection of o status codes (id, name) that are used to specify the status of the data items contained inside the collection.

Data collections are organized for administrative purposes in collection groups. The process data is stored in data items or documents. Besides the actual data, a document contains metadata values as defined by the DC. All data objects contain an XML formatted store for extended properties associated with that particular object.

Authentication and authorization data is also stored inside the FDC. The authentication data consists of user descriptors and credentials. The authorization data consists of:

- Permission objects: triplets of the form (user id, target object id, operation type) that specify what are the allowed operations for a particular user;
- Rules objects: specify additional authorization rules based on the metadata defined by the DC.

The FDS supports data auditing. Each operation belongs to a session that records the user, location and time period. Operations performed against the data objects are recorded into the audit log. Each operation description is associated with a session and contains all a copy of all modified values. The audit log has two purposes:

- Offers the necessary information for security audits;
- Allows the system to recreate any previous system state.

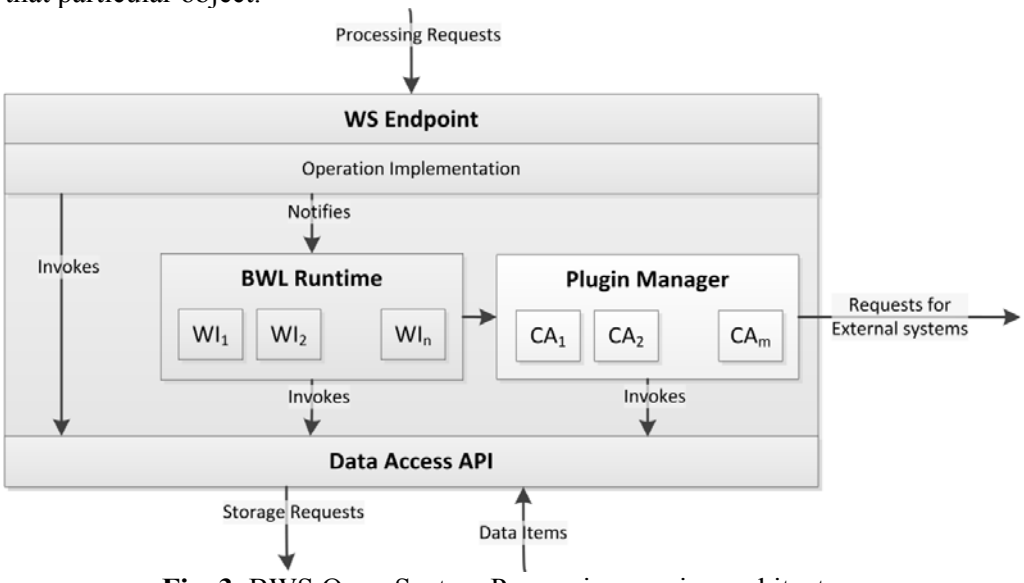

Fig. 3. BWS Open System Processing service architecture

The OSP layer of the BWS architecture (figure 3) is responsible for collecting and processing of all the operations requested by the BWS client or by external systems. The components are:

- *Data Access API*: implements the interface to the FDS and offers services like data mapping, error handling and transaction support;
- WS Endpoint: is BWS service interface to the external world; it collects and decode

all operation requests from the outside and formats the operation results;
- *Operation Implementation*: is responsible for implementing each operation supported by the system; for each request it performs the authentication and authorization, establishes a transaction context, invokes the Data Access API to perform the actual processing and notifies the operation to the BWL runtime;
- *BWL Runtime*: manages the BWL workflow instances, notifies them when relevant event occur and offers them access to the FDS using the Data Access API;
- *Plugin Manager*: is responsible for loading and managing the platform extension components; the processing plugins that run inside the BWS service are responsible for customized data processing of the data items and access to the external systems for system integration purposes.

The Business Workflow Language – BWS implemented inside the service is described in detail in [8] and [7].

The OSC layer of BWS (figure 4) offers end users access to the systems. It is built around the BWS Client Model. The proposed model facilitates the creation of a completely extensible client with full automation support. The BWS Client Model consists of:
- Data structures that describe the UI components and process data;
- Data modification event publishers;
- Operation event publishers.

The main components of the BWS clients are:
- *BWS Client Model*: maintains the current client state, mediates the interaction between client components using events, provides automation support and uses the Service Interface component to send operation requests to the BWS service;
- *Catalog Cache*: maintains a consistent in-memory copy of the systems' global catalog;
- *Scanner Interface*: provides and interface between the client application and scanner machines used to collect document images using the TWAIN or WIA protocols;
- *Transfer Manager*: is responsible for managing data item transfers between the client and the server;
- *Plugin Manager*: loads and executes dynamically the client extensions modules;
- *Scripting engine*: compiles and execute scripts that use the BWS Client Model for task automation inside the client;
- *BWL Designer*: provides a visual editing services for describing business processes using the BWL language;
- *User Interface Components*: are the actual visualization components used to present information to the user and collect input; they invoke operations on the model in response to user actions and subscribe to the published events for updating the interface;
- *Service Interface*: provides data mapping and the infrastructure required to communicate with the BWS service endpoint.

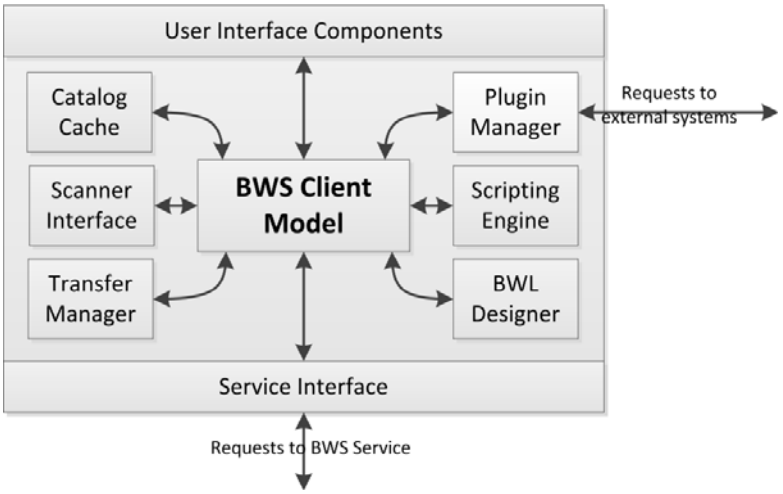

Fig. 4. BWS Open System Client architecture

The client application is responsible for linking the model operation events to the corresponding operation implementations at startup. The extension modules have the ability to intercept and respond to those events in order to implement custom functionality inside the user interface. The extension modules have also the ability to change the user interface by manipulating the *BWS Client Model*.

4 BWS Implementation and Integration Methodology

The BWS architecture is implemented inside the DocuMentor business management platform (Figure 5).

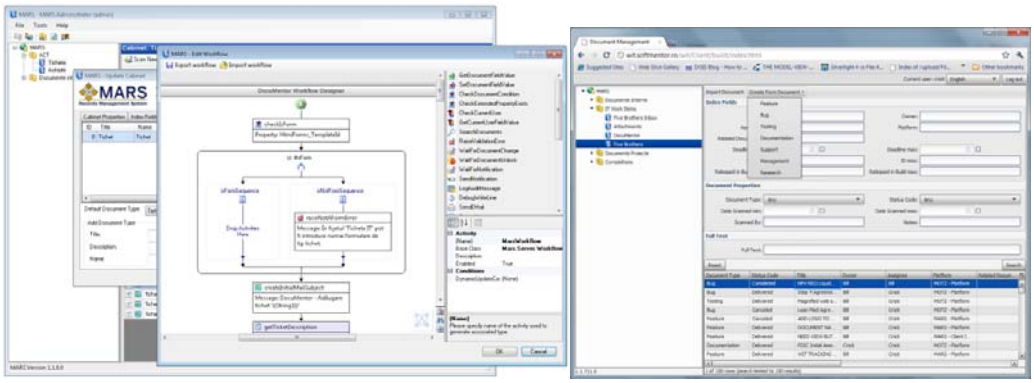

Fig. 5. DocuMentor standard client, business process designer and web client

DocuMentor is an OBES designed to fulfill three roles inside the organization: data management, process management and rapid application development platform.

Running a business process involves handling large volumes of structured or unstructured information ([6]). The platform allows the user to define the structure, retrieval and access rules for the stored entities via metadata stored inside the global catalog.

Data stored inside the platform document is organized according to the BWS architecture through data collection grouped into collection groups named departments. The standard client (figure 5) offers instruments to define the index fields, document types, BWL programs and status codes for each collection. The user interface for data retrieval is generated dynamically based on data collection definition. The platform implements a full text indexing system for data collection that allows users to retrieve data items using content based queries. Stored data items can be any form of digital content up to 2GB in size.

Interpretation of documents is done by extension modules. The platform interprets only the associated metadata.
Access to documents is controlled through authorization and authentication operations applied to each request for modification or retrieval. Authorization of operations initiated by the user is performed based on the permissions and rules defined at the collection level. Changes to the documents have the effect of creating new versions. The system stores the current version of the document and all information necessary for reconstruction of previous versions standard according to the RFC 3284 - The Data Compression and differencing VCDIFF Generic ([9]) standard. All operations performed on its content or associated data are stored in the audit log. This allows identification of changes generated by all operations executed by the system and the recreation of any previous system state. Document retrieval is performed using a specially created retrieval language. The language allows identification of documents based on the index fields defined and on the basis of its content.
As a business process management platform, the DocuMentor platform implements the original Business Workflow Language – BWL defined by the BWS architecture. The language implementation provides the all the necessary statements for specifying flow control and performing basic operations on documents and data collections. Instructions for performing specialized operations or integration with other systems are dynamically added to the language using plugins. BWL programs are created and modified using an integrated visual editor (figure 5). Programs are stored in the global catalog and are associated with document types. Process initiation is performed automatically by the system when a new document is created. Process instances running within the platform are notified on operation performed against the associated document.
Implementation of specialized data processing tasks or integration with external systems is done through extension modules. These are packages of .NET classes stored inside the global catalog and loaded at runtime by the platform. An extension module can extend the user interface and processing operations performed by BWL programs.
Extending the user interface is accomplished by using facilities provided by the client application model. Client component of the extension modules is able to freely use the facilities offered by the Windows Forms component of Microsoft. Net Framework or the operating system.
Extending the server processing model is achieved by creating new BWL activities. These activities are loaded at runtime by the execution engine when it encounters a BWL program using these activities. Activities added by the extension modules are available for usage inside the visual editor.
Development of the DocuMentor platform was carried out in three distinct phases corresponding to the three roles performed by the platform in the organization.
In the first stage the basic data management functionality was implemented in the form of the MARS (Management, Archival and Retrieval System) engine.
In the second stage all the necessary infrastructure for implementing the BWL business process management was implemented.
In the third stage consisted of the creation of the plugin based extension model. It provides the tools needed to create modules or applications necessary to implement specific tasks and to integrate existing systems within the organization's business processes.
Projects that aim to integrate enterprise applications are generally very complex and dynamic. The most known and used methodology for the implementation of integration projects is GERAM (Generalized Enterprise Reference Architecture and Methodology, [3]) developed by the IFAC / IFIP Task Force on Architectures for Enterprise Integration. This involves creating a global business model based on generic enterprise models (GEM), the generic modules (GM) and some specific ontological theories (OT). The integration solution is then developed based on this model. The traditional approach based

on collecting all the requirements the proceeding with the system design, implementation and testing have failed in most situations. According to [10] and [1] approximately 70% of integration projects have failed. The main cause of failure was the lack of tools, methodologies and qualified personnel.

To reduce risks and increase the success rate of integration projects based on the BWS architecture and DocuMentor platform a new iterative development methodology called DM-CPI (DocuMentor - Continuous Process Improvement) was created.

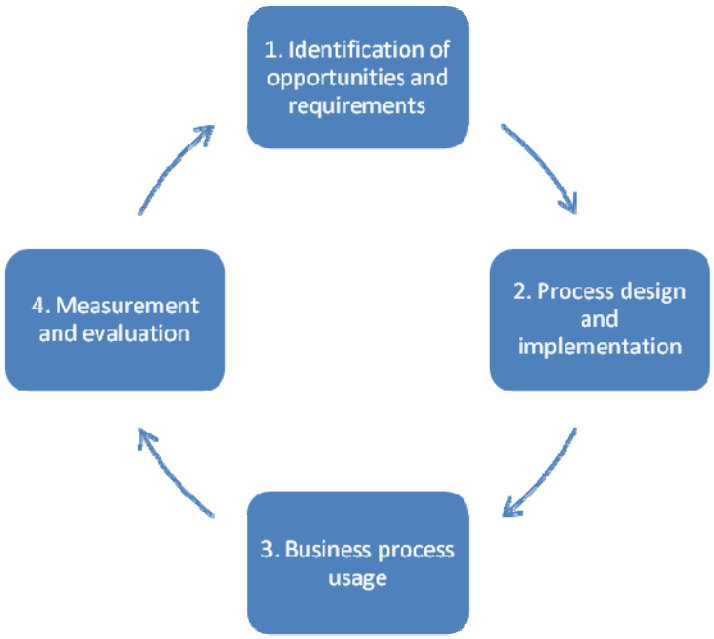

Fig. 6. DB-CPI iteration stages

DM-CPI methodology proposes a gradual approach to enterprise integration. Unlike traditional holistic approaches, DM-CPI does not propose to complete the initial modeling of the organization and its processes in a waterfall project. The proposed method is iterative. The integration project is implemented in successive iterations aimed at integration or improving of a particular process within the organization. Figure 6 shows the four stages that make up a DM-CPI iteration: identification of opportunities and requirements, process design and implementation, business process usage and evaluation.

The first stage identifies the opportunities and requirements and selects a process or a group of processes that will be the goal of the iteration. For each process the impact on the organization's performance and effort required for implementation is assessed. The processes with the greatest benefits / effort report are chosen for implementation. For the selected processes we determine the objectives to be achieved, the integration techniques and acceptance criteria.

The process design and implementation phase involves the development and testing of the processes selected in the identification phase using the tools provided by the DocuMentor platform.

In the business process usage phase processes created in the previous phase are put into production use. During this period we collect information relevant to the process execution by using the platform audit facilities and by interviewing people involved in the process.

In the measurement and evaluation phase data collected during the previous phase is analyzed. The purpose of this stage is to highlight the problems occurring in the implementation process and to identify improvement opportunities. The results obtained in this stage are used as input to the next iteration of the integration process.

The DocuMentor platform three integration techniques: task based integration, user interface integration, and service integration.

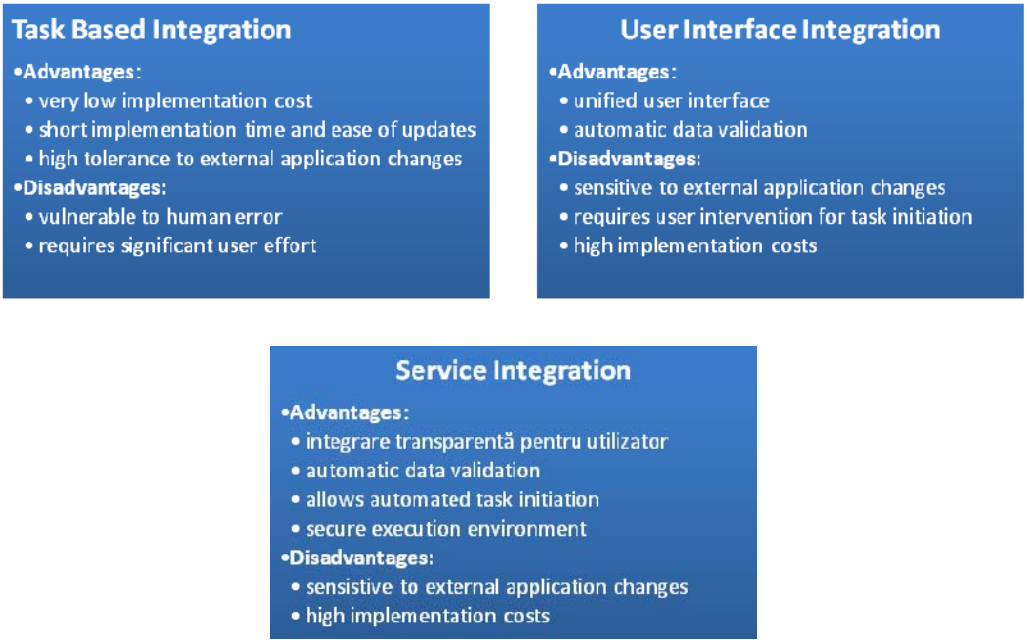

Fig. 7. DB-CPI iteration techniques

Figure 7 shows the advantages and disadvantages of each integration technique. Task based integration is the fastest method of integration. This does not require modification of existing systems or creating new extension modules. The technique involves identifying the tasks to be performed to complete the business process, of those responsible for each task, the documents involved and the flow of execution. Development consists of identifying and defining of the data collections, designing the data entry forms and specifying the process using the BWL language. BWL programs will use only the predefined activities for processing information in stored in documents and task management. Because the implementation requires minimal initial development and maintenance effort, the technique is preferred for initial integration iterations, for processes that suffer temporary or frequent changes and processes that cannot be automated. Another advantage of the task based integration techniques is the loose coupling with existing systems. Changes the external systems do not lead to substantial changes of integration project. The main disadvantage of this method is that the effective execution of most activities still falls in the responsibility of the end users. They are still forced to use existing systems or manually process the information. This can introduce human errors, inconsistencies in data and involves a greater effort from users.

User interface integration requires the creation of specialized modules in the platforms' client for data processing and communication with the external systems involved in the process. Developing a unified user interface requires a significant effort as it involves both using the BWS client automation model and communication with existing systems. The major advantage of the approach is that the end user uses a single UI integrated inside the DocuMentor client. The end users' is simplified and the data quality is significantly improved. Disadvantages of this approach are sensitivity to external application changes and the inability to perform actions that are not initiated by the user. Changes in the external systems must be reflected immediately in the integration module. This can lead to inability to execute processes supported by the module and generates significant addi-

tional costs. Because the module operates in the client application, it cannot initiate actions automatically (for example at some point in time or on the occurrence of an event in the source system). For such situations it is necessary to use the service integration technique.

The service integration technique in DocuMentor implies building custom BWL activities for integration with the target systems. These activities are then used to build BWL programs that implement business process. As in the case of user interface integration, the service integration requires more effort than integrating the task based technique and introduces a strong coupling between the integration solution and the external systems. Advantages of this technique are the possibility of automated operation initiation, execution in a secure environment and that fact that the integration is transparent to the final user.

The three techniques can be used independently or can be combined in an integration project. The DM-CPI methodology encourages the use of the task based technique in the initial iterations and gradual implementation of the user interface and service integration in subsequent iterations. Initial use of the task based integration technique enables rapid commissioning of the system and enables the identification of critical activities will benefit from further automation.

5 Conclusions

This paper proposed a new enterprise system type and the associated BWS architecture. The feasibility of the architecture was demonstrated by its implementation into the DocuMentor platform. DocuMentor was implemented in Romanian and US business organizations such as US Bank, Mortgage Outreach, FB Mortgage Company Services and Securing, Depozitarul Central al BVB – RoClear, Autonom Services, UMB Grup sau Master Business and Consulting. Based on the experience accumulated during these implementations a new enterprise integration methodology named DM-CPI was created and refined. The final section of the paper presented the concepts, stages and techniques employed by the methodology.

References

[1] M. van den Bosch, M. van Steenbergen, M. Lamaitre, R. Bos, "A Selection-Method for Enterprise Application Integration Solutions," *9th International Conference - BIR 2010*, Rostock, September 29 – October 1, 2010, Perspectives in Business Informatics Research, editor Fobrig P., Lecture Notes in Business Information Processing, vol. 64, Part 4, Springer Verlang, Berlin, 2010, pp. 176-187, ISBN 978-3642161001

[2] F. P. Brooks, "No Silver Bullet — Essence and Accidents of Software Engineering," *IEEE Computer*, Vol. 20, No. 4, 1987, pp. 10–19, ISSN 0018-9162

[3] D. Chena, G. Doumeingtsb, F. Vernadatc, "Architectures for enterprise integration and interoperability: Past, present and future," *Computers in Industry*, Vol. 59, No. 7, 2008, pp. 647-659, ISSN 0166-3615

[4] M. Dumas M., W.M.P. van der Aalst, A.H.M. ter Hofstede, *Process Aware Information Systems. Bridging People and Software Through Process Technology*, Wiley Interscience, Hoboken NJ, 2005, 432 pg., ISBN 978-0-471-66306-5

[5] J. Greenfield, K. Short, *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*, John Wiley & Sons, Hoboken NJ, 2004, 696 pp., ISBN 978-0-471-20284-4

[6] A. ter Hofstede, W. van der Aalst, M. Adams, N. Russell, *Modern Business Process Automation*, Springer-Verlag, Berlin, 2010, 664 pg., ISBN 978-3-642-03120-5

[7] C. Ioniţă, "Collaborative Business Process Optimization Using Domain Specific Languages," *9th International Conference on Informatics in Economy*, May 7 – 8, 2009, Bucuresti, Education, Research & Business Technologies, editor Ută I. A., Editura ASE, Bucuresti, 2009, pp. 32 - 38, ISBN 978-606-505-172-2

[8] C. Ioniţă, "Analytic Evaluation of BWL Nets," *Economy Informatics*, Vol. 11, No. 1, 2011, pp. 42 - 151, ISSN 1582-7941
[9] D. Korn, J. MacDonald, J. Mogul, *The VCDIFF Generic Differencing and Compression Data Format*, The Internet Engineering Task Force, Fremont CA, 2002.
[10] G. Trotta, *Dancing Around EAI 'Bear Traps'*, Available at http://www.ebizq.net/topics/int_sbp/features/3463.html, accessed at Sep 18 2011, 2003

Cristian IONIŢĂ has graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 2004. He is a PhD student at the Academy of Economic Studies Bucharest since 2006 and a teaching assistant inside the Department of Computer Science in Economics at Faculty of Cybernetics, Statistics and Economic Informatics. Main interest topics are programming languages, data structures, distributed applications and enterprise systems. He is the author or coauthor of 16 journal articles and coauthor in 2 books on these topics.