# Developing a Realistic Workflow Management Environment for Teaching: An Interface from YAWL to OpenERP

Joerg Evermann

Faculty of Business Administration, Memorial University of Newfoundland, Canada
jevermann@mun.ca

**Abstract.** The paper describes an interface from the YAWL workflow management system to the OpenERP enterprise system. The interface is implemented as a codelet, and provides access to the full range of OpenERP information and functions. The paper provides an overview of the design of the codelet, the data types for its use, and an example application.

**Keywords.** YAWL, OpenERP, workflow management, enterprise system

## 1 Introduction

The YAWL (Yet Another Workflow Language) workflow management system is developed for teaching and research. Together with the YAWL book [1] it provides a state-of-the-art open-source environment for teaching business process management and automation, in an affordable and easily accessible way. The YAWL system is currently used in an introductory, core course on Business Process Management (BPM) in the general business degree curriculum at Memorial University. While this course covers all aspects of BPM, strategic, managerial, operational, and technical, YAWL plays a prominent part in the course to illustrate the capabilities and benefits, but also the complexities of process automation.

In the fall of 2011, Memorial University funded, through a $5000 Instructional Development Grant, a course improvement project aimed at introducing hands-on ERP systems experience into the business undergraduate curriculum. The initial targets of this initiative were three courses: Information Systems, Operations Management, and Business Process Management course. From the BPM course perspective, integrating the workflow management with ‘’legacy’’ information systems provides for a more realistic environment for students to experience and learn about workflow management and process automation.

While the business benefits of integrating workflows and enterprise systems are easy to describe, they are difficult for students, especially those at an early stage in the degree program, to fully appreciate without hands-on experience. Experiential learning, defined as “the sense-making process of active engagement between the inner world of the person and the outer world of the environment” [2, pg. 2], is a “more

effective and long-lasting form of learning" that "involves the learner by creating a meaningful learning experience," [2, pg. 1] and "learning from experience is one of the most fundamental and natural means of learning available," [2, pg. 15].

The project's intended learning outcomes are an improved understanding and appreciation of the capabilities and importance of integrated enterprise IT to business operations.

Given the size of the development grant, and the inability of the faculty to provide additional monetary or human resources to this project, the scale of the project was quite limited. The chosen enterprise system was OpenERP[1], an open-source system that provides core modules such as sales, purchasing, accounting, production management, as well as extensions for point-of-sales, project management, etc. Among open-source ERP systems, it is one of the most mature and feature complete systems. OpenERP provides its own process model and workflow engine. However, the configuration language is XML based and there is no recognizable formal underpinning for the workflow description language.

This suggested the development of an interface from YAWL to OpenERP, so that OpenERP functionality can be used in a YAWL workflow. The remainder of the paper describes the implementation of this interface as a YAWL codelet. The next section provides an introduction to the OpenERP system, followed by design choices for the interface codelet. This is followed by a description of the codelet parameters and data types, and an example workflow.

The codelet, associated XML Schema data type definitions, and YAWL example workflow definitions are available under the GPL v2 license[2].

## 2 The OpenERP System from the YAWL perspective

OpenERP is developed using the Python language and provides a business object model that abstracts, through an object-relational mapping layer, from the underlying physical data structures and functions. On top of this object model, OpenERP defines a process model and workflow mechanism for many of the business objects. The OpenERP workflow model is based on object states and transitions between them. Each object *state* is associated with a method, whereas transitions are either triggered by signals, or triggered by changes in attribute values.

OpenERP provides an XML-RPC based web-services interface to both its business objects and its workflow mechanism. This interface provides the following generic operations on all business objects:

- Create (returns the new business object ID)
- Search (returns a set of business object IDs that match a query)
- Read (returns a set of attribute values for a given list of business object IDs and a given list of attribute names)

[1] http://www.openerp.com

[2] http://www.yawlfoundation.org/pages/resources/contributions.html

- Write (updates the provided attributes of business objects with a given list of IDs with the provided new value)
- Delete ("Unlink")

OpenERP also provides a means to call any method defined on the business objects. However, calling the methods directly (outside the built-in, intended workflow) may lead to issues such as the (implicit) pre-conditions (as per the built-in workflow) not being met, or the consequent actions (as per the built-in workflow) not being executed. Thus, calling the business object methods directly is not recommended. Instead, OpenERP provides a mechanism to send "signals" to its workflows. These signals can be used to advance the built-in workflow for a business object. For example, a signal may be sent to confirm a draft sales quotation and transform it to a sales order. The OpenERP workflow mechanism then calls the appropriate methods on the business object.

While this is a "safe" mechanism to interact with the OpenERP system, the externally defined and controlled workflow (e.g. from YAWL) must be essentially isomorphic to the workflow configured in OpenERP and can only 'mimic' that workflow. Further, developing an external workflow requires a thorough understanding of the built-in workflow and the states of the business objects. However, the codelet is not limited to this "safe" way of interacting with OpenERP, and it can in principle be used to call any method in any order. This, however, requires a thorough understanding of the OpenERP pre-conditions and consequent actions.

## 3 Interface Design

YAWL provides different ways to integrate external systems, the three most prominent being the web-services invoker service, the codelet mechanism, and the external data gateway. Any of these can in principle be used to develop the interface.

As the YAWL web-service invoker service requires a valid WSDL file and uses SOAP rather than XML-RPC, this would have required a translation server that accepts SOAP requests and issues XML-RPC requests in turn. Alternatively, a new XML-RPC web-service invoker service could have been built as an alternative to the existing SOAP based one. Either of these alternatives was considered to be too technically complex for the limited time-frame of the project.

An external data gateway could be constructed to access either the business object information in OpenERP, or directly access the underlying relational data. Technically, one could also imagine that this might be used to access methods or send workflow signals, but this would be conceptually confusing, as the data gateway is intended primarily for data access.

Instead, a simple codelet was developed that accepts input and provides output using pre-specified data types. Two options were investigated:

- Offer access to specific OpenERP business objects, their data, methods, and workflow signals. In this scenario, XML data types would need to be developed that reflect the OpenERP business object model, e.g. a data type for the "sales order" ob-

ject, a data type for "sales order line" object, etc. This would remove the burden of data type development from the YAWL process designer, but would at the same time limit the flexibility of the codelet to a fixed set of business objects, their data and methods as determined by the codelet design.
- Offer access to generic as well as specific OpenERP operations with no abstraction from the OpenERP XML-RPC API. This requires the codelet user, i.e. the YAWL process designer, to develop appropriate business object data types and deal with the specifics of data transformation to the OpenERP API on the YAWL side, e.g. as part of the input and output mappings for tasks. The benefit is that the codelet does not prescribe specific data types, and it can access any OpenERP business object or workflow. The codelet was developed based on this, second model.

The codelet itself is stateless and establishes a new connection to the OpenERP system for every call, thus requiring the OpenERP connection information with every call. While this may not be as efficient as returning a connection handle to the YAWL workflow, the fact that the YAWL workflow may be long-running means that a connection handle in the YAWL workflow data might expire. Further, as tasks in the YAWL workflow may be assigned to different (human) resources, maintaining a quasi-persistent connection handle would also force the same OpenERP user account for the entire workflow, which may not be desirable in practice.

## 4 Use and Example

Table 1 lists the input parameters for the codelet. The content of the `method` parameter is limited to the five generic methods for OpenERP business objects: `search`, `write`, `read`, `delete`, `create`, and the additional `action`, which is used for sending signals to OpenERP workflows. The results that the codelet returns depend on the invoked method. Alternatively, the codelet returns an error, either passed back from OpenERP, or an exception in the codelet, or an error encountered by the codelet, e.g. when the input paramters do not match the invoked method type.

| Parameter | Type | Description |
|---|---|---|
| `URL` | `xsd:String` | Hostname for OpenERP |
| `Port` | `xsd:Integer` | Network port for OpenERP |
| `Database` | `xsd:String` | OpenERP database to select |
| `Username` | `xsd:String` | Username for OpenERP |
| `Password` | `xsd:String` | Password for OpenERP |
| `Object` | `xsd:String` | Type of OpenERP business object on which method or action is to be called |
| `Method` | `xsd:String` | OpenERP method name |
| `Parameters` | `ParameterType` | Parameters appropriate for the called method |

**Table 1.** OpenERP codelet input parameters

Figure 2 below shows the YAWL workflow for creating and processing sales order. That workflow is based on managing sales orders in OpenERP, shown in Figure 1 above. The codelet design decisions have an impact on the usage of the codelet in two important ways. First, the direct representation of the basic OpenERP method calls (search, read, etc.) leads to typical combinations of search-read sequences as two automated tasks in the YAWL workflow. Second, the relatively low degree of abstraction of the codelet parameters suggests that the data transformations in the YAWL task input-/output-mappings are not trivial. Thus, the use of the codelet requires considerable XQuery expertise.

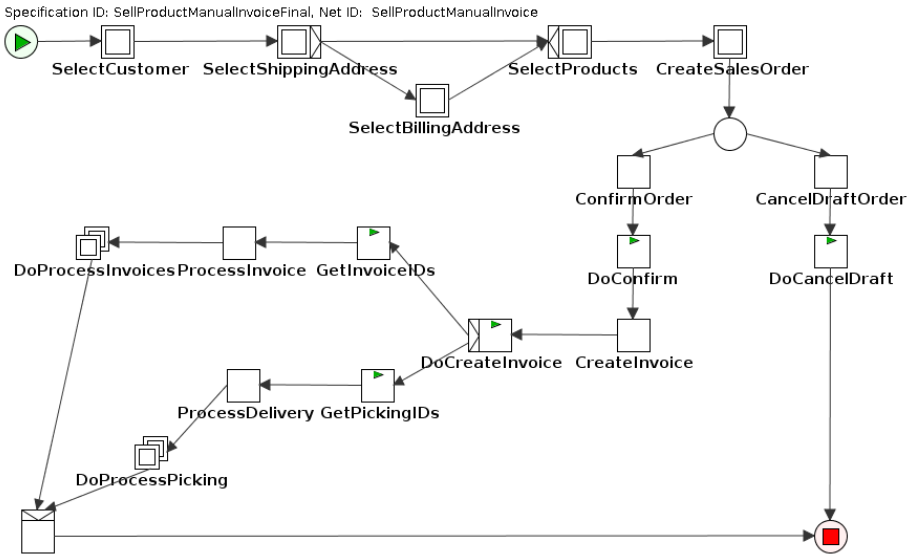


**Fig. 2.** Sales order process

## 5 Use in Teaching and Current Status

When the project was initiated, the intended use of the YAWL-OpenERP interface was to allow students to create realistic workflow definitions for simple processes like sales order processing, as part of an assignment or course project. It was hoped that by using a realistic integration with business data in the ERP system, the usefulness of workflow management could be demonstrated to students and lead to better appreciation and understanding of the business value of process automation. Specifically, the experiential learning is argued to:

- increase student engagement,
- improve student skill development,
- increase student learning and understanding.

As the codelet implementation is now complete and an example process (Figure 2) is implemented, we have found that the level of YAWL, OpenERP, and XML knowledge required to develop integrated workflows is beyond what can be taught in an introductory course that has no computer science or programming pre-requisites.

There are a number of possible responses to this situation. First, we are aiming to have students interact with the process, rather than create their own workflows. For the coming fall semester, we are working to make the defined workflow more robust to user error and to provide a better user interface. At the same time we are developing computer lab exercises for students to use the pre-defined workflow. Finally, we are developing workflows for processes other than sales order management.

A second possible response is to investigate the aggregation of lower-level tasks that access the codelet into higher-level workflow fragments, whose use does not require knowledge of OpenERP, XML, or the codelet design. One option for this is the use of worklets which can then be assembled by students. However, the current course design does not include worklets or declarative workflow design. Another alternative is to provide students with a workflow specification that contains a number of YAWL nets, which can either be copied-and-pasted into students' own work, or to which composite tasks that students create as part of their own workflows can be unfolded. The current example process (Figure 2) already contains many such abstractions as composite tasks. Again, a set of computer lab exercises for students will be developed around this idea in time for the fall semester.

### 5.1 Evaluation

The intended learning outcomes will be evaluated using questionnaires measuring student engagement, adapted from [3, 4], perceived skill development, adapted from [5]. Further, questions on student's understanding of workflow management principles and the role and capabilities of YAWL in process automation will be used:

- In your own words, describe what the YAWL system is. (Q1)
- In your own words, describe the place of the YAWL system in a company. (Q2)
- In your own words, describe how the YAWL system relates to other information systems in a company.(Q3)
- In your own words, describe why and how the YAWL system can be useful to a company. (Q4)

Additionally, Likert-type scales will be used for the following questions:

- I have a good understanding of workflow management (Q5a)
- I am able to explain workflow management to other students (Q5b)
- I am able to use workflow management systems (Q5c)
- I am able to make a business case for workflow management. (Q5d)

. These questionnaires will be administered using a pre- and post-test design before and after a computer lab class that involves the OpenERP interface. This allows us to measure the changes in students introduced by the realistic workflow environment. Additionally, on the post-test questionnaire, students motivation was assessed, using the following Likert-type scales:

- I would discuss related topics outside the class. (Q6a)

- I would do additional reading on related topics. (Q6b)
- I would do some thinking for myself about related issues. (Q6c)

### 5.2 Initial Results

To satisfy the requirements of the instructional development grant under which the project was funded, a preliminary evaluation of benefits had to be assessed, even though the OpenERP interface was not yet fully integrated into the pedagogics of the course. To meet the deadline, the integration between WfMS and ERP was demonstrated by the course instructor using the sales order management process in Figure 2. Students were shown the workflow definition, the OpenERP data, and the running workflow. As expected, a demonstration is not as engaging or interesting as experiential learning, and the initial results reflected this.

From a total of 77 students, 57 responses were received, of which 53 provided information on both the before and after questionnaire.

The understanding questions (Q5a-Q5d) were averaged as they all represent understanding of WfMS. There was no significant difference between the means for the before and after questionnaire (before = 3.80, after = 3.97, on a 7-point scale).

The motivation questions (Q6a-Q6c) were used only on the post-test questionnaire. The results indicate moderate motivation levels (approx. scale mid-point) for Q6a and Q6b, whereas Q6c shows good motivation levels. The difference is not surprising, as the Q6a and Q6b asked students whether they would take some action, whereas Q6c only asked whether they would "think about" the topic.

Finally, we examined the engagement [3, 4], perceived skill development [5] and perceived usefulness (single item). The descriptive results are shown in Table 2.

| | **Mean** | **SD** |
|---|---|---|
| Perceived Engagement | 3.33 | 1.39 |
| Perceived Skill-Development | 4.04 | 1.29 |
| Perceived Usefulness | 4.26 | 1.58 |

**Table 2.** Results for engagement and skill development, on 7-point Likert scales

The results indicate that the demonstration was not engaging to students (mean less than scale mid-point, but not significant as per t-test). However, the demonstration was perceived as improving skill development (mean significantly above scale mid-point, $p<0.01$) and useful (mean significantly above scale mid-point, $p<0.01$). The result with respect to engagement is not surprising as the demonstration required students to watch for 15 minutes rather than interacting with the system themselves in a true experiential way, as originally intended. The results with respect to skill development and usefulness are encouraging, especially given the low level of student engagement. We believe that this can be significantly increased once true experiential interaction with the system is available.

Only 18 responses were received with answers for Q1-Q4 differing between the before-demonstration and after-demonstration questionnaire. The answers were examined to identify improvements in understanding. Of the 18 respondents, only 12 showed improvements in understanding and even fewer showed a marked improvement across all four questions. The low response rate is likely due to the low level of engagement..

## 6 Discussion and Conclusion

This paper presented a YAWL codelet to access the OpenERP system. The codelet exposes low-level access functionality, rather than business-level objects or methods. This low level of abstraction requires the codelet user to have a thorough understanding of the OpenERP data model, methods and workflows. At the same time, this design makes the codelet useful for the widest range of applications. Codelet users and workflow designers may also use YAWL features to build additional layers of abstraction on top of this foundation. For example, YAWL worklets could be defined that aggregate some of the codelet functions, e.g. the search-read combinations, into assemblies that are meaningful at the business level.

Designed as part of a project to introduce experiential learning into an undergraduate business degree, the codelet is essentially feature-complete and robust, but the development of teaching material, example workflows, and associated documentation is still in progress. An evaluation of the usefulness of the OpenERP interface for teaching workflow management is planned, based on goals such as increased understanding, increased student engagement, and increased skill development. An evaluation that was not based on experiential learning has only shown limited improvements on these dimensions, but this is expected to improve once the interface is ready for its intended use in the classroom and computer lab tutorials.

While this project focused on the OpenERP system, we believe that the challenges we faced, and the design decisions we made, e.g. w.r.t. having to "mimic" the internal workflow or the choice between providing abstraction or a direct interface, are not unique to OpenERP and applicable to other open-source or commercial ERP systems.

## 7 Bibliography


1. Ter Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M., Russel, N.: Modern Business Process Automation. Springer, Heidelberg (2010)
2. Beard, C.M., Wilson, J.P. (2006) Experiential Learning: A Best Practice Handbook for Educators and Trainers (2nd. Ed.). Kogan Page Limited, Philadelphia, PA.
3. Webster, J. and Ahuja, J.S. (2006) "Enhancing the design of web-navigation systems: The influence of user disorientation on engagement and performance," MIS Quarterly, (30:3), pp. 661-678.
4. Webster, J. and Ho, H. (1997) "Audience Engagement in Multimedia Presentations," The DATABASE on Advances in Information Systems, (28:2), pp. 63-77.
5. Alavi, M. (1994) "Computer-Mediated Collaborative Learning: An Empirical Evaluation," MIS Quarterly, (June), pp. 159-174.