Implementing CEMIS Workflows with State Chart XML

Jörg Bremer¹, Tariq Mahmoud¹, Barbara Rapp¹

Abstract

Contemporary CEMIS do not cope with requirements from the sustainability discussion. At the time of reporting environmental performance, it is too late to set the right course. Without an early identification of cause and effect to anticipate environmental impacts of decisions for timely intervention, potentials for acting precaution remain unexploited. A resource-friendly design of processes and their controlling demands for sustainability oriented organizational structures. The methodical view is also often neglected. Innovative solutions are in demand that foster the seamless integration of external environmental data together with in-house process, product and organizational data into environmental performance calculations in order to gain a decision base for strategic environmental management. Moreover, the huge variety of individually needed applications or adaptations of applications to an organization's individual needs demands for a means to plug together rather small functionalities to a tailor made system in a flexible manner. In this paper, we introduce the workflow centric approach within the IT-for-Green project (http://www.it-for-green.eu) and give an overview of the used techniques.

1. Introduction

Within the project IT-for-Green, a new type of CEMIS is developed with a service-oriented infrastructure as a runtime environment for services that are integrated on demand (Rapp et al. 2011). Such services may include single tasks in the field of energy estimation, life cycle assessment, sustainability reporting, communication, environmental conscious product design, measuring environmental performance, and many more. In order to integrate all these services so as to obtain an individually configured CEMIS, conceptually, a specialized workflow component will be the glue that will make up the kernel of the new IT-for-Green integration platform. At a later point, the same CEMIS workflows are supposed to drive different platform implementations up to stand-alone desktop applications for client applications on a smaller scale. Hence, we were looking for a flexible, highly customizable and especially lean system for defining and running our workflows.

State Chart XML (SCXML; http://www.w3.org/TR/scxml) is a general-purpose event-based state machine language. SCXML provides an XML-based representation of so called Harel State Tables, which is a state machine notation with a clean and well-thought out semantics for sophisticated constructs such as parallel states. SCXML combines Harel semantics with an XML syntax and event notation. SCXML is considered a candidate for the control language within multiple markup languages coming out of the W3C and is general-purpose event-based state machine language, that can be easily adapted to one’s own needs by flexibly extending it with implementations for own activity definitions. We propose the use of SCXML as a definition language for workflows and present the concept and preliminary considerations for implementation.

The rest of this paper is structured as follows: We start with an overview on the workflow reference model and the IT-for-Green platform. We present the general workflow concept and a short example before conclusion.

¹ University of Oldenburg, Department of Computing Science, Uhlhornsweg 84, D-26129 Oldenburg
2. Related Work

This section gives brief information about the related work in the field of business processes and workflows. Business processes are technically implemented as workflows with the help of different definition tools. These implemented workflows are then executed using the facilities provided by different workflow engines’ vendors. Different players in the field of workflow development exist in the market. The most successful commercial engines in the workflow management domain are Microsoft and IBM. Starting from its third version of .NET Framework, Microsoft Corporation launched the Windows Workflow Foundation in-process workflow engine (Schichtenberg, 2007; Scribner, 2007). This engine can be used to implement different types of workflows within .NET applications. This workflow engine is integrated in several products like for example in the Microsoft Office SharePoint Server (2007 and newer versions) and the Microsoft Dynamics CRM (starting from the 4th version).

Open source workflow engines are nowadays mature and able even to compete their commercial counterparts. An example of such workflow engines is the Java-based software framework: Yet Another Workflow Language (YAWL). This workflow engine is based on its own modelling language. It integrates organizational resources and Web Services (van der Aalst et al., 2004). Other existing workflow engines in the market are: the J2EE-based project WfMOpen, ActiveVOS (previously known as ActiveBPEL) and the IBM WebSphere Process Server (which execute Business Process Execution Language for Web Service modelled business processes).

2.1 Workflow Management Systems

In this section, we are going to illustrate the main workflow reference model as it had been proposed by the Workflow Management Coalition (WFMC) (see http://www wfmc.org/). This reference model is depicted in Figure 1.

As can be seen in the figure above, the main and central part of this reference model is the workflow enactment service. It represents the core component of any workflow system. It is comprised of one or more workflow engines in which each engine deals with bunch of cases (processes). These engines are designed to enhance scalability of the systems’ to which they are applied. These engines are potentially not noticed by the user.
The first interface in this workflow reference model is with the process definition tools. These tools are responsible of creating/deleting cases, routing access among cases, managing case’s attributes, managing/handling triggers, recording historical data, providing a workflow’s summary, monitoring the consistency of a workflow, starting up application software during activity execution and finally submitting work items to the right resources based on specific resource classification. These tools include: process definition, resource classification, and analysis tools.

The second interface in the reference model is with the client applications. It is used by employees mainly to execute processes. These applications include standard or integrated workflow handlers that present the work items and their attributes. These handlers provide relevant properties of a work item, provide sort and select of a work item, and reports the completion of an activity.

The third interface in the reference model is with the invoked applications that can be started while the task performs. These applications don’t normally form a part of the workflow management systems. Rather, they belong to the workflow system itself. They are normally classified into interactive and fully-automatic applications. Interactive application can be initiated by the selection of a work item and can be any kind of text processors, spread sheet, or electronic form filling tools.

The fourth interface in the reference model is with the other enactment services. The main responsibility of this interface is to link several kinds of workflow systems. Via this interface, the cases or parts of these cases can be transferred from one workflow system to another. This interface is mainly designed and developed to enable workflow interoperability in general.

The last interface in this reference model is with the administration and monitoring tools. These tools can be split into the operational management tools and the recording and reporting tools. The first operational management tools can be either not case-related or case-related to administer the cases’ management and to inspect the logistical state of the cases respectively. The recording and reporting tools are responsible of collecting historical data for performance analysis.

In this paper we will explain mainly the Workflow Engine subcomponent in the CEMIS reference architecture of the project. It represents the heart of the system and provides a runtime environment for executing the system’s workflow instances. The main characteristics of this subcomponent are derived from the workflow reference model (Hollingsworth, 1995, p. 20) developed by the WfMC. In accordance with (Hollingsworth, 1995, p. 22), Table 1 summarizes the list of main functionalities covered by the workflow engine. These functionalities include the process definition interpretation, workflow instances controlling, navigation between workflow activities, work items identification, workflow data maintenance, external applications support, participant management, and last but not least the supervisory actions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Definition Interpretation</td>
<td>Interpretation of process models enabled following a specification of the used process description language and the management of control data.</td>
</tr>
<tr>
<td>Workflow Instances Controlling</td>
<td>Includes the control of workflow instances creation, activation, suspension, termination, etc.</td>
</tr>
<tr>
<td>Workflow Activity Navigation</td>
<td>Indicates the navigation between different workflow activities, which may involve sequential or parallel operations, deadline, scheduling, interpretation of workflow relevant data, etc.</td>
</tr>
<tr>
<td>Work items Identification</td>
<td>This functionality deals with the identification of work items for user attention and provides an interface to support user interactions.</td>
</tr>
<tr>
<td>Workflow Data Maintenance</td>
<td>Is responsible of workflow control data and workflow relevant data maintenance. This functionality enables the process of passing workflow relevant</td>
</tr>
</tbody>
</table>
data to/from applications or users.

| Support External Applications | The support of external applications is enabled by providing an interface to invoke external applications and link any workflow relevant data. |
| Participant Management        | Provides the sign-on and sign-off management of specific participants. |
| Supervisory Actions           | This functionality comprises supervisory actions for control, administration and audit purposes. |

As a conclusion, the system’s workflow engine is mainly responsible of executing set of workflow activities, sub-activities, or instances. Since some of the activities are realized as Web Services, the workflow engine connects via an interface with both the system’s service repository that is called the green service mall. Finally, the workflow engine interacts via the DAO interface with the system’s database that provides the required basic database functions required for workflow execution.

### 2.2 The IT-for-Green Platform Ansatz

Resource-friendly design of business processes and their energy- and material-efficient controlling task demands for sustainability oriented organizational structures as well as for proper ICT support. Contemporary CEMIS do not cope with requirements from the sustainability discussion. At the time of reporting environmental performance, it is too late to set the right course. Without an early identification of cause and effect to anticipate environmental impacts of decisions for timely intervention, potentials for acting precautious remain unemployed. In order to map the whole life cycle of a product, starting out from input (measuring energy efficiency of the used ICT and production schemes), going on with transformation processes (environmental conscious production, green logistics and sustainable product development) and ending up at the output stage (business communication and sustainability reporting) and beyond (disassembling planning), new workflow engines for next generation CEMIS are needed in support for

- Long product life, i.e. accompany a product through the whole life cycle
- Capability of flexibly integrating (almost) arbitrary activities
- Integration of overall CEMIS contexts for standardized information exchange and integration
- Capability of frequently interrupting and resuming execution at arbitrary breakpoints
- Integration of freely programmable control-flow structures
- Redefinition of transitions and states and integration of new activities at runtime (for long living workflows)

From a technical perspective, the realization of the services needed for fulfilling an activity is the main interest. This is so to speak the business logic of the whole system. For the sake of an individually composed software system that serves exactly the individual needs of different business and companies, we will have a set of interoperable services for individual tasks. With this modular design principle each company may compass its own tailor made CEMIS.

From the task perspective, a workflow represents an executable arrangement of different activities that guides a user during his work with the system. It will be possible to define (or rather program) complex relations and control flows within a workflow. A workflow may comprise activities from several (maybe external) sources and also from different users. On a time scale, a workflow might be executed in several phases (during product life-cycle) and be saved to disc in the meantime. In order to share information among all these services and phases, a specialized context for information exchange is shared.
3. SCXML Workflows with CEMIS Context

The core system, that will finally make up the next generation in the CEMIS, will be a service-oriented platform that allows for loose coupling and bundling of necessary functions; like linking pre-implemented functions in a programming language. A green service mall will provide a semantically enriched procurement of CEMIS-functionality for individual embedding into workflows or rather application definitions. Embedding environmental considerations into arbitrary (business) processes this way, allows for an intermixed usage of specific functions from self-hosted services, external service providers and non-environmental services. Such architecture allows for a highly flexible integration of environmental tasks into traditional (already installed) information infrastructures with the new CEMIS as the integrating system and runtime environment.

Integration will be achieved by coupling these services that encapsulate the actual CEMIS functions by having them called from within an application description that describes not just a simple orchestration of web services but rather a user interface together with interaction and issued events that are tied to different interaction elements. Each event may issue changes in control flow, execution of activities, changes (or updates) in the user interface, updates within the shared variables space (the context), etc.

In this way, we are going for a system, that allows for literally programming own CEMIS without limiting ourselves to a specific programming language on the one hand and that eliminates the need for in-depth programming skills when defining a CEMIS on the other. At the same time, a definition as a graph allows for the usage of visual modelling tools. At the same time, the use of a state chart representation with SCXML as definition language allows for interpretation at runtime without a need for a compiler environment. Each service method will have access via the context to the formatting scheme as well as to various converting utilities for proper data handling while still be able to post arbitrary, own (complex) data types. Because the data format will include a semantic as well as a content part (similar to the division of meta-data and content in HTML), annotations for proper interpretation might be included in the exchanged data. The SCXML description format for our workflows will prescribe to use the same format internally (DSL, scripting, etc.) for seamless integration. For import from and export to external software systems, a mediator for translating from one format into the other and vice versa may easily be integrated along the same scheme that integrates other functionality: by using web-services.

3.1 General Concept

The term SCXML refers to State Chart XML and is an XML based mark-up that allows control abstraction of an application by a state machine notation. Actually, SCXML is a general purpose event based state machine language (W3C 2012) to be used for example for high-level dialog control or as a multimodal control language. It is especially furthered for speech processing applications (VoiceXML, voice application meta-language, call-center management language, etc.) but also recommended as a general process control language for other purposes.

In the first line, SCXML is a standard for state chart descriptions. For execution, an environment capable of interpreting SCXML is necessary. We have chosen Commons SCXML as execution environment (http://commons.apache.org/scxml/) because of its lean implementation and the huge potential for own extensions and simple activity implementations. Commons SCXML abstracts out all execution environment interfaces allowing for - among others - the registration of custom actions, control flow actions, scripting languages or own extensions of the SCXML semantics, if needed.

In this way, workflows defined in SCXML may connect the results from different service invocations in a programmatic way by harnessing integrated scripting and/ or expression languages (e.g. Javascript or EL) using a variables space within a context that is shared among (or at least accessible by) all registered services as well as by the engine that may contribute to the function of a so build application.
From an interoperability point of view, clearly, data exchange and typification and method parametrization is an issue. For this purpose, we are going to integrate a common data exchange format based on XML representations into the IT-for-Green approach.

3.2 Activities

One major concept of a workflow system is the concept of an activity that encapsulates as a concept the actual functionality of the application described by the workflow.

![Figure 2: Main categories of activities within the IT-for-Green workflow system](image)

Usually, the basic functionality of a workflow is encapsulated within the implementation activities that are called from the engine during workflow execution. Figure 2 shows the main hierarchy of activities (including their general execution sequence and behaviour) that are incorporated in our system.

Other than mostly usual, our workflow definition allows for the integration of arbitrary programming language constructs or DSL elements that might likewise be integrated by a model based development approach. The main hierarchy as shown in figure 2 groups the activities as follows:

**Engine-internal actions**

Internal actions are the most basic sort of activity. These control the internal behaviour of the workflow engine and do not have any outbound effect. Examples are: The initialization of variables within the context, internal calculations, firing of internal events, etc. These are usually treated as atomic and executed once at a time.

**Simple method call**

The method call should be understood as a synonym for several types of procedure calls: simple method calls, remote method invocation, several types of web service invocations (we are going to support at least SOAP and Hessian services) as well as calls to internal (available from plug-in libraries) auxiliary methods (a sort of toolkit for basic everyday tasks). These will be implemented as synchronous calls and will not depend on any user interface elements.

**Asynchronous method call**

Of course, it cannot be guaranteed that each activity completes fast enough for the workflow engine to wait for a return of the method call. An example might be the gathering of sensor data for a longer time horizon. Activities from this group will therefore start the execution asynchronously. This special activity for services that need more time for execution is provided as a specialized service invocation activity that calls the respective method asynchronously. Asynchronously means that the calling method from the
WFMS does not wait for an answer but immediately returns and becomes idle (or free for other activities that may be done in parallel); no active waiting is necessary. On the other hand, this means that the service implementation works independently and has to put the result into the context before it signals a work done event to the WFMS. Writing the result to a communication bus would be an appropriate alternative. Our System will be prepared to incorporate different real time conditions into the execution. For example: if the result is not ready at a given due time, it will be discarded. It is the duty of the activity implementation to send a signal back to the workflow engine when work is finished.

**Activity with view integration**

Some functionality within the modules is likely to be rather complex, composed procedures. For this reason they will have a need for bringing their own user interface into place for proper interaction with the user and because such complex and versatile tasks may hardly be modelled with an external controlled user interface. Editing a sustainability report may serve as an example. An editor for such purpose will for sure be a too complex user interface to be ad hoc defined with the help of the workflow definition. Moreover, it would induce unnecessary additional and error prone work efforts if not pre-implemented and readily available as a component for use. The activity with view integration is the most complex type of activity and has already been described by Rapp et al. (2012) for the use case of energy aware product design.

**User activity**

A user activity incorporates the action of a real person. Such activities might be issued in different ways. The easiest form of a user activity would be the direct request of an input from the user in front of the screen during the course of workflow execution. This type can be embedded directly by defining the user interface e.g. as form for input acquisition within the workflow definition and by rendering it. The form would then send the input plus an identifying done event back to the workflow system. Another type of user interaction would be messaging some different user within a collaboration scenario to do something asynchronously. Such sort of interaction will require the other user to confirm his work.

**Language related activities**

Some specialized activities will be related to integrate programming structures such as structs for control flow (loops, if-then-else construct), memory access (transient and persistent shared data via context), etc. These activities will make up (together with own scripting extension) a customizable form of built in domain specific language.

### 3.3 Programming Abilities

The following code fragment shows a very simple example for the interplay of a user activity with a generated form for acquiring input and then post-processing this input by invoking a given service:

```xml
<state id="input">
  <onentry>
    <itfg:form var="addend">
      <input type="text" name="value" label="addend:" />
      <input type="submit" value="add" />
    </itfg:form>
  </onentry>
  <transition event="formSubmitted" target="add" />
</state>

<state id="add">
  <onentry>
    <itfg:serviceCall serviceID="addservice123" var="sum">
      <parameter name="a" value="${sum}" />
      <parameter name="b" value="${addend}" />
    </itfg:serviceCall>
    <itfg:optionPane text="Do you want to continue adding?">
      <itfg:option text="more" event="moreInput"/>
      <itfg:option text="done" event="calculationDone"/>
    </itfg:optionPane>
  </onentry>
  <transition event="moreInput" target="input" />
  <transition event="calculationDone" target="createChart" />
</state>
```

09.07.2012, YourName.doc
In this example two states are shown from a larger example (not fully shown here) that lets a user repeatedly enter values that are added up by a service. Depending on the users decisions the process is repeated or ends with the generation of a result chart. The input state issues as soon as entered the generation of an input dialog that is sent to the GUI. As in this state definition constituted, entering data will issue an event that is sent to the workflow engine and issues a transition to the add state that first invokes a math service (with parameters taken from the context by EL) and then generates a second form that enables the user to alter control flow.

4. Conclusion

The next generation of CEMIS demands for a way defining environmental management related applications on demand from an easily available and externally (by experts) updated toolbox of remote services bundled together with in-house processes and data. We presented our first consideration regarding a new way of implementing future CEMIS with the help of SCXML. Such definition will allow defining workflows as executable state machine definition with the ability of integrating different DSL and scripting languages as well as extensible mark-up definitions and activity descriptions. A standardized, common data format accompanies the workflow description. In this way, such a system will allow for an easy implementation by definition approach of future highly integrative and tailor made CEMIS systems.

Acknowledgement

This work is part of the project IT-for-Green (Next Generation CEMIS for Environmental, Energy and Resource Management). The IT-for-Green project is funded by the European regional development fund (grant number W/A III 80119242). The authors thank for the support.

Bibliography


