
Volker Wohlgemuth, Tobias Schnackenbeck, Matthias Mäusbacher, Dominik Panić
HTW Berlin, University of Applied Sciences, Department of Engineering II
Wilhelminenhofstr.75A, D-12459, Berlin, Germany
volker.wohlgemuth@htw-berlin.de
tobias.schnackenbeck@htw-berlin.de
dominik.panic@htw-berlin.de
m.maeusbacher@ifu.com

Abstract
This contribution describes concepts and implementation of an extendible tools platform for the development of Environmental Information Management Systems (EMIS), which is based on the open source software framework Empinia. Empinia eases the development of graphical rich client applications for the .NET environment based on plugins which can easily be extended. On top of Empinia the EMIS toolkit platform adds generic functionality for the EMIS application domain and forms a common structure for applications built to provide software assistance that addresses actual problems in the application field of EMIS. Software application developers benefit from this toolkit in a way that they can focus in a straightforward manner on close-to-the-problem functionality instead of having to spend much effort in the development of basic software infrastructure before.

1. Introduction
This paper describes the preliminary results and the potential of a research project at the University of Applied Science (HTW) in Berlin which is sponsored by the German Ministry for Education and Research (BMBF). The project aims to design an open source software application framework for the development of environmental management information systems (EMIS). Due to its high language independence and its widespread support Microsoft’s .NET-technology was chosen for software development (see Microsoft .NET). As a basis the plugin framework Empinia was developed that allows designing and developing dynamic architectures consisting of multiple different software components. In the following this paper presents the basic infrastructure of this plugin-framework and how to build EMIS applications based on the Empinia platform and the concepts of the EMIS toolkit approach.

2. Goal and Scope
The main intention of this project is to provide a flexible, easily extendible software platform to improve reusability and to reduce cost and complexity during the development of EMIS software. The core of this platform named Empinia (www.empinia.org) consists of a plugin infrastructure that allows to assemble applications dynamically using plugins provided by Empinia itself or by others. In particular Empinia uses the concept of extension points to offer the possibility to extend existing components of the framework with user-specific functionality. On top of the plugin infrastructure Empinia provides plugins which enable developers to build applications with a rich client user interface requiring minimum effort. The software concepts and patterns embodied in Empinia help to create well-designed plugins and applications.

Envisioning EMIS for a certain application field consisting of user-defined plugins, this project provides a component-based toolkit approach for the development of such application-specific EMIS. Using this toolkit, EMIS applications can be created by selecting and configuring plugins from a plugin pool. Methods and standards will be provided to assist the seamless and consistent collaboration of
the plugins. It is expected that an open-source community will evolve to continually extend the pool of plugins available to create EMIS.

As proof of concept an example EMIS application shows how to plug together different methods and techniques from the field of material flow management, discrete event simulation and visualization. This example application demonstrates how to integrate components using the modular conception of the underlying Empinia platform. It also uses a toolkit approach for building plugin-based EMIS. Up to now the components developed as part of the project focus on plugins providing simulation features, material flow management concepts and their overall integration to build user-specific EMIS for certain application fields.

3. Architecture and Ideas of Empinia

Because the EMIS toolkit implementation consists of various extensions for the Empinia framework, a short introduction to Empinia’s architecture will be given next. From an EMIS application point of view Empinia provides a software framework mainly dedicated to the development of (desktop) applications with a graphical user interface. To this technical fundament the EMIS toolkit adds common functionality for the EMIS application domain, resulting in a combined application framework specifically tailored to the needs of the EMIS application field. Isolating the EMIS functionality from Empinia’s technical base allows providing similar toolkits for completely different application domains, thus making the Empinia framework more versatile. It also reduces complexity for developers of EMIS extensions or applications because it follows the software design pattern separation of concerns.

At the core of Empinia lies a lightweight microkernel architecture providing the most basic infrastructure for managing the software components named plugins which constitute together the Empinia framework. This so-called Platform Runtime locates, instantiates and wires together the available plugins and allows registering user-specific plugin components. Inspired by the Eclipse OSGI implementation concepts (Eclipse OSGI) components which shall be extendible provide one or more extension points each with contracts defining extension possibilities. This extension-point technology allows to provide high flexibility by a solid system-architecture and a manageable expandability. The following figure 1 shows the relations between the different layers. On top of the basic infrastructure provided by Empinia there is the domain layer as first layer of our EMIS toolkit. This domain layer contains multiple distinct components for creating EMIS applications. Empinia is based on the .NET environment and currently written in C#. Extensions to the Empinia framework can be written in many other programming languages due to the language independency of the .NET technology.
4. Conceptual Design and Development of an EMIS Toolkit

4.1 Description and Objectives of an EMIS Toolkit

As would seem natural the EMIS toolkit consist of several plugins providing functionalities that are typically used in EMIS software and are grouped in corresponding fields of activity. But the EMIS toolkit does not only serve as a container for plugins. It also provides assistance functionality when a user wants to select an appropriate plugin for a specific task and guides the user through every single step of the particular workflow if desired.

The next chapter discusses these guidance and assistance mechanisms which are planned to come along with this toolkit. Subsequently some of the contained plugins are described briefly.

4.2 Assembling and Usage of an application with the EMIS Toolkit

The combination of multiple plugins for accomplishing specific EMIS related tasks is not as trivial as it might seem at first glance. This is rooted in the widespread functionalities and the complex workflow of such tasks. Thus offering a flattened hierarchy of plugins without any additional information regarding i.e. containing building blocks, intended application field etc. would not be a sufficient way of assisting a user in selecting plugins suitable for his specific requirements. Instead an assistance mechanism based on the workflow should be provided when selecting the corresponding plugins. A typical workflow may span modelling, parameterization, calculation, evaluation and more.

An appropriate guidance mechanism should start with prompting the user to define a workflow. The available work steps are defined by the plugins of the EMIS toolkit. Each plugin registers itself to one or more work steps (depends on the complexity of the plugin). Custom plugins can also register to any of the work steps and can even define new work steps. To simplify this process it will be possible to define pre-built workflows, so that repetitive composition of similar workflows will be avoided. After
composing or selecting the workflow the application displays applicable plugins for each work step. Each work step is configured individually and just the plugins supporting the current work step are displayed. The guidance also includes consideration of dependencies, so that necessary additional plugins will be considered automatically. Furthermore the guidance mechanism verifies consistency. Potentially plugins may be incompatible, especially custom plugins. In this case the user will be notified and the application offers an alternative if possible.

After configuring the workflow and selecting the particular plugins the user can process the workflow step by step. As an example the application may show up a graphical editor if modeling is the first step of the workflow. After the user walked through his model in this way the next step begins and each model component has to be configured. The configuration and parameterization can be assisted by querying key parameters, for example. This way the user is guided and assisted through each step of the workflow till he achieves his aim and the results of the task are shown. The implementation of this assistance resides in detail within a particular plugin, but the general mechanism for this assistance will be provided by the EMIS toolkit.

### 4.3 Available plugins for the EMIS toolkit

Up to now the following plugins have been developed as a base pool of specific components for the EMIS toolkit.

As a starting point the fields of discrete event simulation and material flow analysis have been covered. The plugins providing the simulation infrastructure contain general components for executing discrete event simulation runs. The discrete event simulation is a very common technique for analyzing operational production systems, but it is also widely used for planning purposes, especially when decision makers want to evaluate different alternatives of a production processes (see Law and Kelton 2000). Therefore, the plugins also provide components for specific domains, especially a production system domain providing components like workstation, entry point, buffer, conveyor, etc. (see Košturiak, and Gregor 1995).

The plugins for material flow analysis are still under development and mainly deal with material management and material accounting features. Many - if not most - EMIS need to manage information about materials which are used during the production process like energy demands, raw materials, waste or other supplies. The material management plugin allows creating, reading, updating and deleting such materials, which contain various properties and can be grouped. So it is possible to use a shared material management software plugin for different purposes. The material accounting plugin implies structures like in- and output lists and functions for material- and energy-bookkeeping.

Both fields, Simulation and Material Flow Analysis, have in common that they use a graphical editor for modelling purposes and the gathered results have to be displayed in an appropriate way and evaluated. Plugins providing this functionality are also available in the EMIS toolkit just as further plugins for required mathematical functionality in the context of EMIS (e.g. creating random numbers from a specified probability distribution etc.).

### 5. Example Application

Part of this project is as a proof of concept the development of a specialized EMIS, a material flow simulator, which uses (or will use) several components of the EMIS toolkit. In the first approach the components for discrete event simulation and its production system domain were integrated as mentioned in chapter 4.3. In the next step components like navigator- and dialog-factories from the Empinia framework are used to show basic information and to allow users to parameterize specific production system components, like workstations and entry points, as well as experiments.

Since the process of modelling is usually done in a graphical editor, the application was extended by a general Graph Editor (an existing extension of Empinia implemented in the course of the development of a specialized editor (see Joschko 2008) based on graph model components), which allows modelling of directed graphs. The Graph Editor itself provides extension points that define which objects are presented as nodes or edges in a given context and how to do that using basic framework me-
chanism. The last two steps are composed in a new plugin named Simulation.UI which consists of just a few classes. The running resulting application is shown in the figure 2.

After completing the discrete event simulation related tasks, the material flow coupling had to be established. Therefore the material-management and material-accounting components from the EMIS-toolkit have to be integrated. Since simulation components should not be mixed up with material flow techniques and vice versa, a new plugin providing functionality for event-based material accounting was developed. It integrates the more job-based economic view of simulation with the more material-oriented ecologic perspective of material flow analysis using on a combined methodical basis realized through an event oriented material- and energy-bookkeeping system (Wohlgemuth 2005, pages 222-224). Therefore each model component generates events according to the logic of the discrete event simulation engine. On implementation level this new plugin serves as a bridge between techniques of material flow analysis and simulation. On the one hand it registers itself as a simulation observer (via extension points provided by the simulation components) who gets messages each time a simulation event is handled by the simulation engine. On the other hand it uses the functions of the material-management- and material-accounting-components to configure and perform material and energy bookings.

To present and analyse the results of a material flow simulation run, two other components of the EMIS toolkit will be used: A component for reporting, which provides charts, diagrams, tables and configurable balance sheets, and a component for material flow analysis, which in this case is used to calculate material based key performance indicators. A plugin for statistical purposes has still to be developed and will be used to calculate statistical values such as confidence intervals.

Figure 2: Sample GUI of the Material Flow Simulator with Simulation Navigator, Graph Editor and Property Editor

6. Conclusions and Visions

The software components within the Empinia framework are an immense facility in developing software. The platform is a basis for custom developments of plugin-based software tools, where the existing source code can be extended or customized. The planned and already developed components of the EMIS toolkit benefit from this framework and provide plugins that can be used in developing EMIS
software. Additionally the toolkit provides assistance in selecting the appropriate plugins for a given EMIS related application context and even supports mechanisms to assist users in their work with the software.

At the time of the writing of this contribution some of the planned plugins for the EMIS toolkit are already completed and some more will follow. Based on these plugins an example application, a material flow simulator, has been developed as a proof of concept. The development of another application for waste management purpose for a small and medium-sized company is planned in the context of a master thesis and will start in early summer.

Due to the open source character of this research project we invite interested people to commit to the ongoing development of our EMIS toolkit. For information on that consider the web sites www.empinia.org or www.emporer.net.

Acknowledgement

This research project is funded by the German Federal Ministry of Education and Research (BMBF). The authors thank for the support.

References