Enabling Flexible Laboratory Processes: Designing the Laboratory Information System of the Future

Christoph Duelli, Robert Keller, Jonas Manderscheid, Andreas Manntz, Maximilian Röglinger, and Marco Schmidt

Abstract

(a) **Situation faced**: Recent developments in the medical and industrial laboratory market have increased the need for highly flexible laboratory processes. This pressure results from new requirements that have accompanied the internationalization of laboratories and the digitalization of paper-based, bureaucratic work practices. The execution of laboratory processes is supported by laboratory information systems (LISs), which handle the control and information flow of incoming orders end-to-end. State-of-the-art LISs do not feature sufficient flexibility-to-use and flexibility-to-change capabilities. To prepare medical and industrial laboratories for the challenges ahead, LISs require more advanced flexibility capabilities that meet the need for flexibility in complex laboratory processes.

(b) **Action taken**: To address the challenges of medical and industrial laboratories, MELOS, a leading German LIS provider, and the Project Group BISE of the Fraunhofer FIT conducted the *LIS4FUTURE* project. The project team compiled requirements on the flexibility of laboratory

C. Duelli • A. Manntz
MELOS GmbH, Franz-Beer-Str. 6, 86459 Gessertshausen, Germany
e-mail: duelli@melosgmbh.de; andreas.manntz@melosgmbh.de

R. Keller • M. Röglinger (✉)
Fraunhofer FIT, University of Bayreuth, Wittelsbacherring 10, 95444 Bayreuth, Germany
e-mail: robert.keller@fit.fraunhofer.de; maximilian.roeglinger@fit.fraunhofer.de

J. Manderscheid
FIM Research Center, University of Augsburg, Universitätsstr. 12, 86150 Augsburg, Germany
e-mail: jonas.manderscheid@fim-rc.de

M. Schmidt
Fraunhofer FIT, University of Augsburg, Universitätsstr. 12, 86150 Augsburg, Germany
e-mail: marco.schmidt@fit.fraunhofer.de

J. vom Brocke, J. Mendling (eds.), *Business Process Management Cases*,
Management for Professionals, DOI 10.1007/978-3-319-58307-5_19

processes and derived corresponding requirements for the LIS's flexibility-to-use and flexibility-to-change. The lack of configuration capabilities and modularity across all layers of the software architecture was identified as a major inhibitor of flexible laboratory processes. Following an agile development process and grounded on extant knowledge, the project team developed the *LIS4FUTURE* demonstrator, a process-aware LIS with a modular architecture and a rule-based configuration mechanism.

(c) **Results achieved**: Based on identified requirements, the project team iteratively developed and evaluated the modular architecture and the rule-based configuration mechanism as part of the development of the *LIS4FUTURE* demonstrator. The modular architecture allows for the complete replacement of process steps at build time, while the rule-based configuration mechanism makes it possible to meet the ever-increasing demands for flexibility at runtime. The *LIS4FUTURE* demonstrator, which shows the applicability of the developed concepts in real-world scenarios, will help MELOS develop an innovative release of their LIS.

(d) **Lessons learned**: During the *LIS4FUTURE* project, the project team learned that: (1) advanced flexibility-to-use and flexibility-to-change IS capabilities are needed to prepare for flexibility demands on the process level; (2) radical redesign of existing processes and systems should be preferred over incremental improvement in order to tap the disruptive potential of innovation opportunities; (3) the LIS architecture must be aligned with the process paradigm if it is to be flexible; (4) discussions among academics and practitioners are more effective if they are based on running prototypes rather than on theoretical concepts; and (5) project results improve if project team members work a substantial fraction of their time at the same location.

1 Introduction

Flexibility has become an increasingly desirable corporate capability, particularly in the services industry, which is the largest and most rapidly growing business sector in many industrial nations (Fitzsimmons and Fitzsimmons 2013). In search of an optimal level of flexibility, insurance companies, for instance, must continually balance the benefits of automated and standardized and manual and flexible claims-handling processes.

In medical and industrial laboratories, daily operations may use a few highly individual samples or tens of thousands of standardized samples. A typical laboratory process starts with an order entry, such as one that uses a blood sample. The order requests that an examination, such as a hemogram, be performed on the sample. The examination results in a diagnostic interpretation, which must be validated by a physician before being sent back to the requesting physician. Finally, the process ends with accounting of and billing for the order. Despite this simple sequence,

laboratory processes have considerable need for flexibility because of their content- and market-related complexity. From a content perspective, there are many process variations and exceptions (e.g., interdependencies of an examination or test within a single process instance or across multiple instances), as well as country-specific regulations that change regularly. From the market perspective, a high level of cost pressure leads to the laboratory market's increasing consolidation and globalization. Therefore, medical and industrial laboratories tend to expand their service offerings in order to realize economies of scale via, for example, mergers and acquisition. Laboratories also explore new market segments, offer new services (e.g., for human, veterinary, environmental, hygiene, or microbiological purposes), or follow the trend toward digitization (e.g., eHealth or mHealth) in providing advanced graphic diagnostics or keeping electronic health records. In short, the future of medical and industrial laboratories will be *connected*, *diverse*, *complex*, *and fast*, so laboratory processes need a highly flexible process-enactment infrastructure (van der Aalst 2013), commonly referred to as a laboratory information system (LIS).

In automating processes with a significant need for flexibility, a firm must consider business process flexibility and information systems (IS) flexibility jointly. Business process flexibility refers to volume and functional flexibility (Afflerbach et al. 2014), where volume flexibility, a partial abstract of installed capacity, helps a firm cope with risky demand, and functional flexibility helps the firm execute process variants and create even unplanned process output to satisfy customers' needs. While volume flexibility has been researched primarily from a capability and revenue-management perspective, functional flexibility has a rich tradition in business process management (BPM) (Schonenberg et al. 2008). One way to achieve functional process flexibility is to use flexible process-aware IS (Reichert and Weber 2012). IS flexibility can be split into flexibility-to-use and flexibility-to-change (Gebauer and Schober 2006), where flexibility-to-use refers to process requirements that can be supported without requiring major changes in the IS that underpins the process, and flexibility-to-change refers to the ability to extend IS to remain aligned with changing process requirements.

Against this background, LIS must evolve into flexible process-aware IS. In particular, there is a pressing need for an integrated and innovative approach that allows for the configuration and modularization of all software architecture layers, including data management, application logic, control flow management, and user interaction at both runtime (flexibility-to-use) and build time (flexibility-to-change). Because of extensive user-specific configurations and regulations, a LIS also requires capabilities related to efficient development and maintenance (flexibility-to-change).

To contribute to enhancing existing LISs so they are more flexible process-aware IS, collaboration in the research project Laboratory Information Systems for the Future (*LIS4FUTURE*) between September 2014 and October 2016 was undertaken by the Fraunhofer Institute for Applied Information Technology (FIT), a research institute that is experienced in the development of custom-tailored applications; its project group, Business and Information Systems Engineering (BISE), which specializes in BPM; and MELOS, a technological leader in the field of medical

and industrial laboratory software solutions. Using the MELOS LIS as an example, the partners designed the *LIS4FUTURE*, a process-aware LIS with a modular architecture and a rule-based configuration mechanism that facilitates laboratory processes' functional flexibility. The *LIS4FUTURE* demonstrator exemplarily implements the design and architecture with respect to the accounting step of the laboratory process in order to verify the theoretical concepts' applicability and usefulness. The *LIS4FUTURE* project was funded by the Bavarian Ministry of Economic Affairs and Media, Energy and Technology's R&D program *Information and Communication Technology Bavaria*. The *LIS4FUTURE* project relates to the "process implementation and execution" capability area of the IT factor discussed in Rosemann and vom Brocke's (2015) six core elements of BPM.

2 Situation Faced

Before sketching the situation that leads to highly increased flexibility requirements for laboratory processes and LIS, we look at medical and industrial laboratories in terms of the laboratory process and the current market situation. We also provide information about the MELOS LIS that served as one of many reference points and as an example throughout the *LIS4FUTURE* project.

The laboratory process consists of all steps from order entry to accounting of the laboratory service (Fig. 1). After an order is made, the samples to be analyzed and the specification of the required examinations arrive at the laboratory. An order could request a hemogram involving a blood sample or an investigation for cancer in a tissue sample. Based on this information and enriched with customer-specific master data, the laboratory analyzes and tests the samples and summarizes all test results in a single report per order. A laboratory physician then validates the results, checks the report for plausibility, and adds further diagnostic information if needed before the laboratory transfers the validated results back to the customer. Finally, the laboratory charges for the services provided in line with current price lists and regulations.

Many content- and market-related factors affect the nature of the laboratory process. The BPM context framework from vom Brocke et al. (2016) offers guidance in characterizing relevant contextual factors, particularly the factors of repetitiveness, knowledge-intensity, interdependence, and variability.

Laboratories receive an average of about 20,000 samples per day, with three to four required examinations per sample. Because of the high number of orders and the standard examinations that are available, the laboratory process is generally characterized by *repetitiveness*. The processing of samples is highly standardized

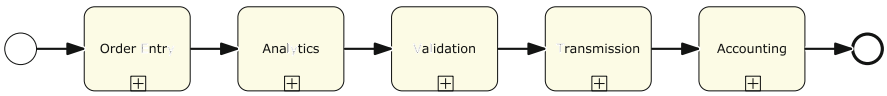

Fig. 1 Overview of the laboratory process

and automated. (E.g., laboratory devices analyze most samples automatically, sometimes leading to follow-up examinations based on pre-defined business rules.) The accounting of laboratory services also follows complex but strict guidelines, so there is little need for manual interaction and creativity. Pattern-detection mechanisms support or replace physicians' work in validating diagnoses. For common cases, physicians formulate rules that automate the process of checking plausibility, enabling them to spend more time on complex cases. Nevertheless, the diagnosis and device setup require domain-specific knowledge, so parts of the laboratory process are characterized by *knowledge-intensity*. As each order focuses on distinct samples, customers, and examinations, there is almost no *interdependence* on the process level. Cross-references need to be considered only for accounting purposes, when examinations are summarized on a quarterly basis and cumulative regulatory provisions apply.

Regarding *variability*, the laboratory process is a routine or runner/repeater process (Lillrank 2003; Johnston et al. 2012). The comparatively simple laboratory process includes an arbitrary but fixed number of variants and defined outputs. Variants of the laboratory process must be specified at design time so samples can be analyzed and orders can be billed. When orders are paper-based, the laboratory process includes an OCR scan, but whether a report is paper-based or not, urgent examination results can be reported immediately via fax or phone. In addition to industry-scale laboratories, specialized laboratories handle a small number of orders and place considerable effort into each order. Such laboratories run the risk that changes in routine or runner/repeater process instances become semi-structured or unstructured problems, referred to as non-routine or stranger instances (Johnston et al. 2012; Lillrank 2003). The processing of such instances requires functional process flexibility, as the following example ("the Example" hereafter) shows:

> A Spanish tourist in France is infected with Salmonella. She appears in person at the medical laboratory for blood sampling, as is usual in Southern Europe. Based on a cooperative arrangement with other laboratories, the French laboratory transfers the sample to a German laboratory close to the border for analytical and diagnostic purposes. However, since the sample was taken in France and it involves a Spanish tourist, both French and Spanish regulations and legal requirements apply. As a consequence, French regulations require that the German laboratory store detailed information on the blood sample (e.g., the place where the sample was taken and the distance from there to the laboratory), and Spanish regulations require that the German laboratory store the patient's health insurance data. In addition, the invoice is split among the patient, her employer, and her health insurances in Spain and France. In Germany, the accounting distinguishes only between private and legal insurance. If the patient were from Austria, the laboratory would also have been required to check the information stored on the patient's electronic healthcare card online.

The current situation in the medical and industrial laboratory market has pushed LIS providers to redesign their systems substantially in order to enable flexibility. For examples, with blurring national boundaries, as illustrated in the Example, medical and industrial laboratories must consider increasing numbers of country-specific regulations that increase complexity and their processes' need for

flexibility. In addition, national authorities like *Kassenärztliche Bundesvereinigung* (National Association of Statutory Health Insurance Physicians) in Germany and the *Centre National de Dépôts et d'Agrément de l'Assurance Maladie* in France update price lists and thresholds for medical values at least once per quarter. This complication becomes even more complex with laboratory mergers that occur because of the competitive environment and cost pressures, as when a laboratory's subsidiaries are located in multiple countries, the LIS must comply with all national regulations. Further, the configuration mechanisms implemented in state-of-the-art LISs are geared primarily toward trained experts with strong backgrounds in computer science, but future demands will come from people with non-technical backgrounds (e.g., physicians) who are expected to use a laboratory's LIS. Therefore, among many other topics, future LISs will have to focus much more on convenient user interfaces and a transparent representation of configuration capabilities that non-technical people can understand. Despite the high number of variants in the laboratory process, most of these variants are substantially constant and require only one-time configuration. One notable exception is the price calculation in the accounting step of the laboratory process, which is subject to frequent changes. Future LISs must be able to implement changes in the laboratory process and changes that affect the accounting logic without touching the system's implementation, such as the system's database and software architecture. This capability can be achieved by leveraging metadata management, a modular architecture, and rule-based configuration mechanisms. These examples provide an impression of the overall need for improvement that LISs must implement in order to meet laboratory processes' need for flexibility.

From a technical perspective, LISs offer support for the control and information flow of laboratory orders end-to-end, so they must be aligned with the steps of the laboratory process. To cope with the high and varying number of orders per day, the order-processing must be highly industrialized, with a high level of automation and interfaces that enable automated communication with external IS and technical devices like CRM systems, medical devices, and specialist software. In response to laboratory processes' increasing flexibility demands, LISs must ingrain highly integrated flexibility-to-use and flexibility-to-change capabilities in all layers of the software architecture and in all software modules (Gebauer and Schober 2006) to enable functional flexibility of laboratory processes in both the short term and the long term. The foundations of flexibility-to-use and flexibility-to-change are rule-based configuration mechanisms, modular software architectures, and efficient development and maintenance concepts. A module is a self-contained unit that includes data, business, and processing rules and, in some cases, a user interface. Each module should be as independent from other modules as possible so it is configurable by a specific configuration mechanism that is based on rules stored as master data of the LIS. Laboratories should be able to customize these rules without having to recompile the LIS.

Although MELOS, with its current generation of LIS, is among the leading LIS providers in Germany, it has to prepare its LIS for a *connected*, *diverse*, *complex*, *and fast* business environment and the related flexibility demands. The MELOS LIS

features powerful mechanisms for configuring and individualizing the IT support for complex laboratory processes. Its software architecture enables profound changes to the information and control flow and provides configurable interfaces for both users and technical equipment. However, these flexibility capabilities do not go far enough to meet the future flexibility demands on LISs.

Currently available LISs are too inflexible to cover laboratory processes' future demands for flexibility, so MELOS and the project group BISE of Fraunhofer FIT initiated their collaboration in the *LIS4FUTURE* project to develop a process-aware LIS with a modular software architecture and a rule-based configuration mechanism.

3 Action Taken

To enable functional flexibility of laboratory processes, the *LIS4FUTURE* project team designed and implemented a process-aware LIS into which are integrated a modular architecture and a rule-based configuration mechanism. The project team iteratively developed the *LIS4FUTURE* demonstrator following an agile software-development process in order to respond quickly to changes from newly identified requirements (Beck et al. 2001). The team used the *LIS4FUTURE* demonstrator to validate the developed concepts' applicability in real-world scenarios. The *LIS4FUTURE* project was comprised of four major phases: (1) requirements engineering, (2) design of the process-aware LIS with a modular architecture and a rule-based configuration mechanism, (3) implementation, and (4) validation of these concepts by evaluating the *LIS4FUTURE* demonstrator (Fig. 2). These phases were conducted iteratively and in an interleaving manner following the agile software development principles.

3.1 Phase 1: Engineering Requirements

In this phase of the project, the project team raised and structured requirements for the design of a process-aware LIS by analyzing the laboratory market, the state of the art in the related BPM and computer science literatures, and the architecture and components of the MELOS LIS. To take a comprehensive perspective, the project

Fig. 2 Plan of the LIS4FUTURE project

team used a multi-method approach to gathering requirements: The team first conducted interviews with industry experts and examined the extant LIS-related scientific literature to collect external demands and ideas for solutions regarding flexibility-to-use and flexibility-to-change. The literature review included work on software architectures, modular data models, software configuration and customizing, rule processing, and declarative business process modelling. Then the team drew from the expertise of the MELOS software developers inside and outside of the project team in order to acquaint themselves with relevant design decisions and to identify the strengths and weaknesses of the MELOS LIS. A detailed analysis of LIS-related use cases identified during a series of workshops led to a broad coverage of possible process variants and process-related requirements. Next, the project team technically analyzed the inputs and outputs of the MELOS LIS, which yielded structural information about the system's functionality without the team's having to review the code itself, an endeavor that would have been much more complicated because of the dominant monolithic software architecture and code structure.

The insights gathered during these steps were aggregated to compile the relevant requirements regarding the modular software architecture and the rule-based configuration mechanism. Some external requirements emerged from the laboratory market and the laboratory process (discussed in Sect. 2), and some requirements emerged from system-internal—that is, functional and technological—boundary conditions of the current MELOS LIS, which already incorporated basic configuration capabilities. These requirements resulted from lessons learned during past efforts or from the technological continuity undertaken to reduce maintenance costs. Examples are the LIS-wide applicability of a single configuration mechanism and the protection of live data from user-defined configuration settings.

3.2 Phase 2: Designing the Process-Aware LIS

Given the requirements identified in the project's first phase, the *LIS4FUTURE* team designed a process-aware LIS that enables advanced flexibility-to-use and flexibility-to-change. The *LIS4FUTURE* team consolidated scientific literature and collected insights on their practical implementations in order to review existing approaches to flexibility-to-use and flexibility-to-change. We found the state of the art of process flexibility from the BPM domain and that related to software configuration in general to provide no ready-made solution that could be directly adapted to the case at hand. However, this review yielded promising approaches, such as the idea of modular software architectures (Sulivan et al. 2001) to enable flexibility-to-change and the use of domain-specific languages (Mernik et al. 2005) for implementing flexibility-to-use in terms of rule-based configuration.

Before flexibility-to-use can be considered, flexibility-to-change must build a solid core, enabling developers to adjust the laboratory of the future quickly to emerging requirements. The modular software architecture must implement

flexibility-to-change capabilities, enabling developers to adjust the process sequence by easily reconfiguring existing interfaces and module interdependencies. With their own experiences in mind and influenced by the scientific approaches, the developers designed and refined a fundamental modular software architecture on which the future LIS can be based. A thorough analysis of the MELOS LIS's existing processes pointed to modules required in the *LIS4FUTURE* architecture. However, the concept of modular software architectures fundamentally differs from the MELOS LIS's monolithic structure, which is not process-aware. In general, modular software architectures are well established in computer science and are based on largely independent modules that operate on a determined set of input parameters and compile a predefined output. When modules call one another, a hierarchical or network-like structure emerges. By standardizing the communication among modules using contracts that define the information flow and reduce dependencies among modules, the project team boosted the modules' interchangeability.

This solid core paved the way for the integration of flexibility-to-use—that is, the independent configuration of single modules at runtime. Although this approach is not completely new to LISs, the targeted extent of configurability was unprecedented. Current configuration capabilities included various rule-based languages that are limited to single LIS modules. Each of these languages have a unique syntax and encompass over 1000 different operators that perform highly specific tasks within the respective domain. Because of limited rule-based configurability, the LIS also differentiates many special cases within the code base, a circumstance that hampers easy maintenance and code transparency. Discussions in the project team and with industry experts emphasized the need for a configuration mechanism that allows emerging flexibility demands to be implemented across module boundaries at runtime, facilitating the future daily use and the further development of the LIS.

The designed configuration mechanism builds on only two components: rules and plug-ins. Rules enable users to change and influence the fine-grained process flow of the laboratory process, while plug-ins can be provided only by developers, enabling the implementation of complex requirements that cannot be addressed by user-written rules. Rule-based configuration focuses on the demands of users who are working with the LIS on a daily basis and need to be able to adapt the LIS easily to their needs. A typical LIS contains more than 2000 rules, of which about 600 rules describe exceptions in the pricing process in the accounting module, such as: "A specific examination can be accounted only twice per quarter of the year for each patient," "The sum of the prices of several examinations is limited to a maximum threshold of, e.g., 20€ per patient and order," and "The accounting of an examination prohibits the accounting of another examination in the same order and for the same patient."

For some examinations, several rules may apply simultaneously, leading to dependencies among rules that can result in unforeseeable behavior. Therefore, the project team refined the configuration mechanism to support the writing of rules with verification capabilities that ensure correct rule-processing. Besides validating

syntax and semantics, the verification detects and warns about possible conflicts and rule interdependencies by statically evaluating rules without the need to access real data. Consequently, this mechanism enables users to monitor flexibility-to-use in the *LIS4FUTURE* demonstrator and facilitates the identification of undesirable behavior in the module in advance.

The *LIS4FUTURE* project team operationalized the rule-based configuration by providing a central rule parsing and processing module that evaluates each rule syntactically and semantically before initiating it. To enable the ex-ante verification of rules, the syntax is based on imperative programming languages like JavaScript to capture the modification algorithmically instead of using individual operators for each task. We designed the functional modules to allow module-specific rule-execution routines to be integrated seamlessly, operating on a domain-specific data model that limits the rules' access to a distinct subset of readable and writable data and prevents the mechanism from altering live data.

Configuration demands that go beyond the rule mechanism's capabilities can be supplied to users as plug-ins written by developers upon request. These plug-ins can attach automatically to mounting points within or between modules. Since mounting plug-ins has been a concept in software development for years, we refrain from providing details here.

In sum, the developed concepts implemented in the *LIS4FUTURE* demonstrator enable flexibility-to-change through a modular software architecture so LIS developers can change the laboratory process if needed. In addition, flexibility-to-change is extended by the ability to write and mount plug-ins that can enhance the LIS's functionality based on customer requests. Flexibility-to-use is integrated by means of a verifiable, rule-based configuration mechanism, providing users with a straightforward rule language to adapt future LIS.

3.3 Phase 3 and 4: Developing and Evaluating the Demonstrator

The project team implemented and refined the *LIS4FUTURE* demonstrator in the course of an agile software development process. The *LIS4FUTURE* demonstrator focuses on implementing the essentials of the modular software architecture and those of the rule-based configuration mechanism while enabling the developed concepts' applicability to be validated based on real data. On the process level, the *LIS4FUTURE* demonstrator focused on the accounting step and on the accounting module's interplay with other modules. This focus was reasonable, as the accounting module is the most complex part of the LIS and it is subject to the most burdensome customer requests and flexibility demands. In fact, all identified requirements on flexibility-to-use and flexibility-to-change could be checked using the accounting module as example. However, the monolithic architecture and manifold interdependencies among existing modules aggravated the refactoring of the existing accounting module and the integration of the new concepts, so the *LIS4FUTURE* demonstrator was implemented from scratch, which allowed us to

incorporate modern software engineering concepts, such as modules with clearly defined interfaces, dependency injection, and unit tests.

Once we familiarized the MELOS developers with these modern software engineering concepts, the development started in a Scrum-like agile development procedure that involved defining features that could be implemented during the next sprint, which we set to 2-week cycles. The wording of features' definitions, which are referred to as "user stories" in Scrum, emphasized the user-oriented benefit that comes with their attainment. The results of each sprint were reviewed at the end of each sprint, and adjustments to the backlog (i.e., the user stories to be processed) were discussed when planning the next sprint. This method facilitated the stepwise integration of the accounting module's functionality and the configuration mechanism's and modular software architecture's validation based on the use cases collected in the project's first phase.

The team frequently discussed the project's overall progress and the *LIS4FUTURE* demonstrator with project-external stakeholders like MELOS employees who were not involved in the project, laboratories that employed the MELOS LIS, and independent industry experts. Feedback from these stakeholders helped to improve the modular software architecture and the rule-based configuration mechanism in iterative cycles. Exemplary feedback was that LISs available on the market lack the ability to track the price calculations back to specific rules. This feature was not part of the initial backlog, but it was included after the preliminary *LIS4FUTURE* demonstrator was presented to external stakeholders for feedback. Thus, the project team extended the demonstrator with a price-tracing module that makes the price-calculation logic transparent to users. Although the price-tracer is currently limited to the accounting module, traceability and advanced logging of process executions can be applied easily to other parts of the process-aware LIS.

Figure 3 illustrates relevant parts of the modular software architecture's implementation of the accounting module in the *LIS4FUTURE* demonstrator, although the figure simplifies the architecture for communication purposes and abstracts from interfaces to other modules beyond the accounting domain, as well as from multiple rule-execution routines that are linked to one component. The core module is the *Quarterly Accounting* module, which calls the *Order Price Computation* module to account the order stack and initiates the price computation of single examinations (*Examination Price Computation*). Each of these steps provides additional *Rule Execution* modules, enabling the configuration of underlying processes. Each step also allows for the mounting of plug-ins, such as plug-ins to implement distance-dependent shipping costs. Finally, the customer is billed using the *Invoice Export* module. The architecture also includes the *Price Calculation Tracer* mentioned above.

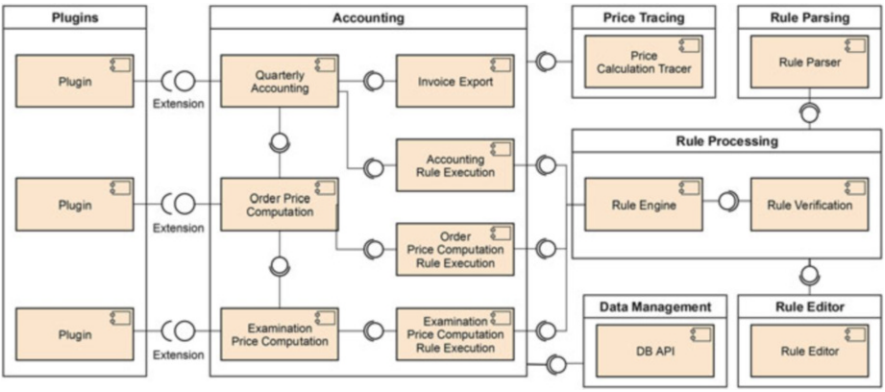

Fig. 3 Architecture of the LIS4FUTURE demonstrator's accounting module

4 Results Achieved

The actions taken in the *LIS4FUTURE* project resulted in the design of a process-aware LIS, prototypically realized in terms of the *LIS4FUTURE* demonstrator. As a preparatory task, we reviewed the need for flexibility in complex laboratory processes and extracted requirements for technological support by a process-aware LIS, considering content-related and market-related context factors. Then we focused on two major deliverables: (1) a modular and process-aware software architecture with largely independent modules and (2) a rule-based configuration mechanism that enables laboratory employees to customize by changing the LIS without recompilation and redeployment. The *LIS4FUTURE* demonstrator verified the developed concepts and confirmed their practical applicability.

4.1 Modular Architecture

To incorporate flexibility-to-change capabilities in future LIS, we designed a modular software architecture that facilitates the LIS provider's ability to add new functionality easily via modules. On the architectural level, modules can be added or replaced with significantly reduced effort. Although new modules still require the LIS to be partially recompiled, existing modules can be activated or deactivated during build time. The use of dependency injection as a software design pattern also helps to resolve dependencies between modules. As a result, the communication between two modules is managed by a third component (the *injector*). Modules need only a well-defined interface and a service level agreement that specify the communication and interaction standards, and other modules can use this interface to communicate with one another. In doing so, a nearly infinite combination of modules is possible to define new paths in the laboratory process.

For instance, referring to the Example, a French accounting module could easily replace or enhance German accounting functionalities.

We also modularized the LIS's support modules. In order to abstract data management from the real data source and to allow the data bases to be interchangeable, we added an intermediate data layer that provides defined interfaces for communication with databases and that can be adapted easily to new data sources. We also logically separated the module's configuration mechanism from the syntactical parsing and domain-specific processing of rules, which is enabled via a LIS-wide rule engine (Fig. 3). This measure also contributes to efficient further development and maintenance.

As a positive side effect, the modular software architecture is not restricted to the laboratory process but can be extended easily to other supportive LIS functionalities. For instance, we developed a rule-editor module to facilitate users' ability to configure the LIS directly in laboratories. In fact, this editor is an autonomous application, but it uses shared functionality in publishing syntactical keywords, which reduces future maintenance efforts.

4.2 Configuration Mechanism

We designed a rule-based configuration mechanism to support ongoing process adaption and LIS adaptation through flexibility-to-use capabilities. The configuration mechanism covers most of the laboratory process's flexibility requirements in terms of routing and calculation decisions, based inter alia on examination results and price lists. Consisting of rules and plug-ins, the configuration mechanism provides software developers and LIS users alike with a high level of customizability.

Rules enable LIS users to modify the process definition at runtime. Based on this foundation, all or selected currently running process instances can be migrated to a new process definition that is, for instance, requested by an external stakeholder. (E.g., physicians can ask for discounts or divergent processing in case of certain diagnostic findings.) Rules are clustered in collections based on the specific type of rule and can be enriched by metadata. In contrast to the existing rule languages, the new mechanism benefits from its own database that is built exclusively for the rule execution at runtime and that consolidates configuration capabilities across module boundaries.

We developed the rule-editor module to prevent the downside of user customization (e.g., higher risk of error). This module supports users using syntactical and semantical rule checks. As the probability that rules will interact increases with the number of rules in the LIS, we also integrated into the LIS a rule-checker that reduces the risk of unpredictable and unintended behavior that is due to disregarded sequence dependencies. This component enables the configuration of rules to be verified and controlled and ensures their compatibility. Overall, the measures taken significantly strengthen data security and increase the transparency of both process

design and execution. The rule mechanism that is extended by the editing support reduces complexity and facilitates the refactoring of existing rule bases.

The configuration mechanism also includes plug-ins to add functionality that simple rules cannot cover. The plug-in concept replaces programs that have unpredictable effects when data is manipulated and that address highly complex, very specific, or seldom-used functionality that exceeds the LIS's core functionality (e.g., route optimization for sample collection by a laboratory's customers). The technical integration of plug-ins into the LIS is also based on well-defined interfaces and specific mounting points. Nevertheless, as plug-ins are not part of the LIS's core functionality, they add functionality without the need to recompile the entire LIS. Therefore, they are located at the intersection of flexibility-to-use and flexibility-to-change.

4.3 Summary

The modular software architecture and the rule-based configuration mechanism enable the future LIS generation to be highly customizable. Their practical application was confirmed by the *LIS4FUTURE* demonstrator. Whereas the modular architecture focuses on flexibility-to-change by allowing for the insertion or the replacement of modules, the configuration mechanism provides flexibility-to-use by enabling rules to be adapted and plug-ins to be added. Together, these two concepts help to address future requirements regarding the functional flexibility of laboratory processes by reducing the customization effort in daily business operations and facilitating procedural and technological innovation. Accordingly, LIS providers and laboratories can react to content-related and market-related context factors with a manageable level of effort.

5 Lessons Learned

To meet the upcoming flexibility requirements of laboratory processes, the project team developed and implemented the *LIS4FUTURE* demonstrator, a process-aware LIS with a modular architecture and a rule-based configuration mechanism. In so doing, the project team had first-hand experiences that can be classified into process-specific, architectural, and organizational lessons learned. We share the most important of these lessons from our perspective.

5.1 Lessons Learned from the Process Perspective

Lesson 1: Rely on both flexibility-to-use and flexibility-to-change IS capabilities to prepare for future flexibility requirements on the process level.

As illustrated in our laboratory market example in Sect. 2 (the Example), new requirements for process flexibility based on such conditions as new regulations,

environmental changes, and other unforeseeable factors can pop up anytime and anywhere. In most cases, new requirements refer to process steps that are part of the extant configuration, but this assumption does not hold true in all cases, as the Example illustrates by highlighting the importance of multi-country support. Demanding requirements can even require inserting additional process steps or eliminating existing ones, a circumstance that usually requires significant changes in the underpinning information technology (IT) systems. We propose to implement flexibility-to-use and flexibility-to-change IS capabilities in order to enable the user to react easily to future changes and to consider flexibility at the time a process-aware LIS is designed.

Lesson 2: Incremental improvement is not always sufficient to achieve a target.

A well-known principle is that employees should question current work practices and refrain from excuses like "that's what we have always done." People are often hesitant to think in terms of radical process reengineering, getting lost instead in best practices, incremental improvements, and local optimization. The *LIS4FUTURE* project radically redesigned the software architecture and many modules of the MELOS LIS, particularly the accounting module. This radical redesign was rewarded with a significant increase in flexibility-to-use and flexibility-to-change as a result of replacing old, inefficient modules that have been grown historically and incrementally adapted to changing requirements. Radically rethinking existing modules and the architecture opened up completely new opportunities.

5.2 Lessons Learned from the Architecture Perspective

Lesson 3: The software architecture must be aligned with process thinking.

LISs that automate large parts of laboratory processes were once extremely complex. As we experienced in the *LIS4FUTURE* project, a significant part of this complexity stems from an inappropriate architecture. Following Stevenson and Pols' (2004), we dived deep into the MELOS LIS's software architecture to find a monolithic architecture that made even small adjustments highly complex. Therefore, we designed a modular and (in particular) process-aware architecture that substantially decreased the effort required in implementing changes and increased the potential of long-term-oriented flexibility-to-change, such as the replacement of entire modules (e.g., country-specific accounting mechanisms).

5.3 Lessons Learned from the Organizational Perspective

Lesson 4: Discussions among academics and practitioners are more effective if they build on running prototypes instead of theoretical concepts.

Although the MELOS management and employees supported the *LIS4FUTURE* project, experienced software developers and architects were skeptical about radically rethinking, based on the latest academic insights, previously made

technological, architectural, and process-related design decisions. We explained this skepticism as being based on the recent rejection of some technologies and the well-known resistance-to-change phenomenon. Bad experiences, when changes that were due to novel requirements or developer initiatives led to enormously increased software complexity, intensified this resistance. There was also general uncertainty about whether academic insights can be useful in solving real-world problems like the design and implementation of a LIS. We learned that just talking about innovations like modularity and configurability on a conceptual level did not help to overcome the practitioners' skepticism. As a result, we switched to a discussion that was based on running code as an outcome of agile development sprints. This approach facilitated a much more constructive and effective discussion of essential ideas and targets. Moreover, theoretically promising but practicably infeasible solutions could be discarded much more quickly.

Lesson 5: If you want your team members to communicate, co-locate them.

In contrast to our initial plans, which intended team members to collaborate as a distributed team, the project team decided to work at the same location to foster informal communication among the academic and industrial team members. Since all MELOS developers worked at the same location anyway and were not familiar with distributed work environments, this measure significantly helped the project team get to know each other and to give feedback more directly and openly. The collaboration also improved in terms of the assessment of each other's strengths and weaknesses, which allowed us to anticipate realistically the outcome of work packages like sprints in our agile software development process. Based on our experience in the *LIS4FUTURE* project, we recommend that project teams share a common work location, at least during the project's setup phase.

References

Afflerbach, P., Kastner, G., Krause, F., & Röglinger, M. (2014). The business value of process flexibility – An optimization model and its application in the service sector. *Business & Information Systems Engineering, 6*(4), 203–214.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland J., & Thomas, D. (2001). *Manifesto for Agile software development*. Accessed July 23, 2016, from http://agilemanifesto.org/

Fitzsimmons, J., & Fitzsimmons, M. (2013). *Service management: Operations, strategy, information technology* (8th ed.). New York: McGraw-Hill.

Gebauer, J., & Schober, F. (2006). Information system flexibility and cost efficiency of business processes. *Journal of the Association for Information Systems, 7*(3), 122–147.

Johnston, R., Clark, G., & Shulver, M. (2012). *Service operations management*. Essex: Prentice Hall.

Lillrank, P. (2003). The quality of standard, routine and nonroutine processes. *Organization Studies, 24*(2), 215–233.

Mernik, M., Heering, J., & Sloane, A. (2005). When and how to develop domain-specific languages. *ACM Computing Surveys, 37*(4), 316–344.

Reichert, M., & Weber, B. (2012). *Enabling flexibility in process-aware information systems: Changes, methods, technologies*. Berlin: Springer.

Rosemann, M., & vom Brocke, J. (2015). The six core elements of business process management. In J. vom Bnrocke & M. Rosemann (Eds.), *Handbook on business process management. Introduction, methods and information systems* (Vol. 1, 2nd ed., pp. 105–122). Berlin: Springer.

Schonenberg, H., Mans, R., Russell, N., Mulyar, N., & van der Aalst, W. M. P. (2008). Process flexibility: A survey of contemporary approaches. In J. Dietz & A. Albani (Eds.), *Advances in enterprise engineering I* (pp. 16–30). Berlin: Springer.

Stevenson, C., & Pols, A. (2004). An agile approach to a legacy system. In J. Eckstein & H. Baumeister (Eds.), *Extreme programming and agile processes in software engineering* (pp. 123–129). Berlin: Springer.

Sulivan, K., Griswold, W., Cai, Y., & Hallen, B. (2001). The structure and value of modularity in software design. *ACM SIGSOFT Software Engineering Notes, 26*(5), 99–108.

van der Aalst, W. M. P. (2013). Business process management: A comprehensive survey. *ISRN Software Engineering*. doi:10.1155/2013/507984.

vom Brocke, J., Zelt, S., & Schmiedel, T. (2016). On the role of context in business process management. *International Journal of Information Management, 36*(3), 486–495.

Christoph Duelli is head of development and joint member of the management at MELOS GmbH in Gessertshausen, specializing in software for medical and industrial laboratories for over 25 years. He holds a diploma in computer science from the Ulm University and worked with Prof. Reif's KIV (Karlsruhe Interactive Verifier) team to develop provably correct software at the University of Augsburg. Apart from teaching duties, he published several papers on this topic and contributed to the KIV development.

Robert Keller is a Research Assistant at the University of Bayreuth. His research centers around Data Processing and Management Platforms. He publishes his work in renowned journals and at international conferences. Furthermore, he is project manager at the Project Group Business & Information Systems Engineering of the Fraunhofer FIT and leads several publicly funded research projects as well as consulting projects in the context of Data Management, Project Management and eHealth.

Jonas Manderscheid holds a doctorate in Business & Information Systems Engineering from the University of Augsburg. He formerly studied Business Informatics (B.A.) and Information Management (M.Sc.), while also serving as IT Consultant at Hamm Reno Group GmbH. From 2012 to 2016, Jonas has been a research associate at the Research Center Finance & Information Management and the Project Group Business & Information Systems Engineering of the Fraunhofer FIT, also engaged in applied research projects with companies like MLP AG and MELOS GmbH. Most of his work centers around business process management (BPM), focusing on the monitoring and improvement phases of the BPM lifecycle. He has published in journals such as Decision Support Systems, Information Systems Frontiers, and Business Process Management Journal and presented his research at relevant international conferences. Currently, Jonas serves as Project Manager for Internet of Things Product Lifecycle Management at Hilti Befestigungstechnik AG.

Andreas Manntz is Managing Director at the MELOS GmbH in Gessertshausen, one of the leading German providers for software for medical and industrial laboratories over the last 25 years. He holds a diploma in Business Administration from the University of Augsburg in connection with a trainee programme at the Siemens AG. He began his professional career at Siemens Nixdorf Informationssysteme AG as a Commercial Product Manager, responsible for product development, in 1990 and changed to Sortimo International GmbH in 1993. Seven years later he complemented his academic career with an MBA degree from the University of Augsburg and Katz Graduate Business School in Pittsburgh, before joining MELOS GmbH as Managing Director in 2003.

Maximilian Röglinger is Professor of Information Systems and Business Process Management at the University of Bayreuth. Maximilian also serves as Deputy Academic Director of the Research Center Finance & Information Management (FIM) and works with the Project Group Business & Information Systems Engineering of the Fraunhofer FIT. Maximilian's activities in research, teaching, and industry center around customers, processes, information technology, and digital transformation. Maximilian publishes in renowned journals such as Business & Information Systems Engineering, Decision Support Systems, Business Process Management Journal, Journal of Strategic Information Systems, and Journal of the Association for Information Systems. Maximilian is also strongly engaged in publicly and privately funded research projects, in which he contributes to the solution of relevant business problems. He already collaborated with companies such as Allianz, A.T. Kearney, Bayerische Bereitschaftspolizei, Deutsche Bahn, Deutsche Bank, HEINZ-GLAS, HILTI, Infineon Technologies, MELOS, Nord/LB, Radeberger, Siemens, and many more.

Marco Schmidt is a research assistant at the Research Center Finance & Information Management at the University of Augsburg and the Project Group Business & Information Systems Engineering of the Fraunhofer FIT. His research focusses on the Digitalized Life and Individual IS, i.e. the development of IS for individuals. Formerly, he studied Business Information Systems Engineering (B.Sc.) and Computer Science and Infonomics (M.Sc.) at the University of Augsburg. Marco engages in publicly and privately funded research projects in the context of BPM and IT Security with companies such as Hilti AG and MELOS GmbH.