A Practice-Proven Reference Architecture for Model-Based Collaborative Information Systems

Adrian Hernandez-Mendez[*,a], Felix Michel[a], Florian Matthes[a]

[a] Software Engineering for Business Information Systems, Technische Universität München, Germany

Abstract. *This article provides a condensed overview of a layered and model-driven system architecture that empowers domain experts to set up and customize collaborative information systems without programming based on layered business-oriented conceptual models. This architecture emerged from a decade of iterative research, design, development, and technology transfer projects which focused on individual modelling and system construction activities. This paper puts these results and publications into a larger architectural perspective, explains the rationale behind this architecture, and argues to consider it as a reference architecture for model-based collaborative information systems in general.*

Keywords. Reference Architecture • Model-Driven Engineering • Information Systems

1 Introduction

Since 2002, our team at the Technical University of Munich pursues the ambitious research goal to build enterprise-scale collaborative information systems based on business-oriented conceptual models only. This approach should empower business users to directly define and customize the systems they use for their day-to-day knowledge-intensive work in fields like engineering, product management, health-care, education, service management, or IT management without programming.

These models are the results of several cycles of development in collaboration with more than ten large and medium-sized enterprises in domains such as logistics and transportation, health care and insurance in Europe (see also Table 1).

This paper is structured as follows: In Section 2 we describe the notion (vision) of model-driven collaborative information systems from an end-user perspective and then explain in Section 3 how the implementation of such systems naturally leads to eight architectural layers which implement and encapsulate distinct capabilities: data storage, schema definition, access control definition, query capabilities, case modeling, case management, and user interface definition. We then elaborate the core abstractions to define and customize collaborative information systems using an integrated conceptual meta-model (Figure 3) which can be subdivided into models corresponding to the abstractions of the layers. In Section 5 we provide references to projects and scientific publications that report on lessons learned using the generic system in different application areas and sketch the use of the system as a platform for patient-centric health care. The paper ends with related work, a summary and an outlook on future work.

* Corresponding author.
E-mail. adrian.hernandez@tum.de
Some of this work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n° 689802).

2 Collaborative Information Systems

We use the term collaborative information systems (CIS) to describe systems which allow different

stakeholders to collaboratively model and manage their information and knowledge-intensive processes. These processes can be internal to an organization (e. g., financial management, IT management) or can involve the cooperation of members of the organization with customers, suppliers and partner organizations (e. g., product design, customer relationship management, customer-centric healthcare).

In continuously changing environments, the understanding of the details of the information structures and work processes evolves over time, in particular in agile organizations and business ecosystems, which requires a consistency-preserving co-evolution of information, information structures, and information management processes. Consequently, collaborative information systems have to support both, structured and unstructured information (texts, hypertexts, diagrams, media files) as well as structured processes and semi-structured, adaptive case management.

Such a model-driven approach allows practitioners to create and reuse models of their problem domain within their running system instance. This results in the creation of an ecosystem specific to the problem and users can instantly collaborate inside the organization and cross-organization using a basic set of concepts (i. e., Workspaces, Wikis, and Pages). These models are directly deployed to a running system avoiding the hassles of model-to-code transformations and redeployments. Therefore, the enterprise can use domain experts without programming knowledge to quickly set-up a new information system. Furthermore, these systems also allow practitioners to dynamically update and evolve their models to meet the demands of the changing business needs. The model-driven architecture adopted by these systems allows to automatically update the persistence and presentation layers.

In our system implementation, we did not commit to a specific concrete (textual or graphical) syntax for configuring specific collaborative information systems. Instead of this, our generic system provides several (RESTful) APIs to create, read, update or delete (data, access, case, user-interface) models also at runtime. Most of our projects use interactive web-based front-ends where end-users can use form-based, text-based or graphical interfaces for model definition, execution and analysis (similar to Excel, Access, Sharepoint, and Camunda) which are then translated to API calls of our model-based back-end (as a service).

3 Layers and Capabilities of the Reference Architecture

Figure 1 provides an overview of the layers of the reference architecture. The graphical representation emphasizes that each layer adds new abstractions (e. g., an access control model) to the abstractions of the layers below. The abstractions are made accessible to clients via RESTful APIs for introspection or modification. Each concept (e. g., Principal, Group, User, Membership) is mapped to a separate resource in the REST API. The semantic relationships between the concepts (association, aggregation, sub-classing) are mapped to (bi-directional) links between resources or specialized resources.

The database symbols and the colored boxes within it indicate that the models are first-class entities in the persistent store. This leads to a nice orthogonality of concepts. For example, access control and version control also apply to higher-level concepts like case definitions.

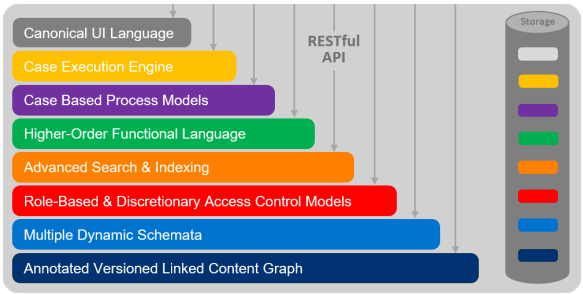

Figure 1: Layers of the reference architecture.

The same colour coding of the layers and the models is also used in Figure 3 in Section 4 to explain which semantic concept is realized at which layer of the architecture.

Three main principles heavily influenced our system design: Empowerment of domain experts and end-users, agility in the model/data and process evolution, and a secure self-contained environment for intra- and inter-organizational collaboration.

After several iterations within collaboration academic and industry projects, we derived a group of orthogonal concepts to ensure the flexibility and scalability of the system despite their expressiveness for modelling the enterprise data. The concepts are Workspaces, Pages, Entities, Attributes, Files, and Principals.

An instance of the system is used by an organization or a network of collaborating organizations. An arbitrary number of personal, team, project or organizational-unit workspaces can be created. This allows different communities of practice to model and manage their information and processes in their workspaces independently, or to selectively share information and models (e. g. CRM data and processes) across community boundaries.

The reference architecture comprises the following eight layers and capabilities:

Annotated Versioned Linked Content Graph: The first layer supports versioned data storage and ensures the capability of the system to store revisions semi-structured data in a form of entities (i. e., JSON objects) and references between them. The rationale is that companies already manage structured and unstructured data, so it is necessary to create a mechanism for the system to import an initial set of semi-structured data (e. g., Excel sheets, wiki pages with embedded tables and media) without strong data schema that can evolve over the time. This layer supports the data-first (schema second) approach for data modelling (Büchner 2007).

Multiple Dynamic Schemata: This layer allows the users of the information system to collaboratively structure their information over time by adding or removing schema constraints involving relationship and cardinality constraints as needed. A data schema is thus not seen as a definition of a container that is subsequently filled with data (like an SQL database), but as a collection of constraints imposed on a flexible content graph that evolves over time. However, this implies that at certain times during system evolution the data can contain inconsistencies, which can be resolved collaboratively without leaving the scope of the system. (Büchner 2007, Matthes et al. 2011)

Role-Based & Discretionary Access Control: This layer addresses the security concerns in the system. The security model comprises users and groups which can be internal or external to the organization. The access rights (administrator, editor, author, and reader roles) are defined initially at the Workspace level and can be overwritten (loosened or tightened) at the Entity level. The rationale is to guarantee the secure access to the data stored in the system.

Advanced Search & Indexing: In this layer, the unstructured data (i. e., long texts or hypertexts) is linked to the semi-structured data (using parsing and full-text indexing technologies). This is a core element of the Hybrid Wiki approach (Matthes et al. 2011) and lays the groundwork for natural language processing techniques and model discovery processes. The rationale behind this design is the fact that not all the data in the enterprise is structured or semi-structured and the system should be able to use all the possible formats of the data in the enterprise without the need of an upfront structuring process.

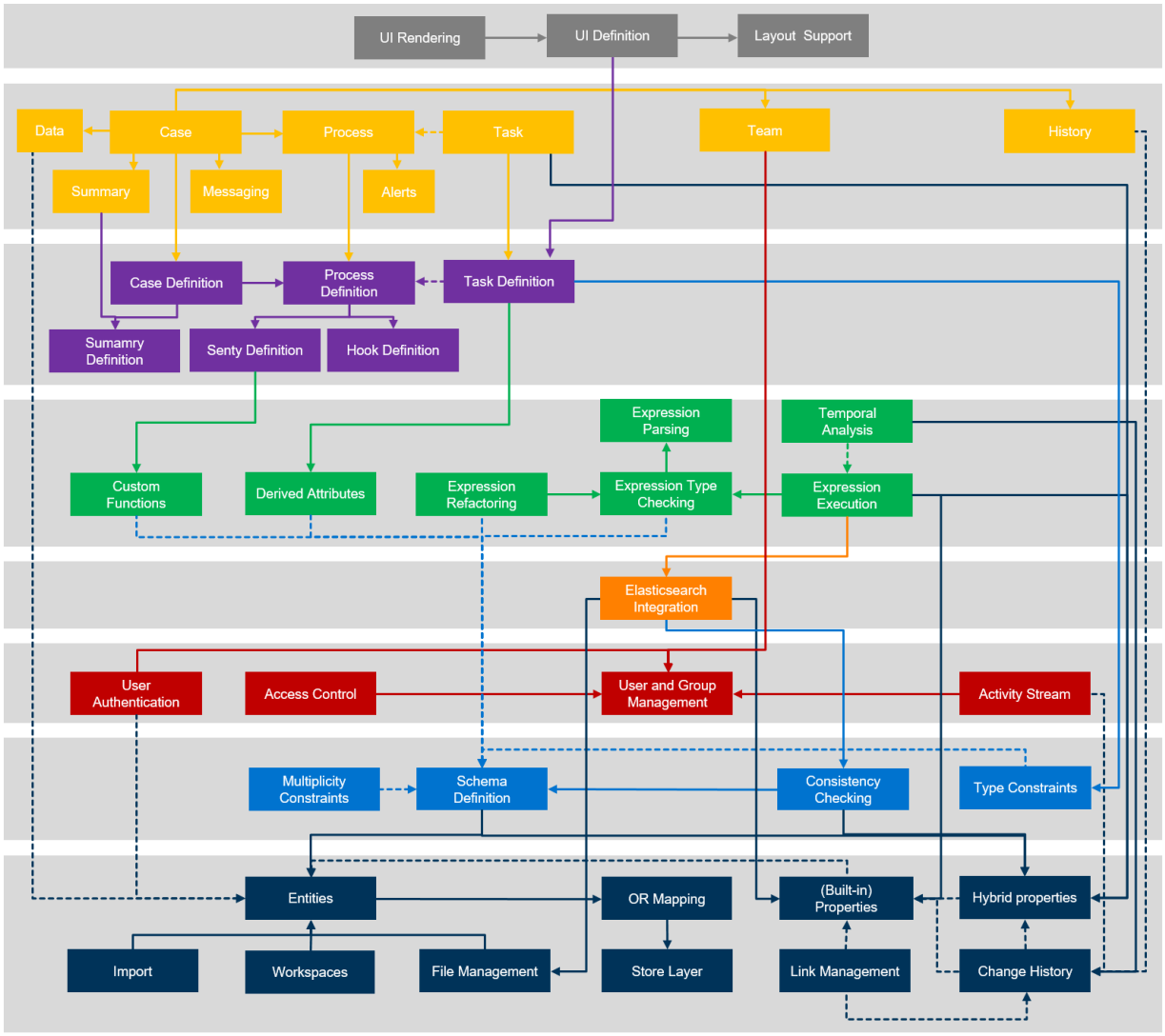

Figure 2: Capabilities ordered according to conceptual layers. A solid line represents the usage of a certain capability and a dashed line represents extended functionality.

Higher-Order Functional Language: This layer introduces a strongly-typed query language (similar to LINQ - Meijer et al. 2006) to access all the structured information in the information system. This language allows adding new attributes to Entities as a result of the computations (i. e., Derived Attributes) and using operations over collections of Entities such as Map, Reduce, and Join. The rationale behind that is creating a flexible mechanism to the user of the system to extract knowledge from the stored data, which is not limited by the basic API of the system, and allows the users to create flexible and tailored data views based on individual information demands. (Reschenhofer and Matthes 2016)

Case Based Process Models: This layer allows the user to define knowledge- and data-intensive processes following the adaptive case management paradigm (Swenson, Palmer et al. 2010). The user can define stages, human tasks and automated tasks that make up a collaborative

process. The tasks can link to one or multiple Attributes which are needed or modified by each task. Thereby, the system allows the user to define standard processes or reusable process fragments which can be dynamically instantiated as needed.

Case Execution Engine: This layer manages the information regarding the current states of all cases being executed in the system and the links to the shared or case-local data and the actors involved in a case. The case execution engine enforces the access control policies defined in the access rights layer and the data management layer keeps an audit trail of the executed steps and data modifications. The rationale is to keep track of the process steps execution and how the users accomplished the case also for future analysis (e. g. for compliance or prediction purposes).

Canonical UI Language: This layer allows the user to define links between the data and its representation using the predefined access control policies. This representation includes information about how the Attributes should be grouped and laid out visually. This ensures a clear separation between the data and its representation, thereby an Entity can have multiple representations based on the context of use. Therefore, Entities can be represented using vertical and horizontal layouts to display their attributes. The rationale is to capture the knowledge regarding the presentation of information in a model without the need to inspect specific UI implementations.

Figure 2 shows all the capabilities that are provided in each layer of the reference architecture and their semantic relationships (capability A uses capability B, capability A extends capability B). The colour coding links the elements to the layers depicted in Figure 1 and the concepts in the conceptual model in Figure 3. We found this capability map to be a very useful starting point for core developers of the platform to understand the key dependencies in the architecture.

4 Conceptual Meta-Model

The underlying meta-model of cooperative information systems (see Figure 3) builds on the already introduced core concepts of Workspaces as containers (name spaces) for Entities, EntityDefinitions, Attributes, and AttributeDefinitions. These concepts structure the model inside a Workspace and capture its current state. An Entity consists of a collection of Attributes, and the Attributes are stored as a key-value pair. The attributes have a name and can store an ordered list of values of different types, for example, strings or references to other Entities. The user can create an attribute at run-time to capture structured information about an Entity. An EntityDefinition allows users to refer to a collection of similar Entities and their common schema, e. g., organizations, persons, amongst others. The EntityDefinition consists of multiple AttributeDefinitions, which in turn contain multiple validators such as multiplicity validator, string value validator, and link value validator. Additionally, an individual Attribute and its values can be associated with validators for maintaining integrity constraints.

The EntityDefinition and AttributeDefinition are loosely coupled with Entity and Attribute respectively through their name. These elements specify soft-constraints on the Entities and Attributes. The use of soft-constraints implies that the users are not restrained by strict integrity constraints while capturing information in Entities and their Attributes. Therefore, the system can store a value violating integrity constraints as defined in the current model (e. g., during schema evolution stages, or after import of non-conformant data from an external data source).

A CaseDefinition is a template for all related case instances and links to a root EntityDefinition which describes the schema of the case data. Every CaseDefinition consists of multiple ProcessDefinitions that are either a complex StageDefinition, a single HumanTaskDefinition or an AutomatedTaskDefinition. StageDefinitions

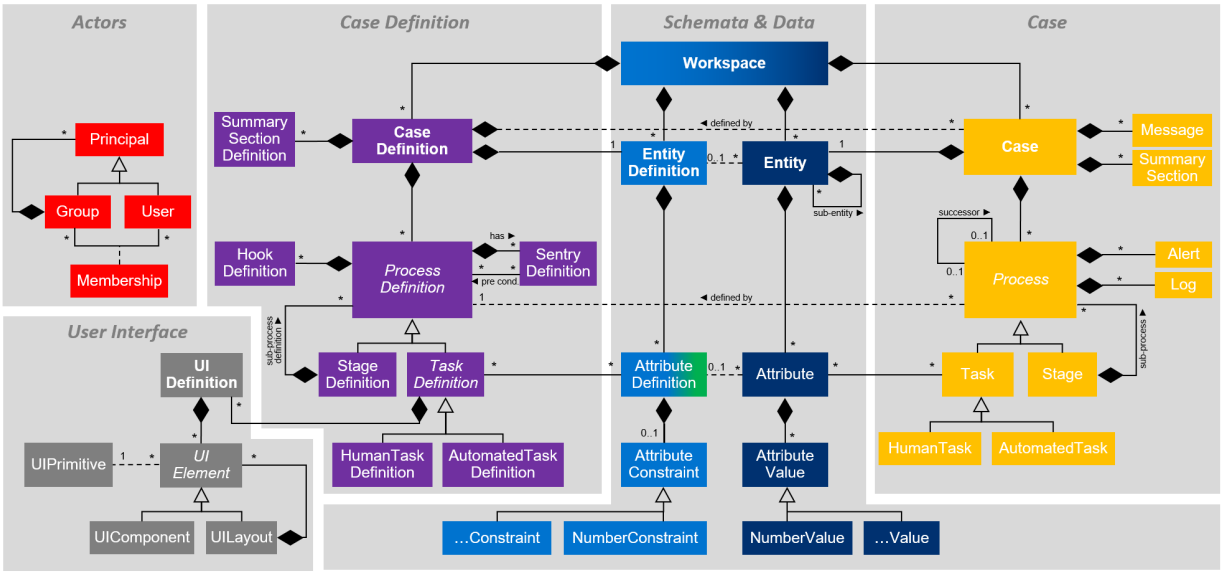

Figure 3: Conceptual Meta-Model.

represent a container that is used to group either sub StageDefinitions or TaskDefinitions. Preconditions to the activation of a stage are expressed as SentryDefinitions that either depend on some previously accomplished Process elements or expressions (Boolean predicates) related to the case data. In order to receive notifications on state changes of a Process element, it is possible to define HookDefinitions related to a Process-Definition. These hooks can be used to integrate services from third-party systems (with REST APIs). Additionally, SummarySectionDefintions are used to create short summaries (data views) based on the data created by the Case.

A Case is an instance of a CaseDefinition and follows the same structure. Each Case model element has a state, such as AVAILABLE, ENABLE, ACTIVE, COMPLETED or TERMINATED. Several model elements such as Message, Alert and Log exist only at the instance level. Cases are mostly knowledge-intensive and often lead to a discussion between all stakeholders involved, therefore Messages are attached to a case. The concept of Alerts helps to notify users involved in a Case, for example about overdue tasks. Also, third-party systems can create these alerts for guiding or alerting users. Runtime errors or exceptions are also stored persistently as Log elements.

A UIDefinition describes how attributes are to be presented to the user in a user interface (e. g., a web form). A UIDefinition is composed of a group a UIElements, every of which can be a single component or a layout element. A UI definition refers to an implementation of a software component that dynamically interprets UI model (i. e., a UI renderer component). Using this model, the knowledge regarding the presentation of the attributes is not tied to a specific technology. For example, the user can group attributes in a UILayout instance and provide configurations such as labels and validation errors that can be used at runtime by the UI renderer to create web forms using the Material Design[1] framework.

5 Applications of the reference architecture

In recent years, the reference architecture has been used in several business domains and in

[1] https://material.io/guidelines (Accessed on: 28/11/2017)

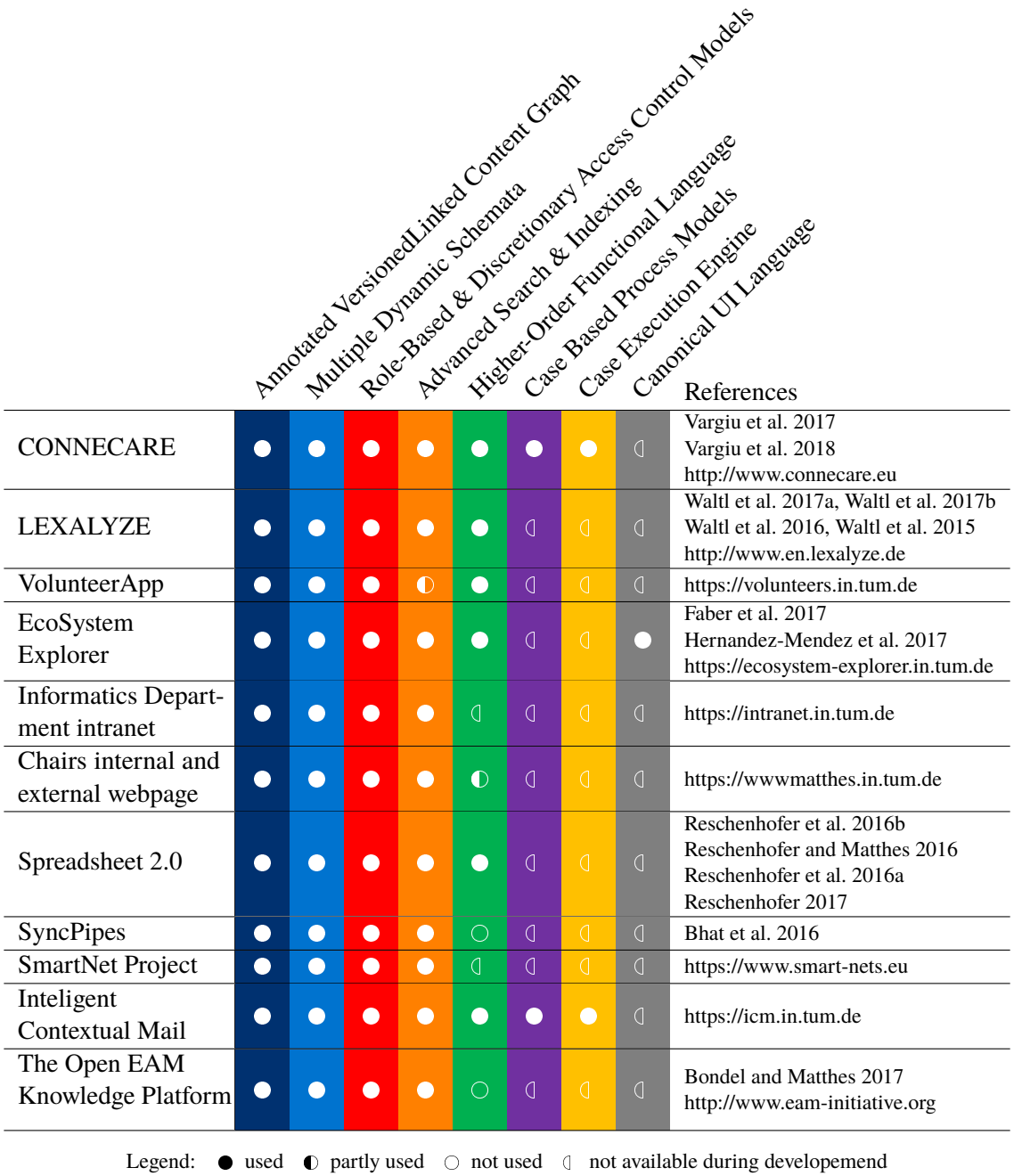

Table 1: Projects that use the reference architecture.

different organizational contexts. Table 1 lists the most relevant use cases, indicates the layers of the architecture being used an provides references to scientific publications with more detailed information about the experience gained using the architecture in each use case.

In the remainder of this section, we explain the application of the reference architecture in the European health-care project CONNECARE which provides a digital platform for Personalized Connected Care for Complex Chronic Patients

that also allow gaining new medical insights based on the analysis of case studies. The general objective is to connect all professionals (doctors, nurses, etc.) directly with the patient via smartphone apps and wearable devices and manage the cases in a model-driven way via an adaptive case management platform. Therefore, three case study types are defined that are applied in four different hospitals that are spread across Europe. Every case study follows the same basic stages such as case identification, case evaluation, work plan definition, work plan execution and finally the discharge stage.

Figure 4 illustrates the workflow of one case study using the CMMN, the highlighted parts represent the basic stages of every case. Within every stage multiple tasks need to be accomplished by the clinicians, e. g. completing a Carlson[2] questionnaire that then automatically computes the related score. During the work plan stage, the clinicians can define tasks that need to be executed by the patient such as to do ten sit-ups per day. The patient performs these tasks assisted by a smartphone app and the clinician is notified about the execution results. For a comprehensive summary of the whole CONNECARE project consult (Vargiu et al. 2017, 2018).

Figure 5 shows the high-level CONNECARE architecture focusing on the Smart Adaptive Case Management system and its related systems. The main components are the Smart Adaptive Case Management (SACM) for all interactions with the professional, and the Self-Management System (SMS) for all interactions with the patient. The SACM system consists of several components such as: 1) *Professional Interface* provides information for professionals such as doctors, nurses etc. The Angular 2.0 client application of this interface uses the domain-specific API, 2) *Decision Support System* support clinicians during the treatment process, 3) *SocioCortex-Server* is an instance of the reference architecture described in this paper and is accessible by a JSON based RESTful API and is responsible for case modeling, case execution, access control, data storage and data modeling and for the integration of all other systems 4) *SACM-Backend* provides a deployment-specific functionality as well as an API for the Professional Interface and the Message Broker. The functionality of the SocioCortex-Server is wrapped and enhanced to provide deployment-specific functionality.

The SACM-Backend is communicating via a Message Broker with the central User Identity Management and the SMS. In the future, the information from the hospital information systems at the various deployment sites in Europe will be exchanged via the message broker as well. All components are deployed as Docker micro services on the Amazon EC2 cloud.

6 Related Work

To the best of our knowledge, this paper provides the first reference architecture for model-based collaborative information systems which integrates process and data modelling in a fully model-based system without depending on programming skills and suitable for productive use in industry.

However, this work builds on the vast knowledge that has been created from research in information systems, conceptual modelling and process modelling (Hesse and Mayr 2008; Reichert et al. 2007).

In the domain of information systems, we can highlight two compelling works that complement the concepts described in the proposed architecture. First, the concept of conceptual independence introduced by (McGinnes 2011) (McGinnes and Kapros 2015). This work describes the problems that arise when the internal software components and their corresponding conceptual models are highly coupled. Likewise,

[2] http://www.bgs.org.uk/pdfs/assessment/cci.pdf (Accessed on: 28/11/2017)

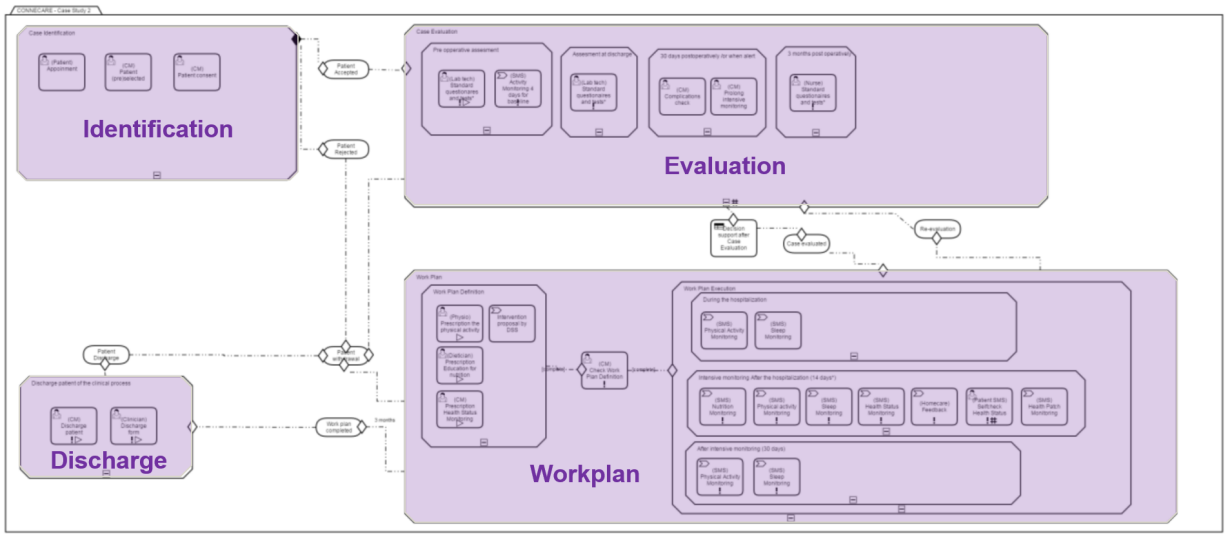

Figure 4: CONNECARE case study sample modeled in CMMN notation.

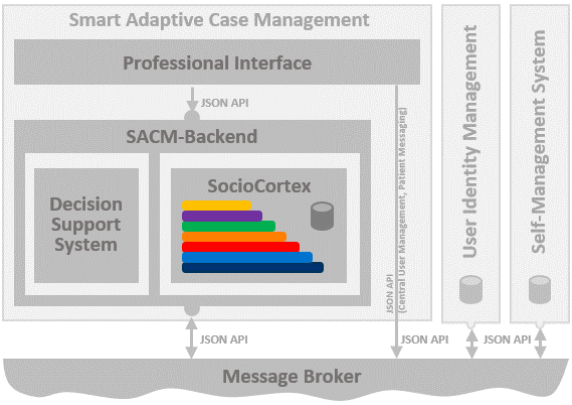

Figure 5: CONNECARE Architecture with focus on the Smart Adaptive Case Management system.

a type of adaptive information system is proposed to resolve the dependencies between the system and the conceptual models. This work shares the same problem that motivates our research. However, it is limited to the analysis of data models and not how the system manages the processes for the creation of knowledge. Second, the Social Knowledge Environments, introduced by (Pawlowski et al. 2014), establishes the benefits of the integration between social software and the creation of knowledge in the context of information systems and motivates the study of such integration. To our consideration, the proposed reference architecture contributes to the study that is motivated in (Pawlowski et al. 2014), because it was created from concrete examples of collaboration between academia and industry that allow analysing the different roles and social implications of the data modelling and knowledge creation processes.

From a process modelling perspective, it is worth to mention two approaches that could lay the ground for concrete implementations of the reference architecture. First, the Normalized Systems Theory, proposed by (Mannaert and Verelst 2009), which establishes a procedure for modelling systems considering evolvability principles. This approach can provide guidelines to experts how to create process models using the concrete implementation of the proposed reference architecture. The second prominent approach is Object-aware Business Processes proposed by (Künzle et al. 2010), and the PHILharmonic Flows framework (Carolina Ming Chiao et al. 2014). This approach provides building blocks to foster flexibility in process models, and has been applied in several contexts such as healthcare (C. M. Chiao et al. 2013b) and

university internal processes (C. M. Chiao et al. 2013a).

7 Conclusions

In this paper, we described the notion of model-driven collaborative information systems from an end-user perspective and explained how the implementation of such systems naturally leads to eight architectural layers which implement and encapsulate specific information modelling and management capabilities. We identified the core abstractions needed to define and customize collaborative information systems and developed an integrated conceptual meta-model which can be subdivided into models corresponding to the abstractions of the layers.

Based on the positive experience of a decade of practical (commercial) use of the architecture in several business domains and deployment contexts (intra-organizational, cross-organizational collaboration), we concluded that the architectures should be considered as a reference architecture for cooperative information systems.

In the future, we plan to integrate NLP, NLG and ML capabilities into the functional query layer and to study how deontic models and so-called smart contract languages fit into the layered architecture of model-based collaborative information systems.

References

Bhat M., Shumaiev K., Biesdorf A., Hohenstein U., Hassel M., Matthes F. (2016) Meta-model based framework for architectural knowledge management. In: Proccedings of the 10th European Conference on Software Architecture Workshops. ACM, p. 12

Bondel G., Matthes F. (2017) Hybride Wikis als Repository für IT-Unternehmensarchitektur. In: IT-Unternehmensarchitektur: Von der Geschäftsstrategie zur optimalen IT-Unterstützung

Büchner T. (2007) Introspektive modellgetriebene Softwareentwicklung. Dissertation, Technische Universität München, München

Chiao C. M., Künzle V., Reichert M. (2013a) Integrated modeling of process- and data-centric software systems with PHILharmonicFlows. In: 2013 IEEE 1st International Workshop on Communicating Business Process and Software Models: Quality, Understandability, and Maintainability, CPSM 2013. IEEE Computer Society, Eindhoven

Chiao C. M., Künzle V., Reichert M., Künzle V. (2013b) Object-aware process support in healthcare information systems: Requirements, conceptual framework and examples. In: International Journal on Advances in Life Sciences 5(1-2), pp. 11–26

Chiao C. M., Küenzle V., Reichert M. (2014) Towards schema evolution in object-aware process management systems. In: Enterprise modelling and information systems architectures - EMISA 2014, Luxembourg, September 25-26, 2014, pp. 101–115

Faber A., Hernandez-Mendez A., Matthes F. (2017) Towards an Understanding of the Connected Mobility Ecosystem from a German Perspective. In: ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems, Volume 3, Porto, Portugal, April 26-29, 2017, pp. 543–549

Hernandez-Mendez A., Faber A., Matthes F. (2017) Towards a Data Science Environment for Modeling Business Ecosystems: The Connected Mobility Case. In: Advances in Databases and Information Systems. Springer, pp. 324–330

Hesse W., Mayr H. C. (2008) Modellierung in der Softwaretechnik: eine Bestandsaufnahme. In: Informatik-Spektrum 31(5), pp. 377–393

Künzle V., Weber B., Reichert M. (2010) Object-aware Business Processes: Properties, Requirements, Existing Approaches. Technical Report UIB-2010-06. University of Ulm. Ulm, Germany

Mannaert H., Verelst J. (2009) Normalized Systems: Re-creating Information Technology Based on Laws for Software Evolvability. Koppa

Matthes F., Neubert C., Steinhoff A. (2011) Hybrid Wikis: Empowering Users to Collaboratively Structure Information.. In: ICSOFT (1) 11, pp. 250–259

McGinnes S. (2011) Conceptual modelling for web information systems: What semantics can be shared? In: De Troyer O., Bauzer Medeiros C., Billen R., Hallot P., Simitsis A., Van Mingroot H. (eds.) 30th International Conference on Conceptual Modeling. Lecture Notes in Computer Science Vol. 6999. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 4–13

McGinnes S., Kapros E. (2015) Conceptual independence: A design principle for the construction of adaptive information systems. In: Information Systems 47, pp. 33–50

Meijer E., Beckman B., Bierman G. (2006) LINQ: Reconciling Object, Relations and XML in the .NET Framework. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. SIGMOD '06. ACM, Chicago, IL, USA, pp. 706–706

Pawlowski J. M., Bick M., Peinl R., Thalmann S., Maier R., Hetmank L., Kruse P., Martensen M., Pirkkalainen H. (2014) Social Knowledge Environments. In: Business & Information Systems Engineering 6(2), pp. 81–88

Reichert M., Strecker S., Turowski K. (2007) Proceedings of the 2nd Int'l Workshop on Enterprise Modelling and Information Systems Architectures - Concepts and Applications (EMISA'07). Lecture Notes in Informatics LNCS4549. GI - Gesellschaft fur Informatik

Reschenhofer T. (2017) Empowering End-users to Collaboratively Analyze Evolving Complex Linked Data. PhD thesis, Technische Universität München

Reschenhofer T., Bürgin P., Matthes F. (2016a) A Social Information Flow Graph: Design and Prototypical Implementation.. In: CAiSE Forum, pp. 137–144

Reschenhofer T., Matthes F. (2016) Supporting end-users in defining complex queries on evolving and domain-specific data models. In: 2016 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2016, Cambridge, United Kingdom, September 4-8, 2016, pp. 96–100

Reschenhofer T., Waltl B., Gjorgievska S., Matthes F. (2016b) A Semantic Meta Model of Spreadsheets.. In: ECIS, ResearchPaper90

Swenson K. D., Palmer N. et al. (2010) Mastering the unpredictable: how adaptive case management will revolutionize the way that knowledge workers get things done Vol. 1. Meghan-Kiffer Press Tampa

Vargiu E., Fernández J., Miralles F., Cano I., Gimeno-Santos E., Hernandez C., Torres G., J. Colomina J. d. B., Kaye R., Azaria B., Nakar S., Lahr M., Metting E., Jager M., Meetsma H., Mariani S., Mamei M., Zambonelli F., Michel F., Matthes F., Goulden J., Eaglesham J., Lowe C. (2017) Integrated care for complex chronic patients background. In: ICIC17 – 17th International Conference on Integrated Care, Dublin. International Foundation for Integrated Care

Vargiu E., Fernández J., Miralles F., Haim R., Nakar S., Weijers V., Meetsma H., Mariani S., Mamei M., Zambonelli F., Michel F., Kelly J., Eaglesham J. (2018) Patient Empowerment and Case Management in CONNECARE. In: GCIC18 – Global Conference on Integrated Care, 2018, Singapore

Waltl B., Landthaler J., Matthes F. (2016) Differentiation and Empirical Analysis of Reference Types in Legal Documents. In: Legal Knowledge and Information Systems - JURIX 2016: The Twenty-Ninth Annual Conference, pp. 211–214

Waltl B., Reschenhofer T., Matthes F. (2017a) Process and Tool-Support to Collaboratively Formalize Statutory Texts by Executable Models. In: International Conference on Database and Expert Systems Applications. Springer, pp. 118–125

Waltl B., Reschenhofer T., Matthes F. (2017b) Process and Tool-Support to Collaboratively Formalize Statutory Texts by Executable Models. In: Database and Expert Systems Applications - 28th International Conference, DEXA 2017, Lyon, France, August 28-31, 2017, Proceedings, Part II, pp. 118–125

Waltl B., Zec M., Matthes F. (2015) A Data Science Environment for Legal Texts. In: Legal Knowledge and Information Systems - JURIX 2015: The Twenty-Eighth Annual Conference, Braga, Portual, December 10-11, 2015, pp. 193–194