Journal of Computer Science

Original Research Paper

Smart-EIS: An End to End Enterprise Information Systems Framework

[1]Jalal Kiswani, [2]Muhanna Muhanna and [3]Abdallah Qusef

[1]Department of Computer Science and Engineering, University of Nevada, Reno, USA
[2]Department of Creative Media, Luminus Technical University, Amman, Jordan
[3]Department of Software Engineering, Princess Sumaya University for Technology, Amman, Jordan

Article history
Received: 29-08-2018
Revised: 19-11-2018
Accepted: 22-12-2018

Corresponding author:
Abdallah Qusef
Department of Software Engineering, Princess Sumaya University for Technology, Amman, Jordan
Email: a.qusef@psut.edu.jo

Abstract: Enterprise Information Systems (EIS) are widely and extensively used in many domains such as banking, telecommunication, e-commerce and government. Although several research studies and investigations were performed that explore the importance of EIS, only a few studies have focused on effective and efficient end-to-end approaches to developing such systems. In this article, a proposed software development framework (Smart-EIS) is presented. The primary objective of Smart-EIS is making the development of high-quality EIS more effective and efficient. In particular, it aims to reduce the development cost and to provide built-in transparent quality, security, performance and user-experience features. A comprehensive review of the traditional EIS is presented. This includes a discussion of the characteristics and patterns of such systems, the layered architectural patterns and the main components of these systems. The working methodology for the work discussed in this article depends on dynamically construct the common and general aspects of EIS at runtime. The methodology starts with extracting metadata models from the traditional architectural and components patterns. Based on these metadata, APIs have been designed and implemented. These libraries were then composed to make the full and complete proposed framework. In terms of validation and evaluation, the proposed framework -including its APIs- has been implemented as open-source projects, used to build a simple human resource management system, then utilized to re-build a student information system. Results of validation and evaluation have been presented and discussed, which show promising potential.

Keywords: Enterprise Information Systems, Rapid Application Development, Dynamic GUI Generation, Dynamic Software Artifacts Generation

Introduction

Enterprise Information Systems (EIS) are important assets for organizations of all levels. EIS implementation aims to automate as much as possible from an organization internal processes, to get the results faster and reduce the operational overhead. Consequently, it reduces cost and increases profit. In fact, not having a proper EIS in place for some domain, such as banking, telecommunications and e-commerce, shall produce risks that threat an organizations existence Tamm *et al*. (2011) Da Xu (2011). However, even though implementing EIS can support organizations achieving their goals, it requires large investments in terms of cost, time and resources. Since the required large number of functional requirements, base of software artifacts and complex hard ware infrastructure, are all factors of increasing that investments. Other challenges include, the availability of the skilled and experienced resources, requirements customization and integration with other systems. In this article, we present an approach that can significantly reduce the cost of EIS systems development, increasing their overall quality and maintainability using metadata-driven development and runtime code generation. Furthermore, we present Smart-EIS framework as the proposed approach's implementation. In particular, it

Science Publications

© 2018 Jalal Kiswani, Muhanna Muhanna and Abdallah Qusef. This open access article is distributed under a Creative Commons Attribution (CC-BY) 3.0 license.

includes the architecture, design and implementation of (Smart-EIS). Motivated by (i) building on main author's experience in the domain of enterprise information systems, (ii) helping in minimizing the need of specialized experts in building such systems, (iii) and supporting the open-source community, we propose our work starting with a philosophical research question: Is there an approach to making the development of high-quality enterprise information systems rapid and cost-effective?

The remaining part of this article is organized as follows: Section 2 presents a comprehensive review of the main concepts and disciplines that the work of this research is based on. Section 3 includes the major work done related to our work. Section 4 presents the methodology of this work, in terms of processes, scope, technology and tools, naming convention and constraints. Moreover, it describes the design aspects of the proposed framework. Section 5 presents the scope, implementation aspects of the APIs of Smart-EIS through a working example. It also presents how Smart-EIS has been put into practice through a case study, where it was implemented in a current student information system, to be analyzed for several quantitative metrics. Section 6 and 7 present the validation and evaluation and the results discussion. Section 8 concludes this article and identifies several future work directions.

Background

This section provides a brief, yet comprehensive, description of the main concepts and literature that this work is based on. In particular, it reviews the current literature in the main requirements for developing EIS and components of information systems.

Information Systems (IS) is a particular type of software application in which data is the essential component of the system Stair and Reynolds (2017) Sommerville (2015). In traditional IS applications, relational Database Management System (DBMS) is used as the main repository of data. The design and architecture process of small to medium IS is straightforward, especially with the availability and maturity of design patterns and best practices Jin *et al.* (2016) Tabatabaie *et al.* (2009) Fowler and Beck (1999) Wolfgang (1994).

Most IS consisting of three categories of low-level software components, user interface, business logic and data access. In this paper, user-interface components are referred as views. In general, information systems may be designed based on the organization of these components into a one-tier, two-tier, three-tier or n-tier architecture Fowler (2002) Cervantes and Kazman (2016).

Requirements for Smart EIS

This section discusses the main requirements for building and developing Smart EIS.

Development of EIS applications requires longtime, significant efforts and special expertise Stair and Reynolds (2017). This is a result of several challenges categorized as (i) complex business rules, (ii) particular quality attributes, (iii) technical requirements.

Firstly, most EIS includes special business requirements and application and logic. In particular, they include:

1. Complex business rules; such as the validation rules required for making financial wire-transfers
2. Dynamic requirements that may change frequently; where EIS should be responsive and elastic to market need and regulation changes, such as changing the regulations by governments or central authorities, or a change of management personnel Stair and Reynolds (2017)
3. Various types of end-users front-end technologies; where enabling software applications to different front-end technologies has almost become mandatory with the trends of web and mobile platforms, to allow external access and provide online services for the clients
4. Various data-sources; enterprise organizations are most likely to have multiple applications running concurrently. For example, it is more likely that any financial organization runs an Enterprise Resource Planning system (ERP), a core-banking system and an e-channels for online services in parallel. In this case, reducing data redundancy is significant and the data interchange between applications is required. Consequently, applications become dependent on various data-sources (i.e., the primary application data source and other applications data sources) Royce (1987), which increase the complexity of managing and supporting them

Secondly, EIS requires special quality attributes (i.e., non-functional requirements) to be taken into consideration. In particular, availability, fault-tolerance, reliability, integrity, scalability and security Sommerville (2015) Bass *et al.* (2012). Therefore, complex hardware environments and data centers are common situations in enterprise-level organizations. In fact, they invest in advanced computer hardware equipments such as blade-servers, storage area networks, load-balancers, traditional and web applications firewalls. This result in a complex hardware environment De Alwis and Sillito (2009). Furthermore, the organization may also utilize cloud-computing by building private or hybrid cloud infrastructure The Economist (2008).

Thirdly, most modern EIS shall include special technical requirements, such as:

1. Localization and internationalization; with the current globalization phenomena, in which clients in

different countries, languages and cultures may use the services provided from an organization, the requirement enabling applications to support localization is significant. In particular, localization shall be done on three levels: User-interface, input fields and reports become mandatory. Furthermore, localization is important for scalability, international exposure Valipour *et al.* (2009) and economies of scale Sullivan (2003)

2. Dynamic validation rules; in enterprise level mission critical industries such banking and governments, the requirements and regulations may change frequently and in many times, these changes become effective immediately, which consequently, may require immediate changes in the business rules and validation of the system. Thus, an efficient way to update the validation rules dynamically without changing the source code is significant. In fact, this will mitigate against the high-cost and risk required by going through the software development lifecycle (SDLC) Sommerville (2015) Bianco *et al.* (2007)
3. Unified technical logging; In the case of failures or bugs, due to the complexity of EIS, the debugging process is more complicated than debugging small-medium applications. Thus, the need to have technical and tracing information will make it relatively easier to support and troubleshoot such errors. Consequently, a standard logging mechanism to record and enable tracing of issues should be unified across the entire application
4. Unified exception handling; during applications runtime, many issues are out of developer's control, such as a network or input/output failures. Handling of such issues should be performed in a consistent approach Borger and Schulte (2000)
5. Software infrastructure. Manual building of large systems is considered long-term and prone to errors Royce (1987). In addition, testing all of the functionality and application's requirements manually on every build is not practical Do *et al.* (2006). In fact, leaving them untested will also be a critical problem since the quality and consistency of applications are not guaranteed. Thus, test and build automation, standard project structure, contentious integration and continuous delivery are significant to boost developers productivity and increase applications quality and stability Kiswani *et al.* (2017a)
6. Standard architectures and best-practices; such techniques of using standard architectural styles and patterns Cervantes and Kazman (2016), are significant to ensure long-term stability and maintainability. Furthermore, being up-to-date with the latest and mature standards and best practices will reduce the risk of being outdated or using old techniques that may affect the overall performance and requirements of the applications Capilla *et al.* (2016) Fowler and Beck (1999)

On the other hand, ensuring that EIS applications deliver the promised agreed quality attributes at production is important Bass *et al.* (2012), which include the application benchmarking and tuning in production environments Dearie (2007). Caching implementation is commons to reducing network traffic and data-sources access by saving the state of some application's components or data on different tiers or layers of applications. Also, utilizing clustering is also frequent in enterprise organizations. In particular, deploying applications on multiple nodes on a network. However, clustering requires a particular –and most likely complex- configurations on both software and hardware levels Schlossnagle (2006).

In addition, in modern applications, some new business requirements have started to be mandatory in every EIS Da Xu (2011) Dearie (2007), such as:

1. Workflow: extracting the business rules and making them configurable without the changing the source code and going through the SDLC again. In particular, implementing a workflow engine such as JBoss jBPM (https://www.jbpm.org)
2. Audit-trail: auditing is a critical process which involves almost all business-critical operations in any enterprise-level organization. Providing the required information for auditing business audits for all transactions is required on the application to ensure the system is used based on the organization internal rules and regulations and compliance to the external regulators
3. Runtime-implementation toolkits: Which is the ability to modify the application user-interface at runtime, by adding, modifying or removing input fields, change the look and feel, or change application configuration without the need of application restart or re-deploy

Traditional Components of Information System Applications

This section briefly discusses the main components that are available in most information systems applications based on Service-Oriented Architecture (SOA).

As seen in Fig. 1, applications may be composed of multiple sub-systems, which are called modules Bass *et al.* (2012). For example, an enterprise resource planning system may consist of general-ledger, purchasing and human resources modules.

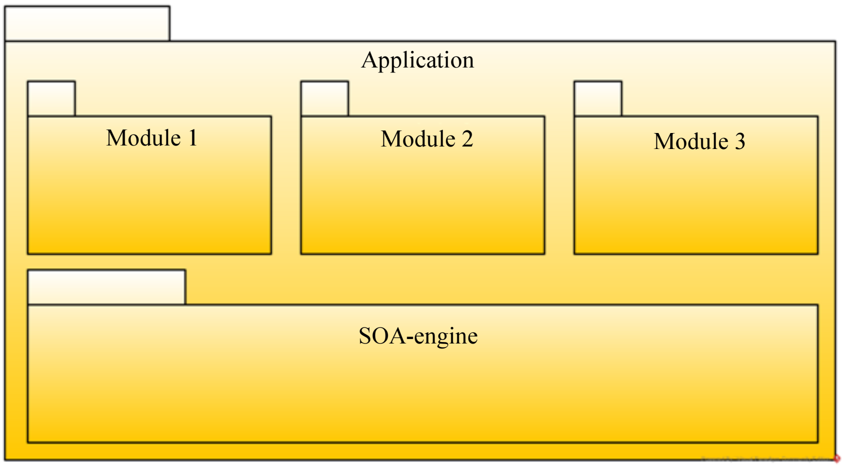

Fig. 1: An IS application structure

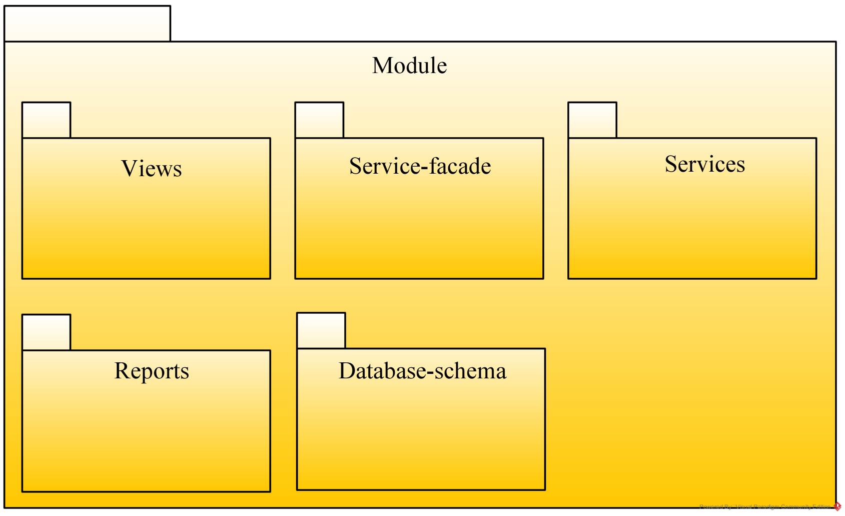

Fig. 2: Components of a module in an information system application

Modules may consist of several components, including user-interface components (views), service-facade, services, reports and database-schema, as shown in Fig. 2. Views are organized and structured using menus, which could be rendered as trees, traditional drop-down menus, expand-collapse, or any other formats McGovern *et al.* (2003). Every module has its services and coarse-grained service facade on module level. Moreover, every module uses its database tables and schema, along with the operational business reports, that most-likely will be embedded in the system, which could be printed out or exported using UI components (e.g., student transcript, salary slip, etc.).

Related Work

This section discusses the related work to the framework presented in this research. It is organized chronologically in terms of the publication date from the oldest to the newest. However, the section follows the literature review to summarize the literature concerning software architecture in general and enterprise architecture. In particular. It covers what have been found and what have been reported to the best of our knowledge in the core disciplines related to enterprise software architecture, best practices and trends, as well as dynamic functionality and components generation. Most of the related work studies and investigations have explored only the theoretical definitions of software architecture, issues, requirements and importance. Moreover, we believe that the amount of research done on dynamic software functionality, software components and software architecture styles for end-to-end design and development is humble in the count and needs a broader exploration.

Winston Royce was the first to introduce the discipline of software architecture Royce (1987). In particular, in 1970, he published an article about managing the development of large software systems. In his article, he referred to his experience working with large software projects in the air crafting and flights management. Moreover, he described the importance of software architecture and design phases during the software project life cycle. Also, he discusses how it is essential to document the architecture and design decisions, to be able to produce a higher-level of quality and deliver software based on requirements and constraints.

In an introductory article to a particular issue on software architecture Garlan and Perry (1995), the authors explained how the software architecture was starting to be a necessary discipline in software engineering. They explored several definitions of software architecture and introduced an evolved one. Furthermore, they described the benefits and importance of software architecture. In particular, they explored the significance of software frameworks in being reusable components with high functionality and how they can make the development of related systems more cost-effective. Moreover, they explained how software frameworks could: (i) Help in understanding the system; by giving stakeholders the ability to see the big picture in terms of what can or cannot be done, (ii) support reusable functionality and components, as well as relatively large frameworks to reduce the development cost and efforts, (iii), respond to changes; by adding, removing or modifying existing components, (iv) make the system analysis more understood and more practical, (v) and help in software management during development, implementation and operation.

In his Ph.D. dissertation Johnson (2002) published in 2002, the author studied the enterprise software systems of companies in the Swedish electricity industry. He reported that applying software architecture analysis in the context of enterprise organization is far from being straightforward. In fact, he identified a need for a better exploration of architectural integration styles. He proposed a modified process for architectural analysis and presented an evaluation of some analysis methods. Da Xu (2011) explored the new trends required in most of EIS, such as business process management, workflow management, enterprise application integration, SOA, among others. He continued his investigation with other researchers and published the results Niu *et al.* (2013). In this article, they discussed enterprise architecture, but this time with a focus on the evaluation of selecting one enterprise architecture over another, on a scenario-based approach. Moreover, the authors discussed how different software architecture styles are used extensively, while there are no standards or best practices on how to evaluate whether the architecture has been correctly designed or not. He and Da Xu (2014) continued the study and reported their findings. They surveyed different architecture styles and technologies used for the integration of distributed enterprise applications. In particular, they presented the advantages and disadvantages of each style along with identifying research trends and needs in that research area. In terms of future perspectives, they identified four trends including (i) Quality-of-Service (QoS) for effective integration as web services, SOA and ESA are increasingly used in integrating applications, (ii) data mining is increasingly developed and deployed for information system integration as data use is increasing exponentially in applications, (iii) integrating web technologies with social networking is expected to grow and (iv) it is expected to see an increase in terms of integrating industry applications with services running on networked, resource-limited mobile devices and smart embedded devices using SOA approaches. The article is concluded with identifying several research challenges in this area, such as user interface integration, reliability, performance management, security risk management, among others.

In 2011, a group of researchers identified the benefits of enterprise architecture to organizations Tamm *et al.* (2011). In particular, they reviewed the current literature on such benefits and consolidated the fragmented knowledge into a model of benefits that is presented in the article. The model proposes that the impact of enterprise architecture leads to four benefits for organizations, namely: Organizational alignment, information availability, resource portfolio optimization and resource complementarity.

In terms of frameworks and in 2004, a patent has been published (US Patent No. 20040133445A1, 2004) in which the authors introduced a generic framework and method to structure, develop and deploy software components to build a multi-tiered enterprise software application. Their high-level framework was based on four sub-components: A client framework, a database framework, a business framework and an external framework to interact with other software processes outside the system.

In 2015, a patent was published (US Patent No. 8996606, 2015) that investigates the dynamic generation of various software components. The patent is based on creating a dynamic decision table for the remote rules engine. This is one of the well-established research studies and patents that was too specific for a small part of the enterprise applications. Moreover, some other research investigations explored creating applications and graphical user interfaces from metadata and models, such as the one presented in Kulkarni and his colleagues work Kulkarni *et al.* (2002). In that research, the authors presented their work of a set of notations to specify different layers of software architecture. Also, they presented a method of transforming a specification into an implementation. They supported their proposed method with some case studies in constructing medium and large-scale enterprise applications. However, utilizing a metadata-driven approach for building such applications can reduce the risks dramatically. In fact, it can enable higher quality and consistency Kiswani *et al.* (2017b).

To the best of our knowledge, no research found that was detailed enough or implemented to cover building an end-to-end framework directed toward enterprise information applications and dynamic components generation.

Although Rajan's and his co-inventors patent (US Patent No. 20040133445A1, 2004) seems close to our work from the higher-level, it does not include details nor clear specifications (e.g., UML, models, diagrams... etc.) on implementation aspects in general or end-to-end in particular. Also, the authors believe that it is somehow stale for modern software application practices.

Furthermore and to the best of the author's knowledge, most of the research studies related to software framework discuss frameworks as a process rather than a higher-level of software abstraction for building reusable functionalities for making the development of software systems relatively faster. However, in this literature review, interesting results were found about components and SOA of enterprise systems, in which they have been reused in the work presented in this work.

Methodology

The approach followed in this work began with analyzing the enterprise applications structure from top to bottom. The rationale behind that was extracting the main characteristics and attributes of every component on every level. Then design for a metadata model to define these characteristics as an Entity-Relationship (ER) model is proposed. The proposal is supported with designing and implementing a fully-working framework and a set of APIs' and tools, to utilize the metadata, towards an effective development of enterprise information systems dynamically on the fly during the runtime.

Evaluating Smart-EIS was challenging, because of the need to implement it on an enterprise level software system, which is most likely consists of a considerable number of database tables, user-interface views and complex business rules. Therefore, we decided to base our evaluation of Smart-EIS on UMS (now known as Solid-SIS) (http://solidsis.com/). UMS is a university management system as well as a student information system that has been developed by Solid-Soft (http://www.solid-soft.net). It is currently implemented in more than 20 academic institutions in the Middle East.

Proposed Framework

This section introduces the proposed work; it provides a detailed explanation and a discussion of the framework architecture, design, implementation and usage.

Scope

The scope of this work is to design and architect a software framework towards making developing Enterprise level Information Systems (EIS) more effective, secure and reliable. Smart-EIS includes management of domain-specific functionalities with its database schema and data-sources. However, EIS are most likely requires integration with third-party applications, either to expose functionality or to be integrable and interoperable Bass *et al.* (2012), in what so-called Enterprise Application Integration (EAI). This requirement can become complicated when there is a need for integration with different systems as well as different protocols and techniques. This is where the Enterprise Service Bus (ESB) becomes useful. Moreover, ESB can also be used with SOA rules engine (e.g., jBPM) to automate the business process at the runtime. Although important, the concepts of EAI, ESB and SOA rules engine require detailed and comprehensive studies and research work that we believe to be beyond the scope of this work.

On the other hand, there are a variety of front-end technologies used in enterprise applications, such as desktop, web and mobile. In this work, desktop front-end is selected for implementation and prototyping. Due to the main author's experience in this field. However, we believe that the work presented in this article can also be implemented as monolithic or cloud native web, cloud, mobile, or IoT front-ends.

Technology and Tools

In Smart-EIS, yED graph-editor was used for diagramming metadata modeling (https://www.yworks.com). In addition, Visual-Paradigm community edition (https://www.visual-paradigm.com) was used for UML class and sequence diagramming. For the development technology, Java (https://www.java.com) was mainly selected because it is standard, open-source and platform independent. Last but not least, MySQL was used as the database management system (https://dev.mysql.coml).

It is worth mentioning that although the authors have used Java technology for the implementation and MySQL as a DBMS to evaluate Smart-EIS framework, the proposed design and architecture can be implemented in any other programming language, since it is based on standard design pattrns and architectural styles.

Naming Convention

For the sake of readability and consistency, we used a unified naming and coding convention for every part of Smart-EIS, such as metadata models, database objects (tables and fields), XML and source code.

In Database naming conventions, table names are plural with an underscore between words and prefixed with the module abbreviation. The prefix makes it easier to group the tables in DBMS clients by the module, especially with the large number of tables in EIS. For example, the name of a students table in a registration module shall be reg students.

Field names should be singular, prefixed with the table abbreviation and separated by spaces. The prefix is used for the join statements with other related tables. The student first name field, for example, should be std first name.

Moreover, every table should have a unique auto-increment primary key that is invisible to the application's end users. The main advantage of this key is to make sure that it is not affected by the business changes. For example, assume that the student id number shall be unique as per the university regulations, the system has been designed based on this rule and every relationship with every detailed table has been configured on this primary key. If this university's regulations change over time to make the primary key unique per year only, the whole system and constraints will become invalid. Furthermore, using a hidden auto-increment key will reduce the development cost by making it easier for database and application developers to create join and DML SQL statements on single fields instead of multiple keys in case of compound keys.

Even though most of the above discussed conventions can be implemented on most of the current DBMS, some features should be handled in particular approaches in some DBMS. For example, Oracle DBMS does not provide an auto-increment feature out of the box; however, this can be achieved using an Oracle sequence object.

For metadata, ER-Models supported by normal readable words for more clarity with the capitalization of the first letter only.

For the source code, we used the standard Java coding convention Oracle (1999).

Proposed Metadata Models

The proposed conceptual ER-model for EIS is shown in Fig. 3. The application consists of many modules. Each module, in turn, consists of user interface views, a database schema, reports and module-specific services. Furthermore, the application consists of SOA-engine to manage the services orchestration in building high-level processes.

All these components have special characteristics and attributes in which will be represented in the following section as metadata models. These metadata models utilized later to be the base for the framework and its APIs.

General Metadata

Some metadata entities and attributes are common across all the components. To avoid duplicating them, we have put them as separate re-usable entities models that can be referenced by other models. The description of this metadata is not duplicated in following diagrams unless they were important to mention. If it is mentioned in next diagram, it will be highlighted in a green color, to distinguish them from entity attributes. Figure 4 shows the diagram of the general metadata attributes, Table 1 provides a brief description of these attributes and Table 2 describes the attributes of the class-definition that is related to the metadata.

Application Metadata

This subsection includes the high-level attributes and entities of the enterprise application. As seen in Fig. 5 the application consists of the general metadata described in the previous subsection. In addition, the application metadata have default language for an application, default database schema and other fields described in Table 3 to 5.

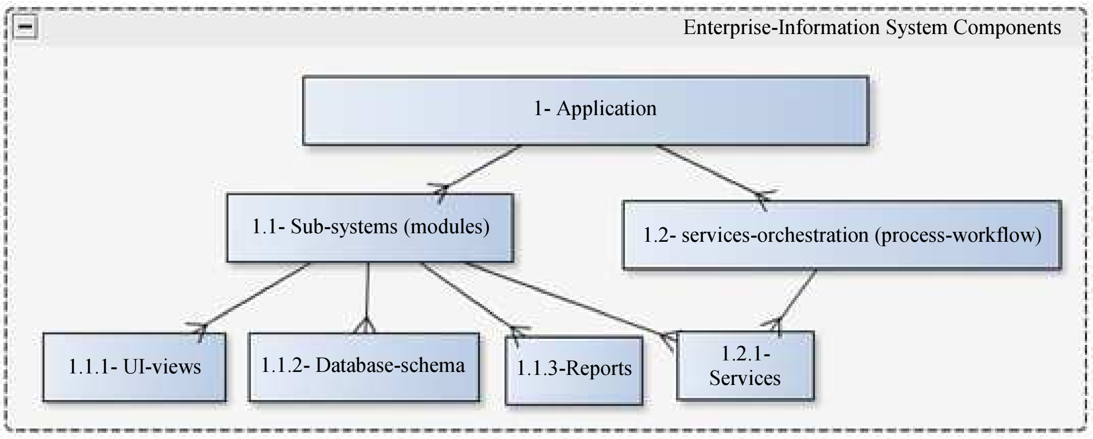
Enterprise-Information System Components
1- Application
1.1- Sub-systems (modules)
1.2- services-orchestration (process-workflow)
1.1.1- UI-views
1.1.2- Database-schema
1.1.3-Reports
1.2.1- Services

Fig. 3: Common components of the enterprise-information system

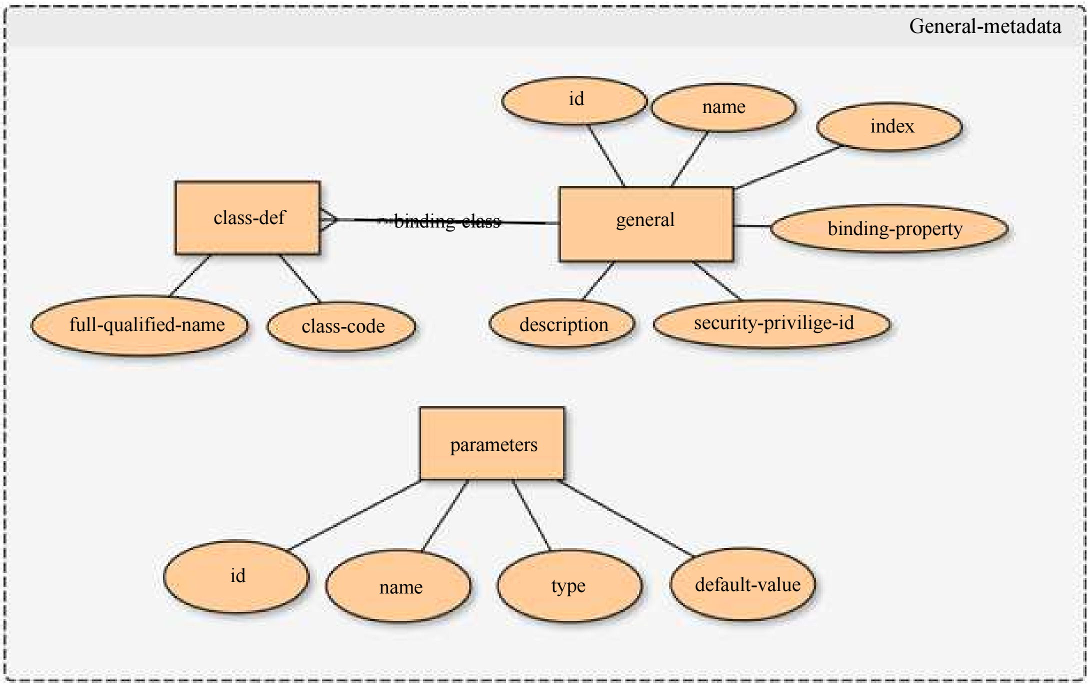

Fig. 4: General metadata

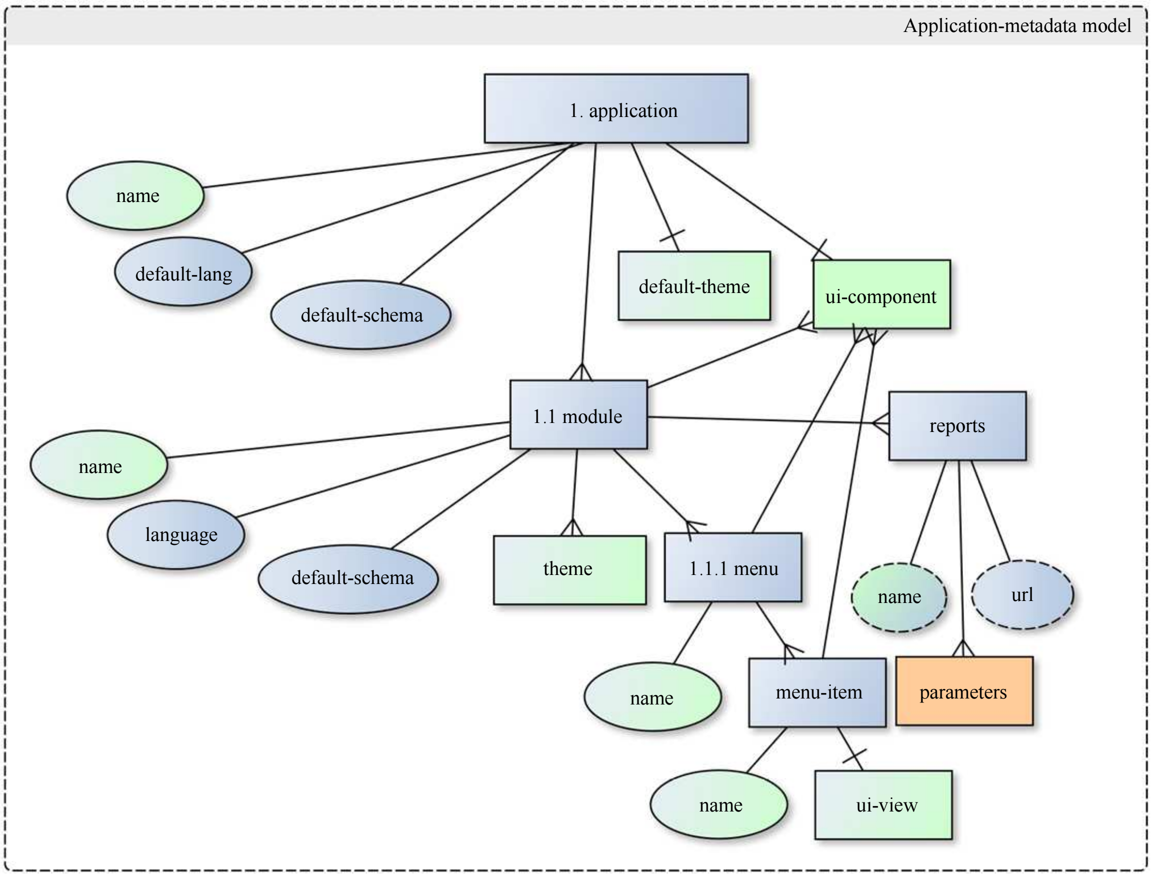

Fig. 5: Attributes and entities of the application metadata model

Table 1: Description of the general metadata

Name	Description	Related model
Id	Unique id for the component	N/A
Name	Name of the component	N/A
Description	Component description	N/A
Index	Order of the component	N/A
Binding-class	The code-class that is component is bounded to Class-definition	N/A
Binding-property	The name of the bounded property in the binding-class	N/A
Security-privilege-id	The id of the security privilege	N/A

Table 2: Description of the class-definition components

Name	Description	Related model
Full-qualified-name	The class name to be loaded at runtime	N/A
Class-code	The source code of the class if it was written in dynamic languages (e.g. groovy, Scala or JavaScript)	N/A

Table 3: Description of the attributes of the application-level metadata

Name	Description	Related model
Default-language	The default language of the application, this will be the language international short code (e.g., EN), country code can be added later as well for more flexibility.	N/A
Default-schema	The default database schema for the application to, this can be overridden on the module level	N/A
Default-theme	The name of the default application theme, this can be overridden on the application level	UI. theme
Default-language	The default language of the application, this will be the language international short code (e.g., EN), country code can be added later as well for more flexibility.	N/A
Default-schema	The default database schema for the application to, this can be overridden on the module level	N/A
Default-theme	The name of the default application theme, this can be overridden on the application level	UI. theme

Table 4: Description of the attributes of the module metadata

Name	Description	Related model
Language	The language of this module	N/A
Theme	The theme of this module	N/A
Splash-image	The URL of theme splash image which will be shown during theme loading	N/A
Home-image	The URL of the home image, which will be shown in the application's main window	N/A

Table 5: Description of the attributes of the module metadata

Name	Description	Related model
URL	The URL for the report	N/A

Table 6: Attributes of the UI-view

Name	Description	Related model
UI-static-view	The model of static UI-views	N/A
UI-dynamic-views	The model of dynamic views which is generated at runtime	N/A

Table 7: Attributes of the UI-static-views

Name	Description	Related model
View-URL	This contains the relative or absolute URL for the view, it could also be used for URL of internal classes packaged as API's or libraries	N/A
Class-definition	The view-executable code If the view is configured at runtime	N/A
UI-inputs	List of UI-input components in this view which can be configured at runtime to change the application behavior	UI-input

Menu-items will be discussed in details in the next subsection.

User-Interface Metadata

As depicted in Fig. 6 and described in Table 6, a user interface view can be either static or dynamic. Static pages are either written as a static code or using scripting languages.

In a static code, pages can be accessed using a URL (i.e., the URL can link to external and internal resources as well) whereas static pages written in scripting languages can be executed at runtime. On the other hand, dynamic views can be built on the fly for various types of components. They can be used to build views for managing database tables, services, or processes, which are all discussed in detail in the following subsections. In the case of building views for managing database tables, the framework will be able to dynamically build a full CRUD operation for the master and details tables by providing the required table-meta id to the model. Moreover, the framework will be able to build the view dynamically based on the metadata available at runtime. The same concept applies to the services and processes dynamic screens, where the name of the required component is provided to the framework, which uses the metadata, in turn, to render the user interface view. Table 7 to 10 provide a description for each of the attributes of the user interface metadata.

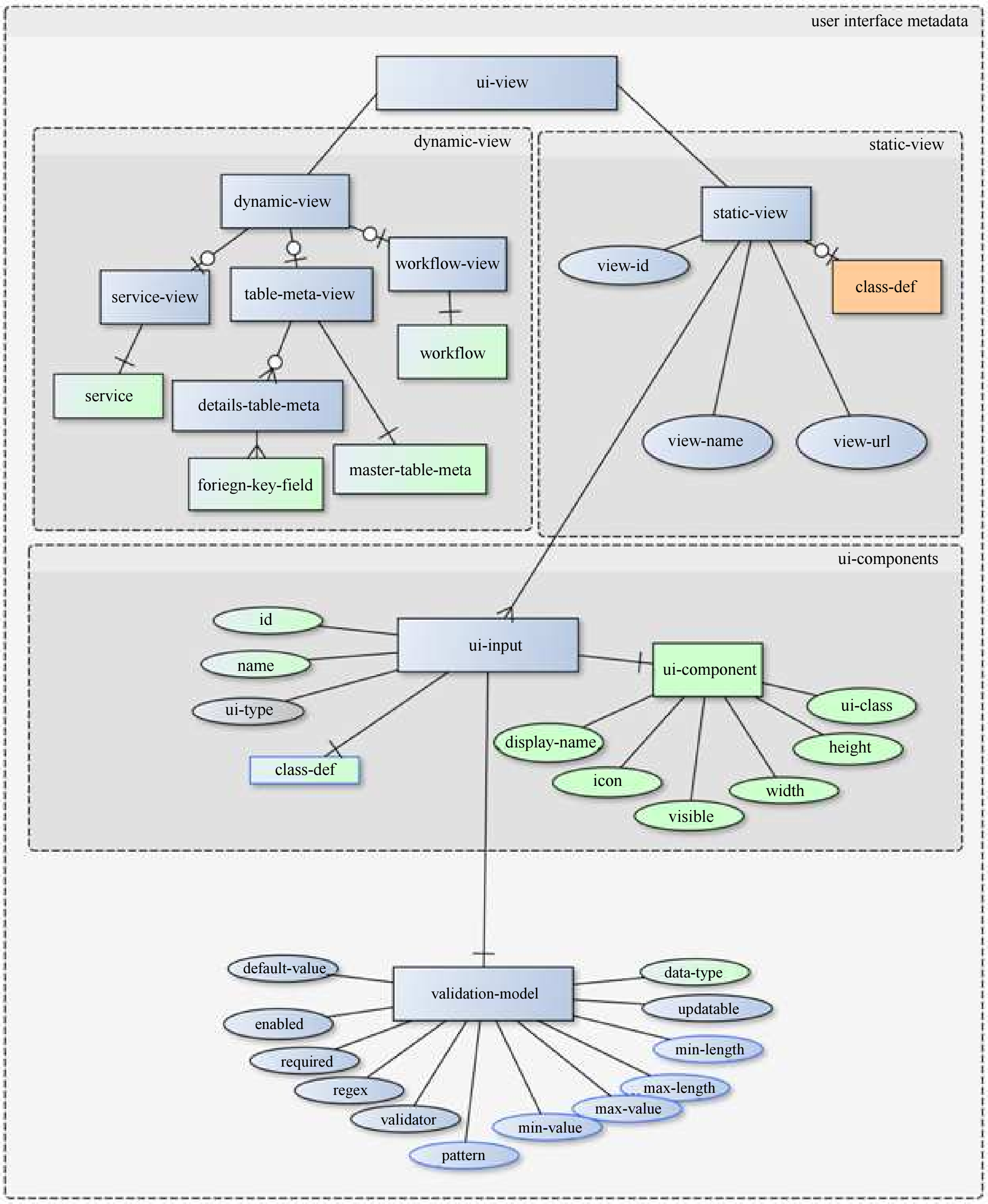

Fig. 6: User interface metadata

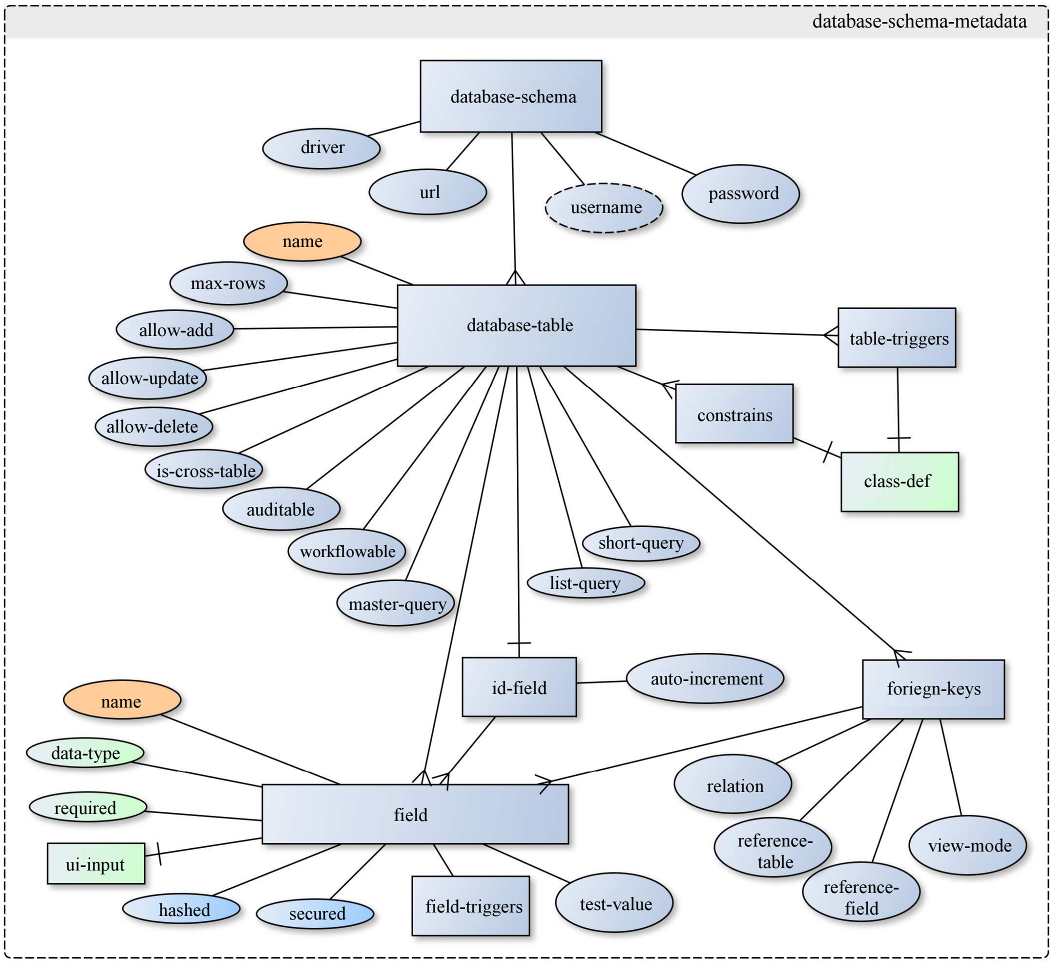

Fig. 7: Metadata of the database schema

Table 8: Attributes of the UI-Input

Name	Description	Related model
UI-type	The data type of input value to be able to create the appropriate UI-component with the appropriate masking	N/A
Visible	Whether visible or hidden	N/A
Icon	The icon of the input	N/A
Width	UI-proposed width	N/A
Height	UI-height	N/A
Validation-model	The validation model of this input UI-validation	N/A

Table 9: Attributes of the UI-validation

Name	Description	Related model
Data-type	The data type of the component	N/A
Default-value	The default value of the component	N/A
Enabled	Whether enabled by default or not	N/A
Required	Required or optional	N/A
RegEx	The regular expression used for validation	N/A
Validatordefinition	If programming validator is set	N/A
Pattern	The pattern used to format the component	N/A
Min-value	Allowed minimum value	N/A
Max-value	Allowed maximum value	N/A
Min-length	Allowed minimum –length	N/A
Max-length	Allowed maximum length	N/A
Updatable	Whether allow to update value or not	N/A

Table 10: Attributes of the UI-theme

Name	Description	Related model
Resource-path	The path to theme resources	N/A
Splash-image	The URL of theme splash image which will be shown during theme loading	N/A
Home-image	The URL of the home image which will be shown in	N/A

Table 11: Database-schema

Name	Description	Related model
Driver	The driver will be used to connect to the database	N/A
URL	Database URL	N/A
Username	Database username	N/A
Password	Database password	N/A
Database tables	The list of database tables of this database	Databasemetadata

Table 12: Table-Metadata

Name	Description	Related model
Id-field	The table primary key	Id-field
Field	The database fields metadata	Field
Max-rows	The allowed maximum number of rows in this table	N/A
Allow-add	Whether adding records to this table is allowed or not	N/A
Allow-update	Whether modification of this table is allowed or not	N/A
Allow-delete	Whether delete of this table is allowed or not	N/A
Is-cross-table	Is this table is cross-join between two tables in many-many relationship or not	N/A
Master-query	The query of this table that will be used in master-view of this table	N/A
List-query	The query of this table that will be used in look-up view of this table when used as foreign-key lookup	N/A
Short-query	The query of this table that will be used when fast search inside dialogs	N/A
Constraints	The constraints that should be applied on this table	Classdefinition
Table triggers	The triggers that should be called when operation are called on this table	Classdefinition
Workflowable	Does this table support a workflow	N/A
Auditable	Weather this table will be auditable or not	N/A

Table 13: ID-Field

Name	Description	Related model
Autoincrement	Weather this field is auto-increment or not	N/A
Field	Field attributes	Field

Table 14: Field

Name	Description	Related model
Field-name	The field-name	N/A
Data-type	Field data type	N/A
Validation-model	Field-validation-model	Validation
UI-general	General UI attributes	Ui-general
Field-triggers	The callback triggers on the field	N/A
Hashed	This attribute will make the framework able to hash the value on insert or update, its most likely used for storing password fields	N/A
Secured	This attribute is used to make the fields encrypted in the database.	N/A

Table 15: Foreign-key-field

Name	Description	Related model
Field	Field metadata	N/A
Reference-table	Reference table Table-meta	N/A
Reference-field Field	Reference-field	N/A
Relation	Relationship type whether one-one, one-many, many-to-one, or many-many	N/A
View-mode	The view-mode (list, dialog, lookup)	N/A

Database-Schema Metadata

This subsection describes the required metadata for the database schema. Most of these metadata are already available in the DBMS metadata. However, the authors proposed a new metadata related to the user-interface, triggers and other components as seen in Fig. 7 and described in Table 11 to 15.

Process-Workflow Engine

The proposed design for the business processes and services is shown in Fig. 8. However and as described earlier in the Section 4.1, they will not be included in the implementation.

Framework UML-Diagram

This section contains the architecture and design of the framework as UML diagrams.

High-Level Package Diagram

Figure 9 depicts the internal package structure of the framework and its relationship with applications, including the dependency between the packages.

The metadata package can be seen at the bottom of Figure 9. This package contains the metadata information and the repository to be accessible by other packages. On the upper level, there are the static and dynamic packages. The static resources compose and utilize some of the dynamic features in the dynamic packages to reduce the development cost. The dynamic package, on the other hands, contains the most significant aspects of the framework, which are used to generate different types of dynamic objects for the applications. These include UI (for dynamic views), data-access, services, auditing, security test-cases and others.

Metadata Class Diagrams

This section includes the class diagrams of the Metadata for Smart-EIS.

High-level Metadata Classes

The class diagram shown in Figure 10 shows the highest level of meta-data abstraction. It starts by the metadata interface, in which every class contains the metadata information to be implemented. In addition, it defines the composition rules between the different high-level metadata where the application contains information about the modules and SOA. The modules, in turn, contain metadata about menus, schema, reports and services. The SOA Metadata, on the other hand, contain the processes metadata, which include service callers that uses the services.

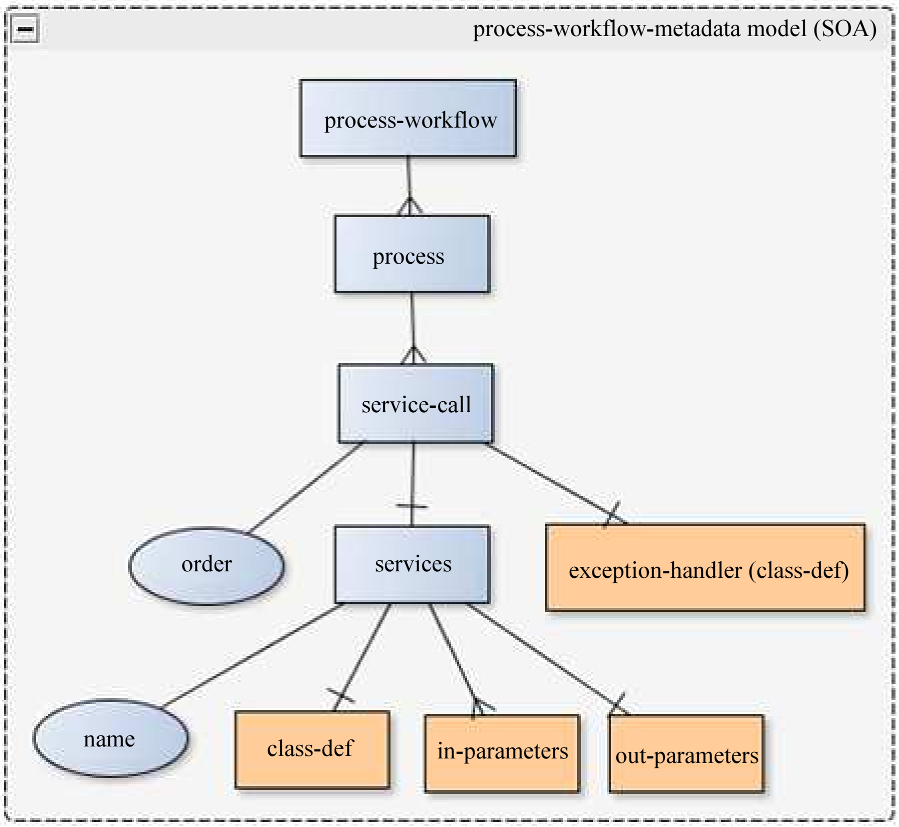

Fig. 8: Process-workflow engine

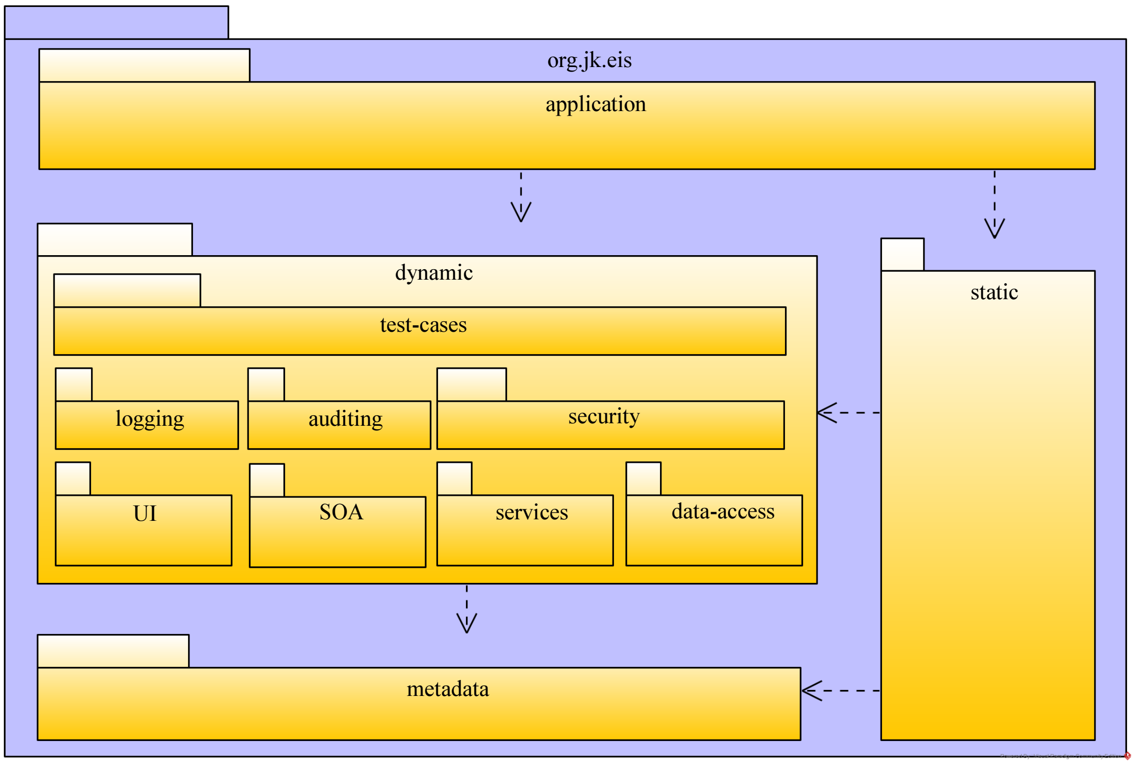

Fig. 9: High-level package diagram of Smart-EIS

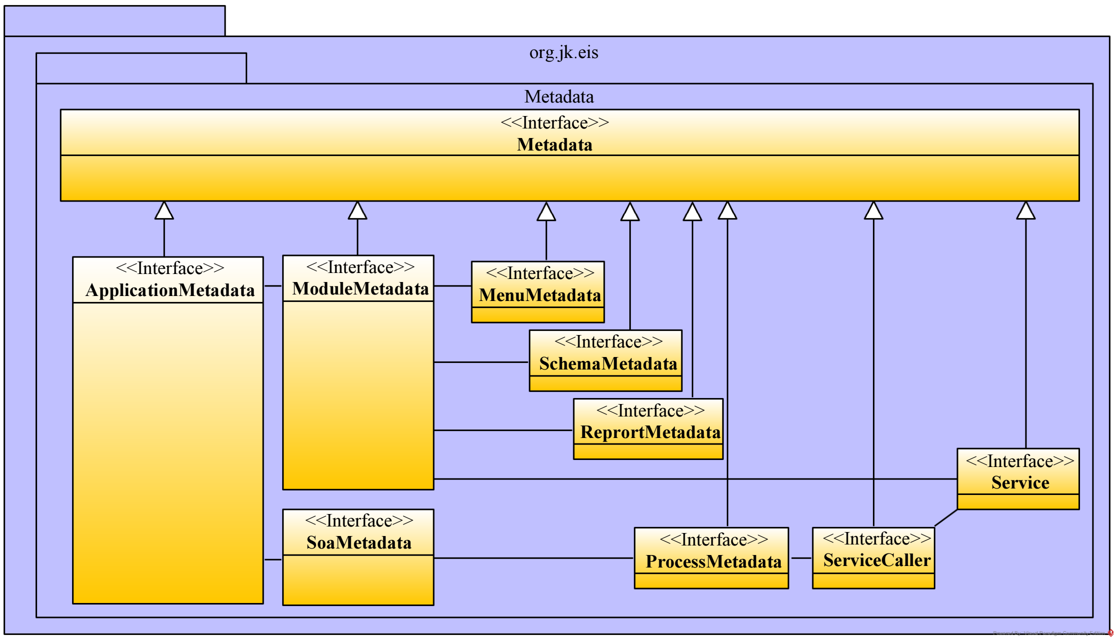
org.jk.eis
Metadata
<<Interface>> Metadata
<<Interface>> ApplicationMetadata
<<Interface>> ModuleMetadata
<<Interface>> MenuMetadata
<<Interface>> SchemaMetadata
<<Interface>> ReprortMetadata
<<Interface>> Service
<<Interface>> SoaMetadata
<<Interface>> ProcessMetadata
<<Interface>> ServiceCaller

Fig. 10: High-level metadata classes

Schema Metadata Class Diagram

Figure 11 contains the schema metadata. From top-to-bottom, it starts with schema metadata, which compose an instance of schema configuration that contains the schema runtime configurations. In addition, it contains the relationships as described in the metadata ER-Model in the previous section.

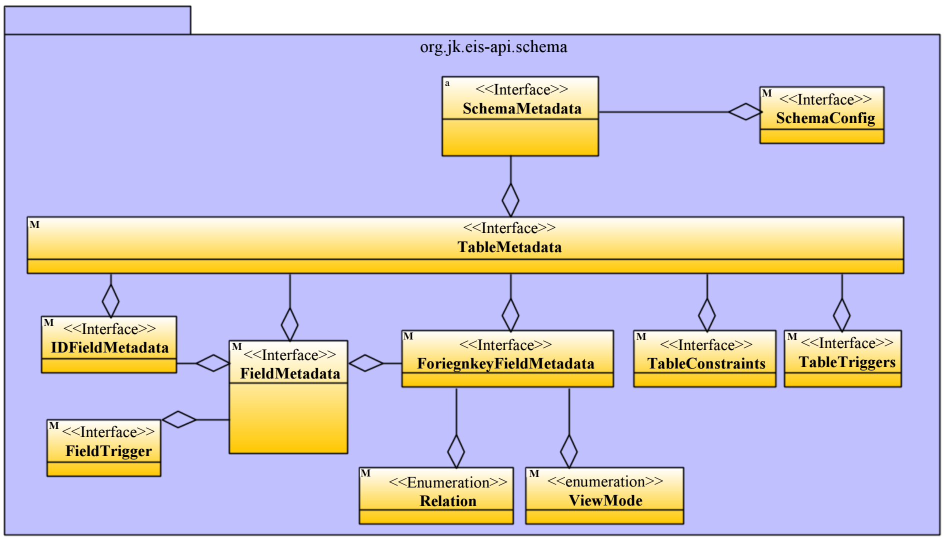
org.jk.eis-api.schema
<<Interface>> SchemaMetadata
<<Interface>> SchemaConfig
<<Interface>> TableMetadata
<<Interface>> IDFieldMetadata
<<Interface>> FieldMetadata
<<Interface>> ForiegnkeyFieldMetadata
<<Interface>> TableConstraints
<<Interface>> TableTriggers
<<Interface>> FieldTrigger
<<Enumeration>> Relation
<<enumeration>> ViewMode

Fig. 11: Schema metadata class diagram

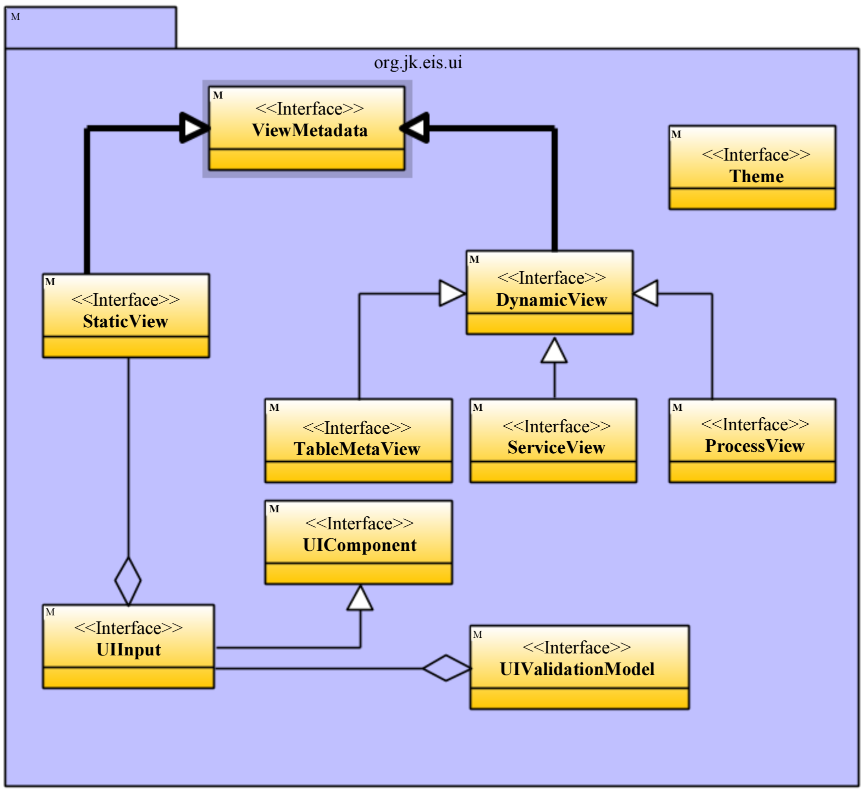
org.jk.eis.ui
<<Interface>> ViewMetadata
<<Interface>> Theme
<<Interface>> DynamicView
<<Interface>> StaticView
<<Interface>> TableMetaView
<<Interface>> ServiceView
<<Interface>> ProcessView
<<Interface>> UIComponent
<<Interface>> UIInput
<<Interface>> UIValidationModel

Fig. 12: Views and UI metadata class diagram

Views and UI Metadata Class Diagram

Figure 12 shows the class diagram for the views metadata. It contains the metadata for static and dynamic views, which is composed of the menu metadata. The figure also shows the relationship with the schema metadata information.

Application Management Classes

The high-level functional side of Smart-EIS is shown in Fig. 13. It starts with the application manager class which is responsible for the application lifecycle (initialize, start, stop, reload). It uses the application metadata to be able to build the application. Moreover, it utilizes its state depending on the metadata repository, which uses metadata loader to load the metadata configurations.

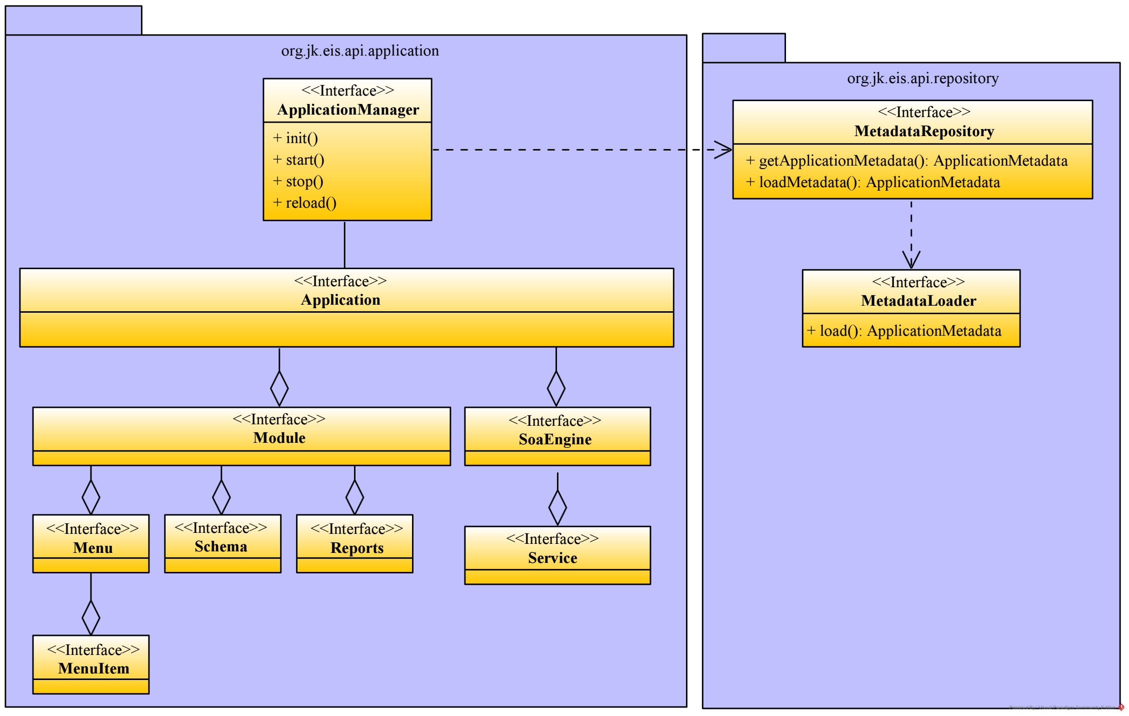
org.jk.eis.api.application
<<Interface>> ApplicationManager
+ init()
+ start()
+ stop()
+ reload()
<<Interface>> Application
<<Interface>> Module
<<Interface>> SoaEngine
<<Interface>> Menu
<<Interface>> Schema
<<Interface>> Reports
<<Interface>> Service
<<Interface>> MenuItem
org.jk.eis.api.repository
<<Interface>> MetadataRepository
+ getApplicationMetadata(): ApplicationMetadata
+ loadMetadata(): ApplicationMetadata
<<Interface>> MetadataLoader
+ load(): ApplicationMetadata

Fig. 13: Application management classes

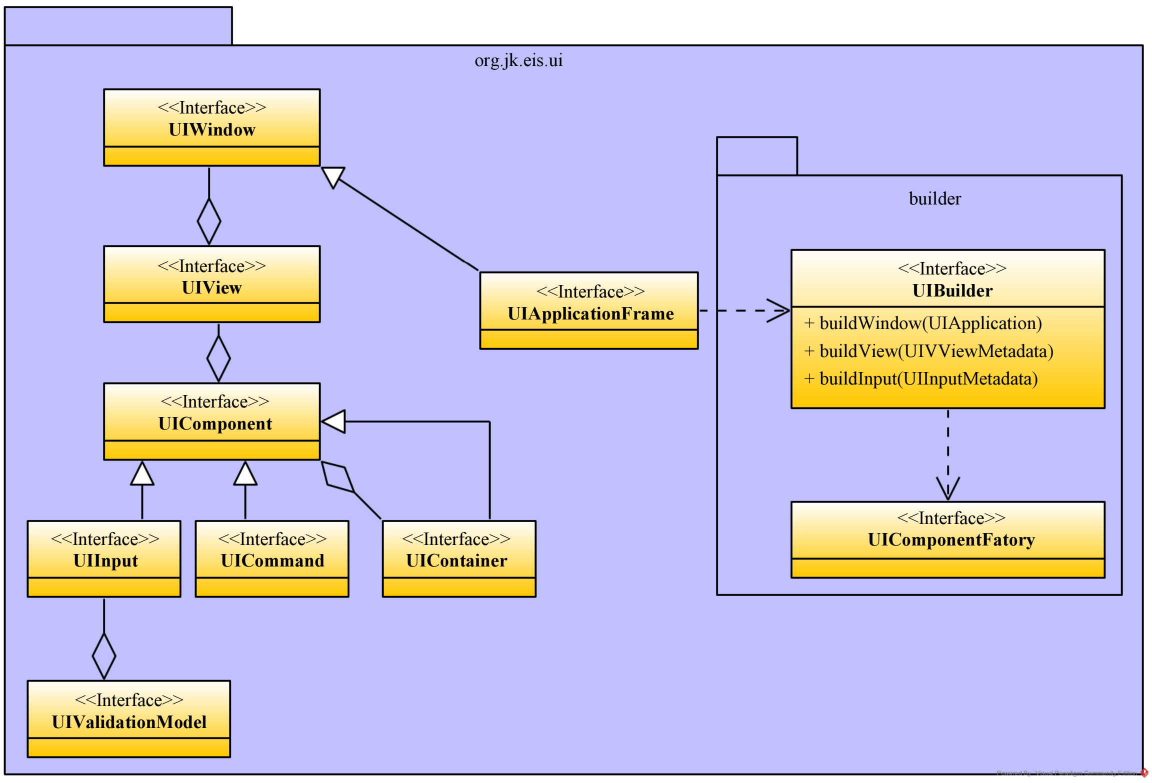
org.jk.eis.ui
<<Interface>> UIWindow
<<Interface>> UIView
<<Interface>> UIComponent
<<Interface>> UIInput
<<Interface>> UICommand
<<Interface>> UIContainer
<<Interface>> UIValidationModel
<<Interface>> UIApplicationFrame
builder
<<Interface>> UIBuilder
+ buildWindow(UIApplication)
+ buildView(UIVViewMetadata)
+ buildInput(UIInputMetadata)
<<Interface>> UIComponentFatory

Fig. 14: User-interface management classes

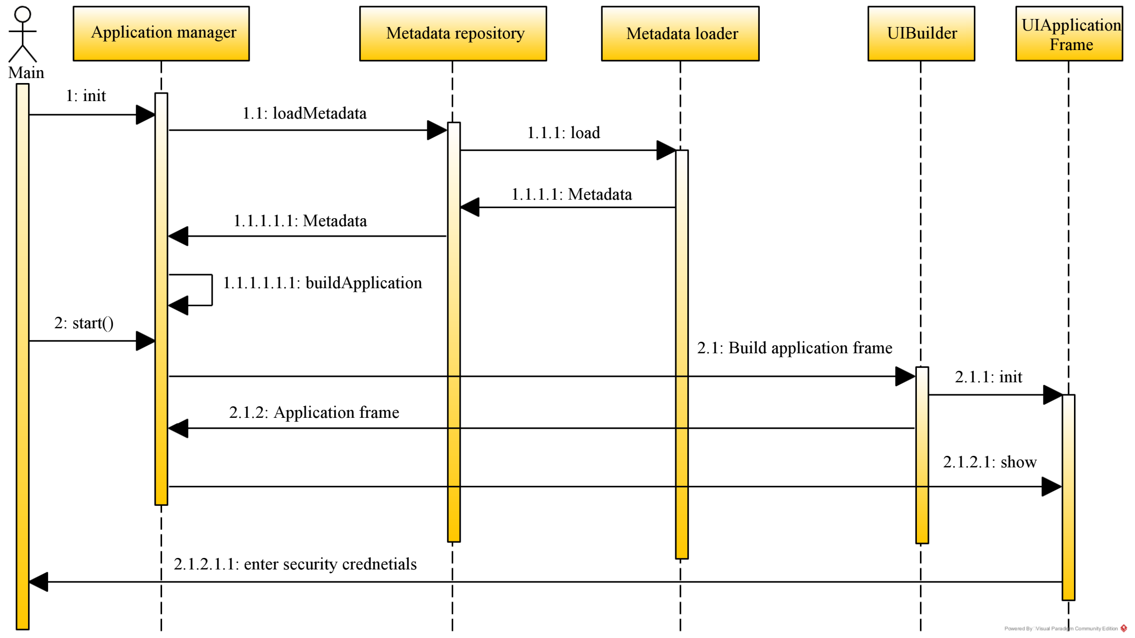

Fig. 15: Application Startup and Authentication

User-Interface Management Classes

Figure 14 shows the class diagram of the process of building the views using the UI Builder class. This class builds the application main windows, views (static or dynamic) as well as input components. The input components building is delegated to the UIComponentFactory.

Application Metadata Bootstrap Sequence Diagrams

Figure 15 shows the sequence diagram of the application metadata bootstrap. In particular, it shows the application startup and authentication through the Application Manager, Metadata Repository, Metadata Loader, UIBuilder and UIApplicationFrame.

Framework Features

The following subsections list the main features of Smart-EIS discussed in detail throughout this chapter. The features are categorized in terms of development and usability. Development:

1. Dynamic generation of configuration views: In most IS applications, most of the database tables are configuration tables that require only simple CRUD operations, using Smart-EIS, the CRUD operations functionality for these tables are generated dynamically at runtime.
2. Follow convention-over configurations concept; to avoid overloading the framework users with a significant amount of configuration, we followed the convention over configuration principle, where there are defaults almost of everything. For example, the metadata for database tables is extracted automatically either in development time using the Database Analyzer, or dynamically at runtime using database metadata provided by JDBC API Kiswani (2018a)
3. Ready-made database-driven components; In Smart-EIS, we have designed and implemented a database-driven GUI component to reduce the development cost of views and business logic. For example, data table builds itself dynamically using metadata extracted from SQL statement, which is also filterable, sortable, printable and exportable to Microsoft Excel
4. Full dependency preparation and setup; project setup is dramatically reduced by using the de-facto standard project-object-model – Maven. With Maven the developer only needs to include Smart-EIS dependency and then Maven will automatically resolve all the required internal dependencies, libraries and API's
5. The ability to disable security on developer's machines; one of the challenges of unit-testing is the security setup and navigation. For example, in most systems, if the developer intends to make manual testing for a specific view, they are required to start the application, log in, navigate to the designated view, then test the views. In Smart-EIS, the developer can start the application in development mode so that he/she can access

the security defaults, which are set automatically for him/her and he/she will be able to access the view directly
6. Simple straightforward data-access-API with a single Line of Code (LOC); Using the proposed APIs, such as the data-access API and exception-handling API, shall reduce the required lines of code, which will result in better productivity, readability and maintainability
7. Transparent localization support; the developers do not need to care about the application localization, which includes: GUI design orientation, component labels and reports. All of these are managed transparently by Smart-EIS
8. User Experience and Usability; usability was given special attention in Smart-EIS for both developers and end-users. In particular:

   - Consistency across all views; since all the configurations views are built dynamically and the other GUI components are built using components provided by the framework, this ensures the GUI consistency across the whole system.
   - Item Special GUI container to show for most access views to reduce the navigation time to reach the view
   - Full shortcuts for every GUI-based functionality in the system, such as menus, views and buttons.
   - Both single view and multiple views are supported

Framework Validation and Evaluation

Validation and evaluation of the proposed methodology and Smart-EIS framework have been performed through three phases. In the first phase, the proposed design of the framework and its API libraries were implemented. Next, a simple Human Resource (HR) management system was built using Smart-EIS to show how the proposed framework can reduce the development time and efforts of building such systems Kiswani (2018b). In the last phase, evaluation of Smart-EIS was conducted. The evaluation was based on several metrics that were studied to compare developing a Student Information System (SIS) traditionally with developing the same system using the proposed Smart-EIS framework. Smart-UMS was chosen for this evaluation.

Framework Components Implementation

To empower our theoretical work in this thesis with the practical part, we have implemented the proposed APIs and design, described in detail in Section 4, into opensource projects that were uploaded into GitHub. Every API has been implemented as a separate open source project, to be utilized and reused by interested researchers and practitioners in the open-source and academic community. The main projects are:

1. General utility: https://github.com/kiswanij/jk-util
2. Database API: https://github.com/kiswanij/jk-db
3. Database metadata: https://github.com/kiswanij/jkdb-metadata
4. Dynamic database API: https://github.com/kiswanij/jk-db-dynamic
5. Application metadata: https://github.com/kiswanij/jk-app-metadata
6. Desktop components: https://github.com/kiswanij/jk-desktop
7. Application: https://github.com/kiswanij/jk-application
8. Smart-EIS-Desktop: https://github.com/kiswanij/smart-desktop

Figure 16 contains the high-level architecture design of the implemented APIs.

Evaluating Smart-EIS: A Case Study

To be able to correctly identify the benefits and advantages of using Smart-EIS on enterprise applications, a university management system, UMS (now known as SolidSIS) was selected as a case study (http://solidsis.com/). Figure 17 shows the main menu of UMS. UMS was chosen for the following reasons: (i) It is a mature software with more than ten years of being on the market, (ii) it is implemented in more than 20 academic institutions in the middle east, (iii) it consists of many modules including admission, registration and student finance, which are developed as desktop applications and (iv) it includes web-based modules such as students e-Service, instructors e-Service and evaluation, along with a smart-phone self-service module for students.

The objective of this case study is to compare the originally built UMS (UMS 9.0) with a newer version (UMS 10.0) that has been built using Smart-EIS framework. This comparison was performed in terms of several identified metrics to show the benefits of using Smart-EIS framework in building such systems, as presented in Table 16.

Table 16 shows that the comparison was based on three main metrics. First, the number of manually developed views was measured in both UMS 9.0 and UMS 10.0. The other two metrics measured the number of manually managed tables in the back-end, which were classified into core business tables and technical ones. Core business tables contain the data required for the development of the application domain business logic. Technical tables, on the other hand, contain the data required for low-level functionality, such as security, auditing and logging. By manually, we mean that the developer has to write code to develop the requirements for the front-end (views) as well as the back-end.

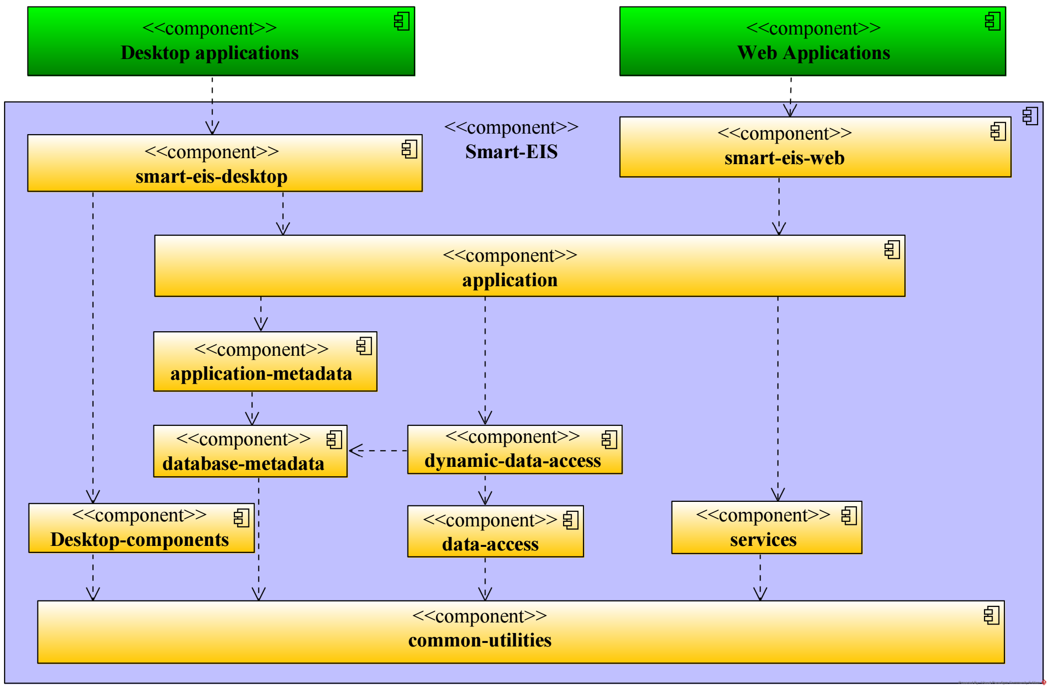

Fig. 16: High-level architecture of the implemented API

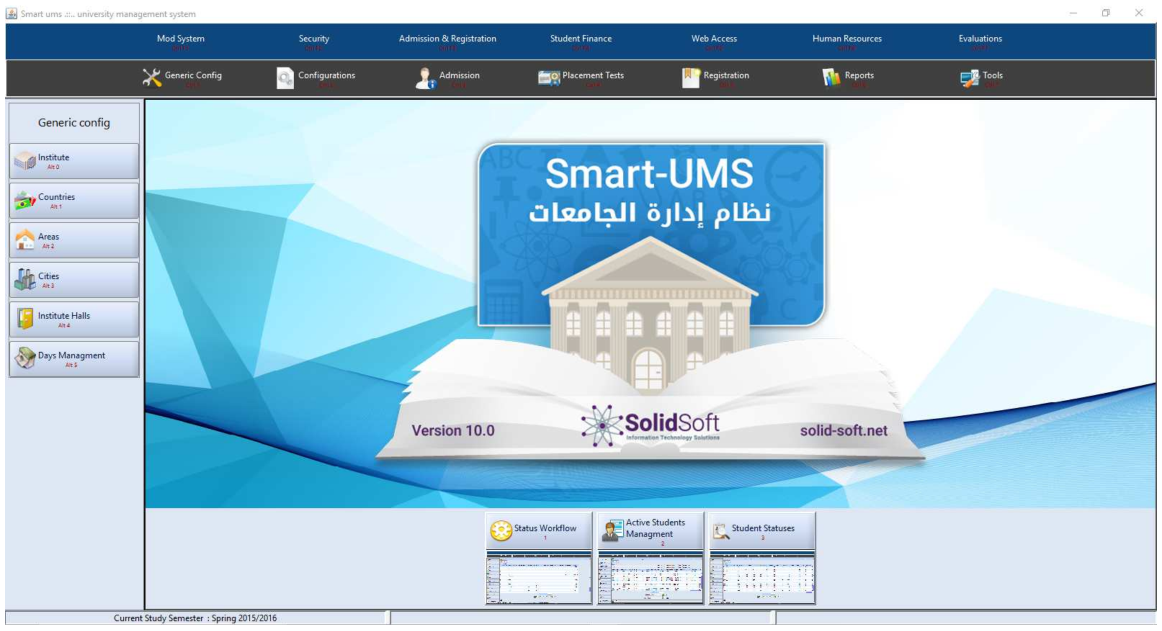

Fig. 17: UMS main window

Table 16: Comparing UMS 9.0 with UMS 10.0

Measured Metric	UMS 9.0	UMS 10.0
Number of manually developed views	395	108
Number of manually managed business tables	247	119
Number of manually managed technical tables	35	0

Discussion

The HR management system was built based on Smart-EIS with only four steps. Therefore, observation shows that using Smart-EIS framework in building such systems should be more rapid and effective than the traditional approach. Moreover, the learning curve of using the proposed framework to build information systems is low in comparison with the required learning curve to build the same system traditionally, where there will be is a need to understand technical and non-technical knowledge as mentioned in the introduction section. In other words, basic knowledge in Java, Maven and MySQL are sufficient to build EIS using Smart-EIS proposed framework.

The evaluation of the proposed framework shows that Smart-EIS had reduced the manual development time and effort of the front-end by 72%. Furthermore, Smart-EIS reduced the manual development time and effort of the back-end by 57%.

The results of the validation and evaluation process shown that Smart-EIS has promising potential in delivering information systems in general and enterprise information systems in particular, rapidly in comparison with the traditional approach. Another potential of the framework can be identified in reducing the development cost as well.

However, the proposed framework may introduce some drawbacks. First, there is a need for specialized expertise in software architecture and development for the customization and troubleshooting of the framework itself. Second, Smart-EIS dynamically generates views with a limited ability to customize their layout, which may limit the flexibility to support custom-designed layouts. Third, the proposed framework has been developed using Java technology, so the applicability of the proposed framework on other technologies is not guaranteed.

Conclusion and Future Work

In this article, an end-to-end software development framework (Smart-EIS) was proposed toward building high-quality enterprise information systems efficiently and effectively. It presented and described a detailed architectural design, models, as well as other design aspects of Smart-EIS. Moreover, open-source projects of the framework's APIs have been implemented to support the design of the framework. Validation and evaluation of the framework have shown promising results, which were discussed as well.

Smart-EIS showed that the framework can reduce the development cost by (i) generating the UI views dynamically using metadata configurations, (ii) following convention over configuration concept, (iii) utilizing database driven UI components, (iv) usage of simple and straightforward APIs that are relatively easy to understand by junior level developers, along with many others. Moreover, user experience and usability have been enhanced by making each view consistent across the whole application. The efficiency of the interface interaction has been enhanced as well through the design and implementation of a smart-detection of frequent-accessed views to make them appear on the home application's window along with full shortcuts on all the views of the system. In addition, support for both single and multiple views management has been presented.

During the literature review phase of the work presented in this thesis, it has been found that there is a lack of research studies and investigations on metadata utilization and dynamic applications generation, which is considered one of the future work directions on this field. We believe that further research in this area will be beneficial for the industry of software development as well as the academic sector.

Another direction of future work has been identified during the application of the framework in the case study. We have observed that the percentage of configuration tables that require straightforward functional management is high (e.g., reached more than 60% in UMS). We believe that a more detailed study is required here to prove the importance of the need for a standard way of handling the configuration tables across technologies dynamically. Moreover and while working on the case study, we have found that there is no standard to generate the technical statistical data dynamically to help in quantitative analysis. We consider this to be another interesting direction of future work.

Finally and since the current trends are apparent in making applications cloud-enabled as well as providing applications that are based on the Software as a Service concept (SaaS), the authors believe that enhancing this framework to be cloud-based and plug-and-play, will reduce the time-to-market and cost of building such applications. It could also be enhanced to support Big data and Internet of things systems.

Acknowledgment

The authors would like to thank the reviewers for their valuable comments and suggestions that contributed to the improvement of this work.

Author's Contributions

Jalal Kiswani: Participated in all experiments, coordinated the data-analysis and contributed to the writing of the manuscript. Contributed substantially to the conception and design of the study, the acquisition of data of the analysis and interpretation.

Muhanna Muhanna: Responsible for the conceptualization of work and its realization, complied the literature sources, interpreted figures and tables and provided final approval of the version to publish.

Abdallah Qusef: Wrote some parts of the manuscript, acted as corresponding author, helped in reference checking and provided critical revision of the article.

Ethics

This work is original and not published elsewhere. The authors confirm that they have read and approved the manuscript and there is no conflict of interest. There is no ethical issue involved in this article.

References

Bass, L., P. Clements and R. Kazman, 2012. Software Architecture in Practice. 3rd Edn., Pearson Education, Upper Saddle River, ISBN-10: 0321815734, pp: 640.

Bianco, P., R. Kotermanski and P.F. Merson, 2007. Evaluating a service-oriented architecture. Technical report, Software Engineering Institute.

Borger, E. and W. Schulte, 2000. A practical method for specification and analysis of exception handling-a java/jvm case study. IEEE Trans. Software Eng., 26: 872-887. DOI: 10.1109/32.877847

Capilla, R., A. Jansen, A. Tang, P. Avgeriou and M.A. Babar, 2016. 10 years of software architecture knowledge management: Practice and future. J. Syst. Software, 116: 191-205. DOI: 10.1016/j.jss.2015.08.054

Cervantes, H. and R. Kazman, 2016. Designing Software Architectures: A Practical Approach (SEI Series in Software Engineering). 1st Edn., Addison-Wesley Professional, ISBN-10: 0134390784, pp: 320.

Da Xu, L., 2011. Enterprise systems: State-of-the-art and future trends. IEEE Trans. Industrial Inform., 7: 630-640. DOI: 10.1109/TII.2011.2167156

De Alwis, B. and J. Sillito, 2009. Why are software projects moving from centralized to decentralized version control systems? Proceedings of the ICSE Workshop on Cooperative and Human Aspects on Software Engineering, May 17-17, IEEE Xplore Press, Vancouver, BC, Canada, pp: 36-39. DOI: 10.1109/CHASE.2009.5071408

Dearie, A., 2007. Software deployment, past, present and future. Proceedings of the Future of Software Engineering, May 23-25, IEEE Xplore Press, Minneapolis, MN, USA, pp: 269-284. DOI: 10.1109/FOSE.2007.20

Do, H., G. Rothermel and A. Kinneer, 2006. Prioritizing junit test cases: An empirical assessment and costbenefits analysis. Empirical Software Eng., 11: 33-70. DOI: 10.1007/s10664-006-5965-8

Fowler, M. and K. Beck, 1999. Bad Smells in Code. In: Refactoring: Improving the Design of Existing Code, Fowler, M., K. Beck, J. Brant, W. Opdyke and D. Roberts (Eds.), Addison-Wesley Professional, Boston, ISBN-10: 0201485672, pp: 75-88.

Fowler, M., 2002. Patterns of Enterprise Application Architecture. 1 Edn., Addison-Wesley Longman Publishing Co., Inc., ISBN-10: 0321127420, pp: 560.

Garlan, D. and D.E. Perry, 1995. Introduction to the special issue on software architecture. IEEE Trans. Software Eng., 21: 269-274.

He, W. and L. Da Xu, 2014. Integration of distributed enterprise applications: A survey. IEEE Trans. Industrial Inform., 10: 35-42. DOI: 10.1109/TII.2012.2189221

Jin, J., S. Sun and X. Bao, 2016. Research of adaptive software design patterns. Int. J. Control Automat., 9: 289-296. DOI: 10.14257/ijca.2016.9.5.28

Johnson, P., 2002. Enterprise software system integration: An architectural perspective. PhD Thesis, KTH.

Kiswani, J., 2018a. JK-DB-metadata-console.

Kiswani, J., 2018b. Smart-EIS desktop HR example.

Kiswani, J., M. Muhanna and A. Qusef, 2017b. Using metadata in optimizing the design and development of enterprise information systems. Proceedings of the 8th International Conference on Information and Communication Systems, Apr. 4-6, IEEE Xplore Press, Irbid, Jordan, pp: 188-193. DOI: 10.1109/IACS.2017.7921969

Kiswani, J., M. Muhanna, S. Dascalu and F. Harris, 2017a. Software infrastructure to reduce the cost and time of building enterprise software applications: Practices and case studies. Proceedings of ISCA 26th International Conference on Software Engineering and Data Engineering, (EDE' 17), pp: 93-98.

Kulkarni, V., R. Venkatesh and S. Reddy, 2002. Generating enterprise applications from models. Proceedings of the International Conference on Object-Oriented Information Systems, (OIS' 02), pp: 270-279.

McGovern, J., S. Tyagi, M. Stevens and S. Mathew, 2003. Java Web Services Architecture. 1st Edn., Morgan Kaufmann, ISBN-10: 1558609008, pp: 832.

Niu, N., L. Da Xu and Z. Bi, 2013. Enterprise information systems architecture analysis and evaluation. IEEE Trans. Industrial Inform., 9: 2147-2154. DOI: 10.1109/TII.2013.2238948

Oracle, 1999. Code conventions for the java programming language.

Royce, W.W., 1987. Managing the development of large software systems: Concepts and techniques. Proceedings of the 9th International Conference on Software Engineering, (CSE' 87), IEEE Computer Society Press, Monterey, California, USA, pp: 328-338.

Schlossnagle, T., 2006. Scalable Internet Architectures. 1st Edn., Pearson Education, ISBN-10: 067232699X, pp: 288.

Sommerville, 2015. Software Engineering. 10th Edn., Addison Wesley.

Stair, R. and G. Reynolds, 2017. Fundamentals of Information Systems. 9th Edn., Cengage Learning, ISBN-10: 1337097535, pp: 560.

Sullivan, A., 2003. Economics: Principles in Action. 3rd Edn., Pearson Prentice Hall, ISBN-10: 0130081515, pp: 712.

Tabatabaie, M., R. Paige and C. Kimble, 2009. Exploring Enterprise Information Systems. In: Social, Managerial and Organizational Dimensions of Enterprise Information Systems, Cruz-Cunha, M.M. (Ed.), ISBN-10: 1605668567, pp: 415-433.

Tamm, T., P.B. Seddon, G.G. Shanks and P. Reynolds, 2011. How does enterprise architecture add value to organisations? CAIS, 28: 10-10.

The Economist, 2008. Let it rise: A special report on corporate IT. Special report, The Economist.

Valipour, M.H., B. AmirZafari, K.N. Maleki and N. Daneshpour, 2009. A brief survey of software architecture concepts and service oriented architecture. Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology, Aug. 8-11, IEEE Xplore Press, Beijing, China, pp: 34-38. DOI: 10.1109/ICCSIT.2009.5235004

Wolfgang, P., 1994. Design patterns for object-oriented software development. Reading Mass.