Conference on Information Technology and Data Science

Technical Improvements of the ILONA System

Zsolt Tóth*

Eszterházy Károly University, Faculty of Informatics, Eger, Hungary
toth.zsolt@uni-eszterhazy.hu

Proceedings of the 1st Conference on Information Technology and Data Science
Debrecen, Hungary, November 6–8, 2020
published at http://ceur-ws.org

Abstract

The Indoor Localization and Navigation (ILONA) System was designed and developed since 2015. During this period, the ILONA System was used to record data data sets, perform experiments and numerous students contributed to its development. Keeping the project up-to-date from the viewpoint of technology is a constant challenge which has both pitfalls and success stories. Collecting data with the ILONA System allowed us to get experience with its usage in real scenarios and set further directions of improvement. Testing different positioning algorithms showed the flexibility of the system. Modular decomposition of the monolithic first version of the ILONA System was successful with some overstatements. Development is an ongoing process with a couple of new features. Applications, experiences and further developments of the ILONA System are both detailed.

Keywords: Indoor positioning, architectural design, ILONA System

1. Introduction

The Indoor Localization and Navigation (ILONA) System [5] was designed in 2016 in order to facilitate the implementation and testing of indoor positioning algorithms. The ILONA System is a web application which provides web services for

*The author's research was supported by the grant EFOP-3.6.1-16-2016-00001 ("Complex improvement of research capacities and services at Eszterhazy Karoly University").

mobile clients and a management interface via web. Extend ability and flexibility were key requirements during the design of the ILONA System. In addition, the recent technology trends were also considered during the design and development. Over the last few years, ILONA System was used to record data sets [4, 6], perform experiments in positioning and implement navigation [2] algorithms as well.

Modifications each have contributed to the development of the ILONA System and the current paper gives an overview of the development process. Based on our experience during the development, some technology, technique and approach were proven useful and beneficial while others were dead ends. The overview of these experience may facilitate the design of other projects, can summarize some good practices and outline further developments of the ILONA System. System wide, component or technology specific and implementation techniques were both used during the development. Currently, modular decomposition, data access technologies and automated code formatting will be presented.

2. Methods

In the following, there is a brief overview of used methods, tools and techniques. The motivation behind their applications is also presented.

2.1. Modules

The ILONA System has a modular design where four subsystems were distinguished. These subsystems have the same architecture in order to standardize them. Each subsystem contains `controller`, `model`, `persist`, `service`, `web` modules that are the common building blocks of web applications. The ILONA project is managed with Maven that is a build tool and dependency manager for java. Maven facilitates the creation of projects and allow to organize them into hierarchy by creation of so called `bundle` projects which are packaged as a `pom`. The `bundle` project can contain more projects so the project hierarchy can be organized. Unfortunately, there is no single, commonly accepted approach when should a new project created, so the size and structure of the project vary between development projects.

The modules and architecture of the ILONA System has four major variants. Initially, the entire code base was represented by a single monolithic project where the subsystems were separated on the level of packages. Then, subsystems were outsourced into separate projects so they had independent development life cycle. Due to the success of separation of subsystems, each subsystem was defined as a `bundle` project and its modules became independent projects. Finally, the subsystems were flattened by merging the modules because the management of the tiny module was costly and difficult.

2.2. Data Access

Data access is a particularly common task in information systems so there are a wide range of tools and techniques. The ILONA System was designed to be independent of any database so data can be stored via Data Access Objects. These Data Access Objects are interfaces which define the so called CRUD methods; create, read, update, delete. The loose coupling of data storage allowed the separation and exchange of implementation due to the Dependency Inversion Principle. The `measurement` subsystem uses MySQL database which was accessed with two different technologies; mybatis and hibernate.

Mybatis is a data access technology which can hide SQL statements behind Java methods by defining them in XML format. The implementation of these interfaces is generated by the mybatis based on the XML description. So mybatis allows us to create custom SQL statements and use them via method invocations. On the other hand, mybatis require the handling and management of the XML descriptions which may be confusing.

Hibernate is an Object–Relational–Mapping tool for Java which can store, fetch, update and delete objects via repositories. The stored objects must be properly annotated and they are called entities. Although there is a need for conversion of model and entity objects both of these kind of objects are defined by Java classes and the SQL is hidden. The conversion and then handling of the repositories are hidden behind the Data Access Objects which gives one more indirection into the data handling.

2.3. Code Style

Coding conventions and styles are vital for huge projects with developers with various experience. Unfortunately coding conventions were overlooked during the first years of the project and automated checking tools was recently introduced into the project. Although Implementation Patterns [1] and Clean Code [3] were given as guidelines but nor rigorous style check or automated tools were used. As result, the code base is pretty heterogeneous but the main interfaces were already defined. In addition, some naming convention and coding styles were copied through the projects due to the existing code base.

The lack of automatic code analysis has not caused any problem yet but it would standardize the code quality. Enforcing these coding standards is essential for the project due to the following two reasons. First, some technologies should be changed because they became deprecated or no longer supported so there is a need for refactoring. Second, students are contributing in the development of the ILONA system and their coding skill could be improved by the standard. The code quality is ensured by the application of `checkstyle` and its integration into the version control process.

3. Results

The experimental results with modules, data access techniques and code style checking are summarized in this section.

3.1. Modules

Modular decomposition was successful on the level of subsystems but it was an overstatement on lower levels of the system. The initial success with decomposition made us enthusiastic about modularization which lead to numerous tiny modules. Monolithic systems are easy to design but their maintainability decrease quickly so the separation of subsystems was necessary. Each subsystem contains the same modules that represents a well–defined part of the system. While the subsystems are huge building blocks of the entire system and they can be developed independently, the modules of a subsystem are connected and depend on each other.

Although the subsystems are implemented with the same technologies currently, their separation allows us to use different technologies or programming languages in the ILONA System. For example, the `measurement`, `tracking` and `navigation` subsystems can be efficiently implemented with Java and Spring Framework. On the other hand, our current research is focused on the application of machine learning techniques so using Python would be easier in the `positioning` subsystem. Because each subsystem has an own development life cycle and they can be deployed separately these subsystems can be virtualized with containers which is the base of micro service architecture.

Separation of the modules seemed to be a great idea but the dependencies between the modules made the development slow. Developers should simultaneously work on different project because each module was represented as a project. Hence they made more error and the build process was also longer because each module had separate build processes. In addition, the separation of build process was unnecessary because the build was usually started from the root of the subsystem. Consequently, these modules were merged back into the subsystem which is represented as a single project currently.

3.2. Data Access

Data access was recently changed in the `measurement` subsystem from mybatis to Hibernate. This modification was performed in three major steps. First, the Hibernate based implementation was added to the subsystem while the mybatis implementation was still used. Then the system was configured to use the Hibernate based implementation which required only the modification of a few configuration files. Finally, the mybatis was removed from the project in a refactoring phase. Changing mybatis to Hibernate was a good experience because it showed the benefits of loose coupling and Hibernate is currently more widely supported.

Loose coupling of data storage is part of the ILONA System from the initial version. The definition and implementation of the data access object seemed to be an unnecessary step because the system supported only a single technology. Services used the definition of the data access objects while the implementation was initialize during when the application started. This separation served educational purposes too. During the implementation of the hibernate based data access, the interfaces and the expected behavior were already defined which set the expectations about the implementation. In addition, the implementation was tested with unit tests and allowed experiments with the entire system too. Furthermore, the migration from the mybatis implementation was quite simple because it required only a few modifications in the configuration classes to initialize the Hibernate based solution.

Hibernate seems to be dominant data access technology in the local software industry therefore migrating to it was necessary. Mybatis was chosen for data access because it was popular when the development started. In addition, based on our experience Hibernate has a way better documentation and support in forum and tutorials which facilitate the development. Finally, Hibernate is part of the curriculum which helps students to participate in the development of the ILONA System and makes their onboarding process easier.

3.3. Code Style

Automatic code checkers were introduced recently in order to standardize the code quality. In the first few years the students had no standard but they had to follow some recommendations. As a result, the following three observations were made; naming the variables depends on the developer, they organize their code differently and the code quality decreases when the deadline is getting closer.

Naming a variable or a function is a difficult processes even for experienced developers. Using proper names increase the code readability and facilitates its maintenance. Due to the lack of their experience, the students chose different name for similar tasks or objects. In general, they adopted the practice to introduce a `result` variable in each method with the return type and this variable is returned in the function.

4. Discussion

Development of the ILONA System is an ongoing process with technical challenges and changes. Some experience with the development were summarized and organized in this paper. These observations could be useful for other projects and can be used to improve the development too. Architectural, design and implementation level observations were presented in details.

Defining modules is a difficult task and it is strictly related to architectural design of the system. The initial monolithic system was organized into independent subsystems successfully. Then these subsystems were split into projects which made

the development slow. Our experience showed that the zealous modularization could be harmful for the project.

Design of the data access hide the technology and implementation details behind interfaces and used loose coupling. The recent change of technology from mybatis to Hibernate was simple and successful which clearly demonstrated the benefits of this design. The change of technology required only the implementation of the corresponding interfaces and some configuration. As a result, the ILONA System fits better to the current trends in the local software industry.

Implementation patterns and coding standards were neglected in the first years of the development which inevitably leads to poor code quality. Due to the existing code base and the recommended literature the students adapted some good practices although the automatic checking was missing. Proper naming of variables and methods was particularly challenging for inexperienced developers.

To sum up, the 5-years development of the ILONA System yielded important experience in all levels of development.

References

[1] K. Beck: *Implementation patterns*, Pearson Education, 2007.

[2] D. P. Kun, E. B. Varga, Z. Toth: *Ontology based navigation model of the ILONA system*, in: 2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI), 2017, pp. 000479–000484,
doi: https://doi.org/10.1109/SAMI.2017.7880357.

[3] R. C. Martin: *Clean Code-Refactoring, Patterns, Testen und Techniken für sauberen Code: Deutsche Ausgabe*, MITP-Verlags GmbH & Co. KG, 2013.

[4] Z. Tóth, J. Tamás: *Miskolc IIS hybrid IPS: Dataset for hybrid indoor positioning*, in: 2016 26th International Conference Radioelektronika (RADIOELEKTRONIKA), 2016, pp. 408–412,
doi: https://doi.org/10.1109/RADIOELEK.2016.7477348.

[5] Z. Tóth: *ILONA: indoor localization and navigation system*, Journal of Location Based Services 10.4 (2016), pp. 285–302,
doi: https://doi.org/10.1080/17489725.2017.1283453.

[6] Z. Tóth, P. Magnucz, R. Németh, J. Tamás: *Data model for hybrid indoor positioning systems.*