Application Framework Development and Design Patterns: Current State and Prospects

Marko Jurišić
Infonova GmbH
Lassalestrasse 7A, 1020 Vienna
mjurisic@gmail.com

Dragutin Kermek
Faculty of Organization and Informatics
University of Zagreb
Pavlinska 2, 42000 Varaždin, Croatia
dragutin.kermek@foi.hr

Abstract. *Design patterns are one of the fundamental skills for software developers as they describe best practice solutions for recurring problems. Frameworks provide standardized solutions for a specific application domain.*

This paper provides an overview of design patterns and frameworks and explores the relationship between design patterns and frameworks. Two of the most popular frameworks for building Java web applications were analyzed regarding their design pattern use.

Keywords. Design pattern, Framework, Spring, Java EE

1 Introduction

Design patterns help to identify, name and abstract recurring problems in software development and to identify best practice solutions. Christian Alexander introduced design patterns as a means to easier communicate common concepts that occurred in designing buildings such as corridor, seating place etc. [1]. Design patterns have been used since in many other areas but they have an irreplaceable significance in the field of computer science because they enable experts to exchange ideas and high-level overview of a system in a common language – language of patterns.

One of the introductory books which brought pattern languages to wider audience is a book Design Patterns: Elements of Reusable Object-Oriented Software [2] also known as Gang-of-four or GoF book because this book was written by four authors: Eric Gamma, Richard Helm, Ralph Johnson, and John Vlissides. The GoF book described 23 classic software design patterns dealing with common problems involving object creation, internal structure and behavior concerning communication with other objects.

Design patterns are high-level descriptions of recurring problems and their solutions. They are usually presented in a well-defined format with pattern name, problem and motivation, proposed solution and finally limitations and interaction with other patterns. A pattern has three main characteristics [3]:

• The context is a surrounding condition under which specified problem exists.

• The problem is a difficult and uncertain subject area in the domain. It is limited by the context in which it is being considered.

• The solution is a remedy for the problem under consideration.

Ascending systems complexity and the need for quick and efficient solutions have resulted in a vast number of frameworks. A framework is a set of cooperating classes together forming a reusable design for a specific class of software [2] or a reusable, semi-complete application that can be specialized to produce custom applications by implementing missing functionality or changing predefined parts of the framework [4].

We begin by describing connection between design patterns and frameworks in Chapter 2, continued by framework classification in Chapter 3, give an analysis of most prominent Java frameworks in Chapter 4 followed by brief discussion in Chapter 5. Chapter 6 concludes the paper.

2 Design patterns and frameworks

Since GoF many other authors from the field of computer science addressed design patterns. Notable contributions are Pattern Oriented Software Architecture series which deal with large applications [5], service access and configuration, event handling, synchronization, and concurrency [6], effective resource management in a system [7], distributed computing [8], pattern languages and pattern collections [9]. Design patterns and pattern languages have become one of the crucial elements of education for every software developer.

Relationship between a framework and a design pattern is described in GoF [2]:

1. Design patterns are more abstract than frameworks.
2. Design patterns are smaller architectural elements than frameworks.
3. Design patterns are less specialized than frameworks.

Some frameworks have been implemented multiple times in different programming languages and they can be considered patterns too, for example Model-View-Controller (MVC) [10], first introduced in Smalltalk programming language [11], shown on Figure 1. Framework used dictates the architecture of the application, overall structure, classes and objects, their collaboration and control. The main benefit from frameworks is that they provide modularity, reusability, extensibility and inversion of control which means that the framework controls which classes get called and when [11].

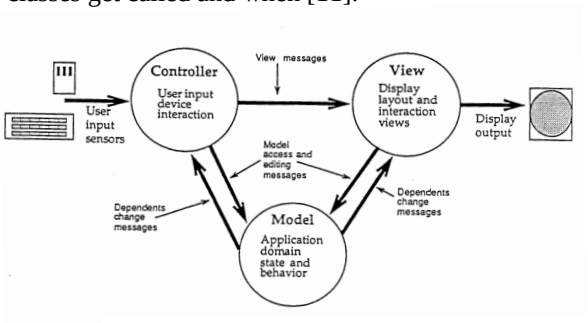

Figure 1MVC pattern [10]

3 Framework classification

Frameworks can be classified by their structure and communication with the rest of the application as white box and black box. White box frameworks contain incomplete or abstract classes and methods which must be overridden in order to achieve desired functionality by either inheriting from base classes or overriding default functionality using patterns such as Template Method. Black box frameworks offer finished components and desired functionality is achieved by composing those components or defining new components that implement interfaces offered by the framework and integrating those components into the framework using patterns like Strategy. Points of adding specific application functionality to framework are called hot-spots [12].

Another way of framework classification is by their scope [13]:

- System infrastructure frameworks
- Middleware integration frameworks
- Enterprise application frameworks

System infrastructure frameworks simplify the development of portable and efficient applications and are generally used only internally within software organization, they are not offered to customers as separate products. Examples are communication frameworks, user interface frameworks or language processing tools.

Middleware integration frameworks are used to integrate distributed applications and components, enhancing the ability to modularize, reuse and extend the software. Examples are message-oriented middleware or transactional databases.

Enterprise application frameworks are used in broad application domains and are the cornerstone of enterprise business activities. Comparing to other types of frameworks enterprise application frameworks are expensive to develop or purchase and their learning curve is steeper but they can provide a substantial return on investment since they support the development of end-user applications and products directly. These frameworks are best suited for domains where numerous similar applications are built from scratch because they offer a possibility to create working full-featured applications in a very short time.

Developing a framework is vastly different from developing a standalone application because a framework addresses problems that are, at least on the surface, different from those that justified the creation of the framework. If the application development is hard, toolkit development harder, framework development is hardest of all [2]. Design pattern catalogs essentially attempt to pick out frameworks that are not too domain-specific [14] and describe them as patterns via established pattern language.

The most profoundly elegant framework will never be used unless the cost of understanding it and then using its abstractions is lower than the programmer's perceived cost of writing them from scratch [15]. Overuse of design patterns and multiple layers of abstraction obscure many implementation details and sometimes it becomes impossible to understand particular design decisions without hints or detailed documentation [4].

Sometimes applications are based on multiple frameworks that have to be integrated with one another as well as with class libraries and existing legacy components which brings new problems because most frameworks are designed with the assumption of full control over event loop [16].

Most modern frameworks use the convention over configuration approach. Convention over configuration is a software design paradigm which reduces necessary effort required to achieve standard functionality by providing standardized methods. Flexibility is not reduced because standard behavior can be easily overridden through additional configuration [17], usually via property files so that the application must not be recompiled or redeployed for every configuration change.

4 State of the art Java Web frameworks

Java has been used for building of all kinds of applications in different domains and there are

GWT	Google Web Toolkit - development toolkit for building and optimizing complex browser based applications, used internally in Google products such as AdSense. Developers write code in Java which is then converted to optimized JavaScript which runs in all browsers.
JSF	Java Server Faces – part of Java EE standard for building server-side user interfaces
Play	Available for Java and Scala, based on a lightweight, stateless architecture and features predictable and minimal resource consumption (CPU, memory, threads) for highly-scalable applications
Spring	Core support for dependency injection, transaction management, web applications, data access, messaging, testing and more for a wide range of applications
Struts	Apache Struts is a free, open-source, MVC framework for creating elegant, modern Java web applications. It favors convention over configuration, is extensible using a plugin architecture, and ships with plugins to support REST, AJAX and JSON.
Vaadin	Built on top of GWT, has large number of ready-to-use interface components, controls and widgets, supports client-side and server-side programming model
Wicket	Mark-up/logic separation, a POJO data model, powerful, reusable components written with plain Java and HTML

Table 1 Overview of Java Web Frameworks

frameworks that facilitate all kinds of tasks. In this section we focus on web and enterprise frameworks.

During Java history many frameworks were popular at a time but later faded out. Today we have a choice of frameworks with similar features but different focuses which can be used to build comprehensive web applications. Some of them are listed in Table 1 for reference

We focus on Java EE&JSF and Spring framework in the rest of the chapter because they are currently the most widely used and offer most additional features.

Enterprise Java platform has gone through many fundamental changes, going from bloated and difficult to develop infrastructure of J2EE to a lightweight standardized solution of Java EE [22].

Java EE comes with many features that are supported out of the box such as transactions via Java transaction API - JTA, object persistence via Java persistence API – JPA, scalability through stateless beans managed by container, security, messaging etc. Web pages are usually made with backing beans and JSF for frontend using one of many faces libraries which bring additional user interface elements.

Java Server Faces (JSF) technology is a server-side user interface component framework for building Java technology-based web applications. It is part of the Java EE specification and is considered the standard solution for building web-based user interfaces.

Java Server Faces consists of the following [23]:

- An API for representing components and managing their state; handling events, server-side validation, and data conversion; defining page navigation; supporting internationalization and accessibility, and providing extensibility for all these features – it is possible to write new components or to combine existing components for easier reuse
- Tag libraries for adding components to web pages and for connecting components to server-side objects

Java Server Faces technology provides a well-defined programming model and various tag libraries. There are also numerous extensions and additional tag libraries which bring new components such as sortable and editable data tables with automatic paging and various widgets. Some examples are ICEfaces, MyFaces, OpenFaces, Primefaces, RichFaces.

JSF is a part of the Java EE specification focused on building user interfaces but a quick source code analysis shows that most of the GoF patterns are used throughout the framework with their original names. Some of the patterns that are not mentioned specifically by name but are used nevertheless are State and Observer patterns – among the main features of JSF are automatic preservation of state between requests and updating of visual components when backend data changes.

J2EE was intended to solve problems associated with distributed application development. The development process with J2EE platform was very complicated and difficult to test because components could not run outside of container (this problem has been solved in newer versions of Java EE), so development community started to build mainly open source alternatives for elements in the J2EE software stack such as Struts which is based on servlet API or Hibernate which deals with object persistence.

Spring, built on the idea of overcoming the limitations of J2EE, offers a full stack solution, providing out-of-the-box components and integrating best single-tier frameworks such as Hibernate. Spring framework assembles best and well-tried open source solutions and brings a full stack solution. Spring programming model is based on POJOs (plain old Java objects) which allows easier testing of software using automated unit tests for single class testing or for integration tests starting up the test application context and testing entire workflows.

Spring framework is a Java platform that provides comprehensive infrastructure support for developing Java applications, handling infrastructure so that developers can focus on the actual application [18]. It was built on top of Java EE technologies and created with a notion of simplifying development of Java enterprise applications, aiming to make J2EE easier to use. One of Spring's main benefits is its layered architecture which allows selecting only the needed modules. Spring framework consists of about 20 modules which are grouped into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, and Test, shown in Figure 2. The system is very flexible because modules can be used independently which enables developers to use only some modules and build the rest of the enterprise application with non-spring components e.g. using JSF for Web GUI.

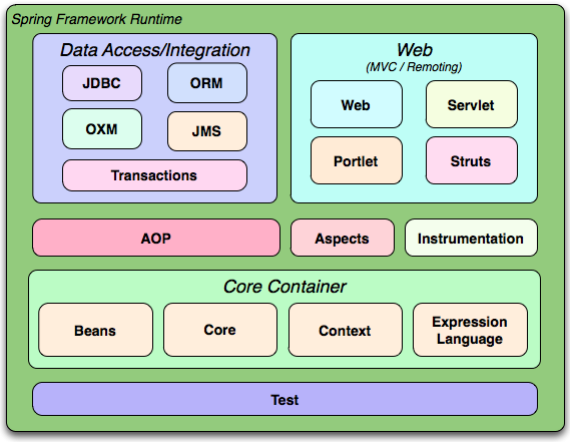

Figure 2 Spring framework overview [16]

Spring framework was created with vision of providing best practice solutions for common problems so it is basically a collection of state of the art design patterns. Overview of J2EE design patterns grouped in several tiers (presentation tier, business tier, integration tier) and their implementations with Spring framework is given in [3]. J2EE had a set of established patterns for different business tiers of which some are still used, although a change of technology and new versions of Java EE have made most of them obsolete or brought new solutions which work out-of-the box [19].

Central pattern used in Spring is Inversion of control (IoC), also called Dependency Injection (DI) [20], although IoC can be viewed as one of the possible ways of achieving DI. Usually in applications the developer controls which objects are instantiated, which methods are called and when. Inversion of control inverts this principle releasing the control of some aspects to the framework – framework calls the application code and manages the flow of control. The principle is usually described in literature as "Hollywood principle" i.e. "Don't call us, we'll call you" [21]. It is used in many other frameworks, notably Google Guice and Java EE platform since version 5.

A quick analysis of Spring source code shows that most of GoF patterns are frequently used. This is especially true for creational and structural patterns which deal with object creation and internal structure, the only exception being Flyweight pattern but since all most classes in Spring are POJOs Flyweight pattern is implicitly used. The same conclusion can be made about Chain of responsibility – it is not explicitly named or documented but one of the primary Spring functions is forwarding requests between different application aspects via various controllers.

5 Discussion

Any nontrivial solution is built using one or more application frameworks. Today nobody writes their own object-relational-mapping or model-view-controller framework because there are high-quality

	Java EE	**Spring**
Server type	Java EE application server such as Jboss/WildFly	Web container (Tomcat/Jetty) or Java EE application server
Dependency injection	JSR-330, CDI	IoC
Aspects	Interceptors/Decorators	Spring AOP, AspectJ
Persistence	JPA2	JPA2, Hibernate
Transactions	JTA, EJB 3.1	JTA, JDBC, JPA, Hibernate, JDO
Presentation framework	JSF 2	JSF2, Spring MVC
Web Services	JAX-WS, JAX-RS	JAX-WS, Spring MVC Rest, JAX-RPC, XCF, Axis
Messaging	JMS, EJB 3.1	JMS, Spring Integration
Testing	CDI, EJB 3.1, JPA2	JUnit, TestNG
Security	JAAS	Spring security
Scheduling	EJB 3.1	Quartz
Batch processing	JSR-352	Spring batch

Table 2 Java EE and Spring side by side [27]

frameworks available and most of them are completely free for commercial use. There are many advantages of using tried frameworks – not many organizations can afford months or years of framework development for skeleton that does not bring business value – it is much easier and cheaper to use existing and well tested frameworks which have proven their worth in thousands of systems. One downside is a sometimes steep learning curve but if the company does development with common frameworks it is easier to find developers that already have experience in their use.

In enterprise Java landscape the most prominent are Java EE reference implementation which comes bundled with Java EE Application servers such as WildFly or Glassfish and Spring which can run on any servlet container such as Tomcat or Jetty.

Java EE is advertised as a standard solution but that is also the main pitfall – Java Community Process achieving consensus about what goes in the specification is a long term process. The process of defining Java EE 8 features is currently in progress - Java EE 8 Community Survey shows that the most wanted features in Java EE 8 are JCACHE, Java API for JSON binding, standardization of server-side events, MVC alternative to JSF, security interceptors, standardization of logging, embedded web and Java EE container and pruning of legacy technologies such as CORBA and EJB 2.x [24] but it might take years before they are accepted and implemented and a few more years before we can use them in production systems.

Open source frameworks such as Spring are free to grow and adapt to current technology trends in a much more agile way. Spring already has built-in support for noSQL databases, Android components, Social Network support, and even Java 8 support on the day of the official release [25].

Although there are frequent heated debates about which framework is better and more suited for specific needs both frameworks can be used together, one can for example use Spring for the business layer and JSF with backing beans for the frontend. Final functionality of both approaches is similar; Table 2 shows a brief overview. Best ideas are exchanged and end up in both frameworks, sometimes only with small changes – one example is JSR-352/Spring batch [26]. Balanced competition is always good for both developers and vendors.

6 Conclusion

Design patterns have become one of the essential tools in software developer's toolbox. Knowing the best practice patterns and being able to recognize when and how to apply them makes solving difficult problems easier. Another important aspect of design pattern usage is facilitation of communication between developers because design patterns help to convey complex architectural ideas in a well-established jargon and avoid misunderstanding.

Frameworks such as Spring or Java EE enable developers to create new applications quickly and those applications have better quality than applications based on frameworks developed in-house because they can build on thousands of hours invested in developing, testing and debugging framework code. Framework does the heavy lifting and takes care of common aspects such as transactions and security and developers can concentrate on solving specific business problems. Another advantage of using well known frameworks is that it very likely that new developers will already have knowledge of the framework and will sooner be able to contribute to the project.

Spring and Java EE will continue to co-exist and exchange ideas and best practices. Spring is by definition more agile and free to pick up the trends via versatile Spring projects which can quickly cover any emerging technologies while Java EE as standard implementation can always count on large enterprise support from the likes of Oracle, IBM and RedHat.

Using standardized frameworks and design patterns facilitates communication between developers and also speeds up the development process while increasing the quality of applications so that the additional effort spent on learning the frameworks and design patterns usually pays off quickly.

References

[1] C. Alexander, S. Ishikawa, and M. Silverstein, *A pattern language: towns, buildings, construction*, vol. 2. Oxford University Press, USA, 1977.

[2] J. Vlissides, R. Helm, R. Johnson, and E. Gamma, "Design patterns: Elements of reusable object-oriented software," *Reading: Addison-Wesley*, vol. 49, 1995.

[3] D. Kayal, *Pro Java EE Spring Patterns: Best Practices and Design Strategies Implementing Java EE Patterns with the Spring Framework*. Apress, 2008.

[4] W. Pree and H. Sikora, "Design patterns for object-oriented software development (tutorial)," *Proceedings of the 19th international conference on Software engineering - ICSE '97*, pp. 663–664, 1997.

[5] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-oriented Software Architecture: A System of Patterns, Volume 1*, vol. 1. John Wiley & Sons, 2008.

[6] D. C. Schmidt, M. Stal, H. Rohnert, F. Buschmann, and J. Wiley, *Pattern-oriented Software Architecture: Patterns for*

Concurrent and Networked Objects, Volume 2. Wiley, 2000.

[7] M. Kircher and P. Jain, "Pattern-oriented software architecture volume 3: Patterns for resource management," 2004.

[8] F. Buschmann, K. Henney, and D. C. Schmidt, *Pattern-oriented Software Architecture: a Pattern Language for Distributed Computing, Volume 4*, vol. 4. John Wiley & Sons, 2007.

[9] F. Buschmann, K. Henney, and D. C. Schmidt, "Pattern-oriented software architecture: On patterns and pattern languages, vol. 5." John Wiley & Sons Inc, 2007.

[10] R. Johnson, "Components, frameworks, patterns," *ACM SIGSOFT Software Engineering Notes*, no. 217, pp. 1–23, 1997.

[11] G. E. Krasner, S. T. Pope, and others, "A description of the model-view-controller user interface paradigm in the smalltalk-80 system," 1988.

[12] W. Pree and D.- Constance, "Essential Framework Design Patterns."

[13] M. Fayad and D. Schmidt, "Object-oriented application frameworks," *Communications of the ACM*, 1997.

[14] W. Pree, "Framework development and reuse support," 1994.

[15] G. Booch, "Designing an application framework," *Dr Dobb's Journal-Software Tools for the Professional Programmer*, vol. 19, no. 2, pp. 24–35, 1994.

[16] M. Mattsson, J. Bosch, and M. Fayad, "Framework integration problems, causes, solutions," *Communications of the ACM*, vol. 42, no. 10, 1999.

[17] N. Chen, "Convention over configuration," *http://softwareengineering. vazexqi. com/files/pattern. htm l*, 2006.

[18] R. Johnson, J. Hoeller, A. Arendsen, C. Sampaleanu, R. Harrop, T. Risberg, D. Davison, D. Kopylenko, M. Pollack, T. Templier, and others, "The spring framework-reference documentation," *Interface21.(accessed 30.04. 07)*, 2008.

[19] A. Bien, *Real World Java EE Patterns Rethinking Best Practices*. 2009.

[20] M. Fowler, "Inversion of control containers and the dependency injection pattern." 2004.

[21] R. Johnson, "J2EE development frameworks," *Computer*, vol. 38, no. 1, pp. 107–110, 2005.

[22] A. Bien, *Real World Java EE Night Hacks*. .

[23] Oracle Corporation, *The Java EE 6 Tutorial*, no. January. 2013.

[24] Results from the Java EE 8 Community Survey, 28.04.2014 *[https://java.net/downloads/javaee-spec/JavaEE8_Community_Survey_Results.pdf]*

[25] J. Hoeller, *Java 8 in Enterprise Projects*, 21.03.2014 *[https://spring.io/blog/2014/03/21/java-8-in-enterprise-projects*]

[26] D. Woods, *Java EE 7, Spring Standardize Batch* 21.03.2014 *[http://www.infoq.com/news/2013/06/ee7-spring-batch*]

[27] R. Rahman, Spring and Java EE Side by Side, 2013 *[http://de.slideshare.net/reza_rahman/java-ee-and-spring-sidebyside]*