

Contents

- [1 Introduction](#)
- [2 Three types of patterns](#)
- [3 Design Patterns Definition](#)
- [4 Patterns Format](#)
- [5 Why Design Patterns?](#)
- [6 Pattern Language](#)
- [7 Catalog and Organization of Design Patterns](#)
- [8 Mobile Internet Technical Architecture \(Nokia MITA\)](#)
- [9 Problems of Mobile Application Development](#)
- [10 Patterns Supporting Mobile Application Development](#)
- [11 Architecture](#)
- [12 Patterns of Enterprise Application Architecture](#)

Introduction

This article focuses on the important facts on how Design Patterns support mobile application development.

Three types of patterns

- Architectural Patterns ? An architectural pattern expresses a fundamental structural organization for software systems or schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them.
- Design Patterns ? A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes commonly recurring structure of communicating components that solve a design problem within a particular context.
- Idioms - An idiom is a low-level pattern specific to a programming language. An idiom describes how to implement particular aspects of components or the relationships between them using the features of the given language.

Design Patterns Definition

Design Patterns are developed in the context of architectures. There are three types of Architectures Building architecture, Software architecture, and Enterprise application architecture. Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice. Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a

solution. As an element in the world, each pattern is a relationship between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain spatial configuration which allows these forces to resolve themselves. As an element of language, a pattern is an instruction, which shows how this spatial configuration can be used, over and over again, to resolve the given system of forces, wherever the context makes it relevant. Pattern language is extremely practical... It is a language that we have distilled from our own building and planning efforts over years.. The elements of this language are entities called patterns.

Patterns Format

- **Pattern Name and Classification** ? The pattern's name conveys the essence of the pattern succinctly. A good name is vital, because it will become part of your design vocabulary.
- **Intent** ? A short statement that answers the following questions: What does the design pattern do? What is its rationale and intent? What particular design issue or problem does it address?
- **Also Known As** ? Other well-known names for the pattern, if any.
- **Motivation** ? A scenario that illustrates a design problem and how the class and object structures in the pattern solve the problem. The scenario will help you understand the more abstract description of the pattern that follows.
- **Applicability** ? What are the situations in which the design pattern can be applied? What are examples of poor designs that the pattern can address? How can you recognize these situations?
- **Structure** ? A graphical representation of the classes in the pattern using a notation based on the Object Modeling Technique. Additionally interaction diagrams are used to illustrate sequences of requests and collaborations between objects.
- **Participants** ? The classes and/or objects participating in the design pattern and their responsibilities.
- **Collaborations** ? How the participants collaborate to carry out their responsibilities.
- **Consequences** ? How does the pattern support its objectives? What are the trade-offs and results of using the pattern? What aspect of system structure does it let you vary independently?
- **Implementation** ? What pitfalls, hints or techniques should you be aware of when implementing the pattern? Are there language-specific issues?
- **Sample Code** ? Code fragments that illustrate how you might implement the pattern in e.g. C++ or Smalltalk.
- **Known Uses** ? Examples of the pattern found in real systems. Gamma et al. include at least two examples from real domains.
- **Related patterns** ? What design patterns are closely related to this one? What are the important differences? With which other patterns should this one be used?

Why Design Patterns?

- Patterns can be used by programmers, designers, and architects who are building applications and who want to improve either their understanding of architectural issues or their communication about them.
- Patterns are common solutions to recurring problems. If you have worked in applications for a while, you may well know most of them. They are industry's old ideas. If you are new, pattern book can help you learn about those techniques. If you are familiar with the techniques, pattern book can help you communicate and teach them to others. An important part of patterns is trying to build a common vocabulary for communication.
- Patterns constitute an effort to build on the collective experience of skilled designers and software engineers.
- Experts already have solutions to many recurring design problems.
- Patterns capture proven solutions in an easily-available .
- Patterns support both novices and experts in software development.
- Common vocabulary
- For designers to use to communicate, document and explore design alternatives. Design patterns reduce system complexity and increase design efficiency by letting people talk about their designs at a higher level of abstraction than that of a design notation or programming language. Without this vocabulary they would always have to put effort on finding descriptions for the problem at hand. The common language will also span team boundaries making it even more universal.
- Documentation and learning aid
- Knowing design patterns helps understand existing systems. As patterns provide solutions to common problems, they thereby help novices act more expert-like and allow them to produce effective designs. Furthermore, when designs are presented in terms of patterns, outsiders (members of other teams or even organizations) that are previously not familiar with a particular system should be able to get a rapid higher-level understanding and overview of it.
- Promotion of design reuse
- Design reuse is different from e.g. code reuse; the former is typically more efficient than the latter as design is more abstract and is therefore more likely to apply in a number of situations. An effective design is, however, harder to come by than an effective implementation: a particular implementation may be fully functional but that alone does not guarantee it's understandable to others and reusable (c.f. program code)
- An adjunct to existing methods

Understanding Design Patterns

- Object-oriented design methods are, in themselves, supposed to promote good design by teaching newcomers proven design principles, and by standardizing the way designs are done. Design patterns add to these qualities by providing a means of describing more of the "why" of a design as opposed to just recording the results of one's decisions.
- A target for re factoring
- One of the problems of developing reusable software is that it often has to be reorganized or re factored. Design patterns help determine how to reorganize a design and can reduce the amount of re factoring later on.

Pattern Language

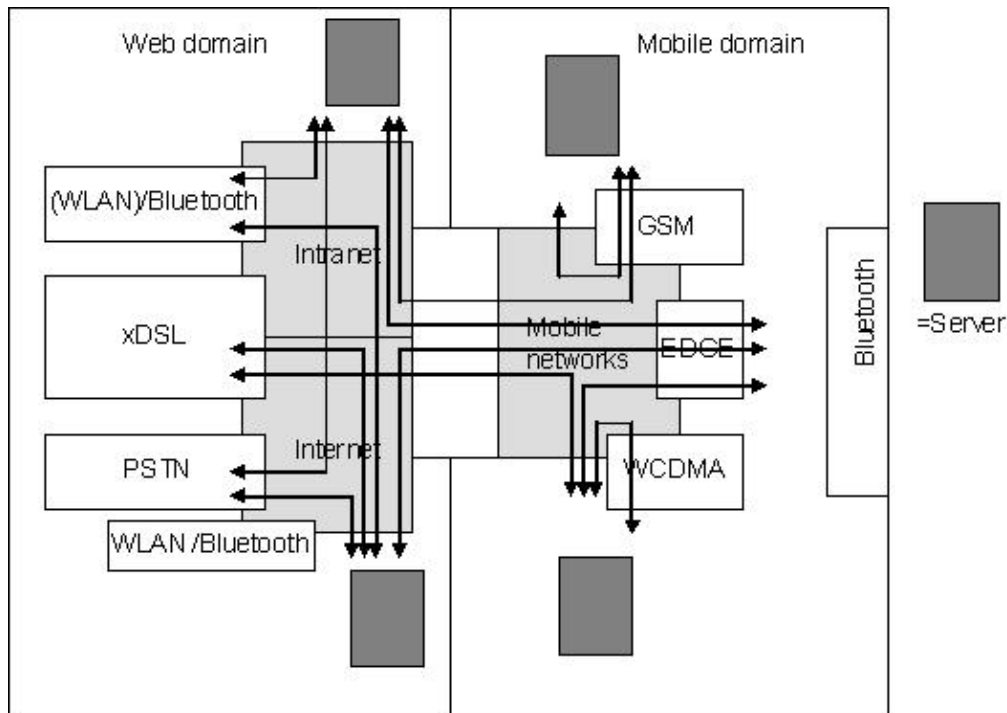
Two essential purposes behind the specific format. To present each pattern connected to other patterns, so that you grasp the collection of all 253 patterns as a whole, as a language, within which you can create an infinite variety of combination. To present the problem and solution of each pattern in such a way that you can judge it for yourself, and modify it, without losing the essence that is central to it. The patterns are ordered, beginning with the very largest, for regions and towns, then working down through neighbors, clusters of buildings, buildings, rooms and alcoves, ending finally with details of construction. We hope that many people try to improve these patterns? Patterns are very much alive and evolving? Each pattern may be looked upon as a hypothesis like one of the hypotheses of science. All free to evolve under the impact of new experience and observations.

Catalog and Organization of Design Patterns

		Purpose		
		Creational	Structural	Behavioral
Scope	Class	Factory Method	Adapter (class)	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter (Object) Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Mobile Internet Technical Architecture (Nokia MITA)

All Mobile application design is based on complex technical architecture and every developers should keep basic device architecture in mind before designing any apps.



Problems of Mobile Application Development

Architecture of every device so complex. It becomes very difficult not impossible to learn and understand it. Many basic things should keep in mind such as device components, their qualities and etc. Technologies that are currently in use are very fast developing and nature is Heterogeneous. Everyday new API's are coming and developers have to learn those news things to bring something new into application. Mobile apps development field is relatively very new. It is not much experienced like web development. In current Market every one wants apps to be developed in short period of time. Every mobile apps development is now project based and most often it would not well planned. Developers bring solutions to apps problems very fast and on the requirement basic. There are still problems in information exchange between users and developers. There are lots of companies are involved in mobile apps field like service provider, content provider, network operators, technology provider etc. Without this companies supports it is impossible to make fully functional application. Every companies have interoperability of each other.

Patterns Supporting Mobile Application Development

If developers know Design patterns of device they can handle complexity very well. Patterns provide developers and detailed Analysis and Description of the Device for which they are developing application. Developers can be more systematic to design process of application. Learning Patterns provide developers insight into the device and its architecture and can get to know how to exchange information between device and application.

Architecture

Architecture: Design, the way components fit together. It may also be used of any complex system, e.g. "software architecture", "network architecture". Architecture is a term that lots of people try to define, with little agreement. There are two common elements: One is the highest-level breakdown of a system into it's parts; the other, decisions that are hard to change. There isn't just one way to state a system's architecture; rather, there are multiple architectures in a system, and the view of what is architecturally significant is one that can change over a system's lifetime. Architecture is a subjective subjective thing, a shared understanding of a system's design by the expert developers on a project. Commonly this shared understanding is in the form of the major components of the system and how they interact. It is also about decisions, in that it's the decision that developers wish they could get right early on because they're perceived as hard to change. The subjectivity comes in here as well because, if you find that something is easier to change than you once thought, then it's no longer architectural. In the end architectural boils down to the important stuff.

Patterns of Enterprise Application Architecture

Enterprise applications are about the display, manipulation, and storage of large amounts of often complex data and the support or automation of business processes with the data. Enterprise applications have their own particular challenges and solutions, and they differ from embedded systems, control systems, telecoms, or desktop productivity software. In case you haven't realized, building computer system is hard. As the complexity of the system gets greater, the task of building the software gets exponentially harder. As in any profession, we can progress only by learning, both from our mistakes and from our successes. Presented patterns represent some of the learning written in form which learn these lessons quicker.