Summary of the Scientific Text

At the begging of the book, just before the first chapter, authors state, that this book is not an introduction to object-oriented technology and design as many people think but instead it is a book one should read when one is proficient enough in atleast one object-oriented language. It is also stated that this book is not an „advanced technical treatise“ but rather a book of design patters which are described with simple and elegant solutions to specific problems.

Authors want one to imagine the work of a future software archeologist, tracing the history of computing. The fossil record will likely show clear strata: „here is a layer formed of assembly language artifacts, there is a layer populated with the skeletons of high order programming languages“. The described layer will be intersected with the imprint of other factors that have shaped the software landscape: components, residue from the great operating system and browser wars, methods, processes, tools. Each line in this strata marks a definitive event: below that line, computing was this way; above that line, the art of computing had changed. This book draws a clear line of demarcation and represents a change in practice of computing. The impact of this book cannot be overstated: this book is a collection of projects from modern patterns movement, which are yet to be documented.

It is indeed a very hard process to design object-oriented software and it is even more exhausting job to design reusable object-oriented software. Many novices get lost in the jungle of many different collections, methods and types and in most cases fail to design a viable reusable software. On the other hand experienced programmers make good designs mostly thanks to one‘s long experience in the field and knowledge, novices just do not have. Good designs are also result of reusing the same principles solving similar problems and just improving the current design and polishing it. It is known that design experience is very valuable as one get‘s that „deja vu“ feeling when writing code and solving problems similar to ones solved in the past. However, we don't do a good job of recording experience in software design for others to use. This brings us out book‘s purpose – to fill it with records of object-oriented software experience as design patterns. Each entry is firstly systemised by name, problem, solution, and many other criterion.

Design pattern simply describes a problem which occurs commonly in used environment and the describes the core of the solution to that same problem in such a way that you can use that same solution a dozen times over and over again without ever doing the same thing twice. In general, a pattern has four essential elements. First one is the pattern name is a handle we can use to describe a design problem, its solutions, and consequences in a word or two. Naming a pattern immediately increases our design vocabulary. It lets us design at a higher level of abstraction. It makes it easier to think about designs and to communicate them and their trade-offs to others. Finding good names has been one of the hardest parts of developing our catalog. Second one is the problem. It describes when to apply the pattern. It explains the problem and its context. Sometimes the problem will include a list of conditions that must be met before it makes sense to apply the pattern. Third one is the solution which describes the elements that make up the design, their relationships, responsibilities, and collaborations. The solution doesn't describe a particular concrete design or implementation, because a pattern is like a template that can be applied in many different situations. Instead, the pattern provides an abstract description of a design problem and how a general arrangement of elements (classes and objects in our case) solves it. The last but not least is the consequences are the results and trade-offs of applying the pattern.The consequences for software often concern space and time trade-offs. Since reuse is often a factor in object-oriented design, the consequences of a pattern include its impact on a system's flexibility, extensibility, or portability.

For this book authors have concentrated on patterns at a certain level of abstraction. Design patterns are descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context. A design pattern names, abstracts, and identifies the key aspects of a common design structure that make it useful for creating a reusable object-oriented design. The design pattern identifies the participating classes and instances, their roles and collaborations, and the distribution of responsibilities. Each design pattern focuses on a particular object-oriented design problem or issue. It describes when it applies, whether it can be applied in view of other design constraints, and the consequences and trade-offs of its use.

As the book contains several dozen of different design patterns it would be a fatiguing work to summarise them all and not to mention it would take several pages of information about every and each one of the patterns. That is why one pattern was selected as an example of pattern structure. The example is „Builder“ design patter. It is used to separate the construction of complex object from its representation so that the same construction process can create different representations. Mainly, a solution is tto configure a Reader class with value which converts object data to another textual representation. Subclasses are used in order to initialize the main interface class. Each subclass takes that same mechanism for converting and assmebling new objects. This design pattern is mostly used when the algorith for creating a complex object should be independent of the parts thaaat make up the object and how they are assembled. There are three main consequences to using this particular design. First one is that it lets you vary a product‘s internal representation. Second one is that it isolates code for construction and representation. And finally it gives you finer control over the construction process. „Builder“ is commonly used in Smalltalk-80.

In summary, this book is really helpful and interesting to ones seeking new ideas and looking to increasing their own problem solving skills. This book provides complex but rather easily understandable design patterns leads one through all the steps required to implement selected pattern and even gives real world examples to make a better picture.