Preface

P1. Content at a glance

System design and implementation are demanding occupations, in all respects. Besides professional credentials, having a passion is almost a mandatory requirement for one to work in this business. Physical and intellectual endurance, as well as the ability to peacefully work in stressful environments, which inherently involve lots of uncertainty and stone-engraved deadlines are also a must. Why is it so? The reasons are numerous. However, one of the major causes is this. Although one can do system design and implementation slowly, moving fast in the system development business is a great advantage. The shortened time-span allows keeping the whole process and vision of the system "in-memory", thus making all of the project's structure and components' interrelationships exposed and transparent. If you try to interrupt the design and development process for a month, it is unlikely that something good will happen. Of course, the above by no means allows one to neglect the project documentation – a fast moving train has to work by itself, but the railway should be in good condition too.

Whenever the author hears the phrase "two year project", it makes him feel uncomfortable. In his view, for such projects, it is worth to look for efficient and elegant solutions (which always exist, for reasons this book uncovers). Perhaps one will need to allocate more resources, but the result will be that the project can be done in several months. The author never encountered a situation where this was impossible.

Of course, is it *people* who do things. In order to keep up a fast pace, the project needs appropriate people capable of doing so; certain conditions must also be present, these are not always readily available, but can be created. This book considers such "environmental" aspects in detail. So, one of the threads of this book is how to organize and optimize available resources in order to implement a high-quality software system in the shortest possible time.

When we do some job, we use tools. In software development, we have lots of tools and many more enter the scene every month. How to approach choosing the right tools for a particular project is another topic, which we study in this book, comprehensively and from different perspectives.

Once we outlined our preferred conceptual design and development framework, thus acquiring a set of general criteria for proper design and implementation practices, we begin the analysis of available system design approaches, comparing their pros and cons.

The author's most cherished area of system design is development of distributed high performance software systems, which by definition involve intensive and extensive interprocess communication. So, the last chapter is entirely about interprocess communication mechanisms and appropriate system designs. We will present results of original research done by the author, concerning system design on the basis of fast interprocess communication mechanism founded on synchronized access to memory mapped files (shared memory). These mechanisms are widely used in the industry. However, this concept has more potential, which the last chapter uncovers. Previously, the author pioneered the practical application of this concept and implemented it as a set of libraries called "Joiner" libraries, so that this material, to some extent, is a continuation of his previous research. However, this chapter should not be viewed as standalone material, unrelated to the rest of the book. On the contrary, this material is a demonstration of the application of conceptual considerations, discussed in the previous chapters, to solutions of important practical problems.

Beside these major threads, the reader will find lots of other relevant material, including examples of particular system designs, discussions of merits and drawbacks of certain tools and approaches, real examples, relevant coding excerpts and many other things, which in one way or another affect the quality of system design and implementation. The

book includes lots of illustrations and diagrams, which facilitate comprehension of the material.

This way, we will cover the whole spectrum of knowledge required for successful system development. We will not only teach the sailors general navigational principles in open seas, while sitting in the dreamy comfort of classrooms, but also take them on a real sea voyage through the rough waters of challenging problems, exposing hidden currents and crushing cliffs of lonely uninhabited islands. You will see that it is possible to successfully cross the most challenging seas in record time, even if you travel on a relatively small boat. Of course, the boat's quality is important too. However, it is the navigational skills, courage and the overall vision that make the difference, and not the boat.

P2. System design "philosophy"

If somebody would ask the author to concisely describe a system design "philosophy" which he thinks is a proper one, and which he actually acquired by completing numerous projects, then it would sound as follows:

- Always look for efficient, generic, elegant and optimal solutions, at all levels and in all phases of project development; wrap things, that is design, code, functionality, interfaces, etc; collapse everything into smaller, simpler, generic structures – it is always possible.
- Do not rely on "off-the-shelf" universal solutions, they are never optimal. By definition, they include lots of overhead and unneeded things, otherwise they could not be universal.
- Understand your toolbox, but use only the necessary tools. Use these tools to craft your own solution, unique and tailored exactly to your problem. Do not allow any of your tools to define the design process, this job is best left to a human being.
- In general, stay away from extremes and boundaries. Remember: all in measure, including measure.
- Stockpile resources! This means all sorts of resources, such as robustness and scalability of system design, generic solutions at all levels, which will be able to accommodate more requirements, code implementation that anticipates more changes, and above all time, time, time! In the initial system design, the required system scalability reserve should be at least five-six times of the initial intentions.

In this book, we will introduce conceptual level considerations related to system design and software development, in order to create a *conceptual* software development framework. In fact, any discipline is based on some conceptual foundations. Deep and clear understanding of these fundamentals significantly facilitates comprehension of a subject. This is not unlike having an overall map of an area we are traveling in.

On the other hand, each professional, such as a scientist or a software system designer, builds and uses his own version of such a conceptual framework. Sometimes, conceptual deviations from the average perception of some discipline's foundations can be really large – this depends on the discipline and the person. In this regard, the area of software systems' development presents limitless opportunities for creating a great variety of custom conceptual frameworks. The reason is the existence of many opinions, concepts, and development systems, as well as the enormous number of degrees of freedom which the mere nature of software design and implementation provides.

It is true that any system, such as a mechanical one, can be designed and implemented in different ways. However, in software development, the number of possible implementations is larger by orders of magnitude, compared to the majority of other systems. For instance, you can think of designing mechanical devices as navigating river systems, while software development is more like traveling in a sea with zillions of islands.

We will explore in detail some areas of particular interest, which, in our view, are important components of successful system design. We will not explore the whole area, but

we hope that the reader will get enough insights, ideas and general concepts in order to continue this expedition on his own. In this regard, let us make a side note. Some authors present their subjects and disciplines in an unnecessarily overcomplicated way. This might happen partly because of inability of some writers to think clearly and rationally, and partly for the purpose of presenting their writings in a way that makes them seem more meaningful than they really are. Such authors apparently forget the Aristotelian "golden mean", which, in our case, means that the *form* should be commensurate with *content*; the form should *match* the content, in order to make them unified. This is the only way to make the *whole* thing appealing and meaningful.

The best knowledge is acquired by learning conceptual paradigms and solving concrete practical problems in such a way that the general conceptual view is not lost but reinforced. So, within our framework, we will consider practical things and their interrelations with conceptual foundations.

Real understanding and professionalism come from a clear perception of factors which are meaningful and critical for the problem, considered in their *dynamic interrelationships*. Achieving this level is *impossible* without conceptual thinking and knowledge. We have seen books written in an instructional manner, confined to particulars of the subject. Reading such books is like traveling in a deep trench. The walker will see something, but not much and not far away. He will not see the whole landscape; doing so is an essential requirement for any journey to succeed. Besides, even if one somehow gets to the final destination, this trip is going to be boring.

So, as a result of the thinking paradigm we described, this book differs somewhat from many other software design and development books, whose only goal is to teach the reader how to do some very particular things. We would rather put in more effort to give the reader a more broad description of system design methods and approaches, emphasizing the multidimensional and multifactor nature of the whole discipline. The reader will be able to observe both the panoramic and stereoscopic picture, although in some instances without lots of details (there not enough room in this book to provide high resolution imagery and, in fact, there is no need to do that for our purposes). However, the level of detail will be sufficient to get a feeling for the conceptual framework of the system design business in its entirety.