Let us consider methodologies. They are a parallel dimension to the lifecycle models; the methodologies assist in software product development of mission-critical systems. Usually, a methodology means a set of techniques, models, methods and tools. Here, it will also mean a set of best practices for software development. Within a software development methodology, we can seldom find rigorous mathematical models other than these used for economic evaluation and feasibility study of the project. A number of approaches, especially in case of agile methodologies, such as Scrum or Agile, suggests a flexible set of customizable best practices, i.e. practical methods for software systems development. Therefore, it is often meaningless to consider a methodology as a purely theoretical research subject. In this respect, many of the above considerations on the lifecycle models for software systems development will be applicable to the methodologies, in case we include the best practices and certain classes of tools into the scope of our discussion.

In our view, a software development methodology is a parallel direction in relation to a lifecycle model. We have already mentioned the differences in lifecycle for software products and software projects; our focus is the products. A methodology is useful as a process framework and a set of practices for efficient software product (and project) development, including mission-critical applications and crisis conditions. Therewith, the methodologies that we are going to discuss can support various lifecycle models. For example, the Rational Unified Process (RUP) methodology can use either waterfall or spiral lifecycle as a basis. The other large-scale methodology that we are going to discuss, the Microsoft Solution Framework (MSF), also supports a number of lifecycle models. In terms of software systems development and the lifecycle stages, a methodology is a less formal approach than a model. It is often possible to scale a methodology up or down, since it is a framework, which depends on the size and scope of the software product. For example, Rational Unified Process initially intended for large-scale software development can scale, i.e. it can use more or less detailed development plan, processes and deliverables. Similarly, for the Microsoft Solution Framework there are more flexible implementations (often called MSF Agile) and more detailed ones (often called MSF Formal). An adequate size-to-deliverable ratio for methodologies and size-to-phase ratio for models is the key to crisis management of software product development.

According to the product size and scope, there are certain methodologies that are initially designed to build large-scale and mission-critical systems. We can call them large-scale, heavy, or formal. These are somewhat similar to the full-scale lifecycle models, which embrace the entire lifecycle and produce elaborate product documentation. However, each of these large-scale methodologies being a framework of principles, best practices, processes and deliverables, allows for downsized implementation of software systems development. Additionally, there is a number of more flexible methodologies, which are suboptimal for large-scale and mission-critical software products, and which are designed to accommodate crisis conditions, such as high risks, requirements fluctuation, and high uncertainty. The large-scale methodologies as process frameworks can support all the stages of the lifecycle models, such as the above-mentioned waterfall or any of the iterative patterns. The large-scale methodologies are RUP and MSF. The RUP is the standard of IBM, and MSF is the Microsoft standard. Interestingly, the MSF methodology originated from the synchronize and stabilize lifecycle model [16]. The MSF methodology is complex and agile at the same time; it supports multiple and highly scalable software development teams.

Potential benefits of MSF for crisis management result from a certain level of equality of the project team members, and clearly distinct personal responsibilities at the same time. Such team organization allows for scalability and crisis agility even in case of team of teams. In crisis, certain project team roles may overlap, according to the trade-off matrix of roles. For example, project manager and product manager are typically different people; however, it is possible to combine their responsibilities for a small project or a local crisis. Likewise, some other role combinations are possible.

Common vision is another potentially promising concept of MSF, especially in crisis. Vision is an original idea, clear yet informal, of the fundamental differences and customer values for the future software product as compared to the existing ones, and its benefits after the implementation. A distorted vision can easily cause chaotic development and local crisis; however, sharing common vision through open communication is often a remedy for the crisis. The progress of software development is sequential elaboration of the software product, and, of course, the vision is the most abstract representation of it. Before the product development based on this representation actually starts, the developer side needs a high-level, and, later on, a more detailed document that describes the product specifications. A better common vision for the project team promotes more accurate product specifications. As soon as the project team is formed, the project schedule is developed. This includes roles and their activities for each product development stage. The project plan also includes the key activities duration, the primary control points, i.e. milestones, and the results at each stage of the product development, i.e. deliverables. After the planning phase, the development phase starts. In crisis, it is very important to detect the project activities and the product deliverables, which lie on the critical path in terms of resources including labor, timeframe and quality. It is equally important to ensure that the shared project deliverables will provide operational quality within the milestones set, at least for the mission-critical functions.
Often, a project team is recruited for the only project. Thereafter, it often happens that the product exists independently from the development team, as the maintenance team is different. That is why product documentation quality is critically important as it ensures continuous operation of the entire product line, such as Microsoft Windows, Office applications, and so on.

MSF as a methodology embraces not only lifecycle of a software system, but also the methods and techniques of software product and project development, including processes and roles in the project, responsibilities and deliverables, communication and teamwork, and project documentation.

Concerning agile methodologies, we are going to discuss Scrum, Agile, and eXtreme Programming (XP). These are sets of best practices, i.e. recommendations for crisis management of software product development under high uncertainties and risks. Given the budget and timeframe, the aim of the agile methodologies is to develop the product of a certain quality level, or to cancel development if this is impossible. As in case of the lifecycle models, we will also describe the advantages and disadvantages of the software development methodologies.

Overall, the methodologies are practically oriented approaches focused on cost optimization. However, optimization in this context does not use a rigorous mathematical model, although the methodologies use certain product metrics to monitor, evaluate and plan the product development. Since there is no clear way to develop a mathematical model of software development with these methodologies, it is not entirely correct to say that they result in the optimal solution. However, with the help of processes and metrics the developers can reach sufficiently good and justified project decisions, which is still suboptimal in a mathematical sense.