# Task Analysis

**How this Task Analysis was made**

In this document we list various skills that are often used in software development, data processing and other IT engineering solutions. The task analysis is focused on the skills and objectives that are necessary to survive and to thrive in a medium-to-large IT organization. This course can also be described by a single phrase - “An introduction to your future specialty”.  
We list the tasks that we expect the learners to perform after the completion of this course. Four top-level tasks are divided into several sub-tasks each. We try to state the sub-tasks in a specific format: “**Given AAA, perform action BBB**.” We do not provide any measurable objectives at this point.

## Part 1: Collaborating

**Summary:** We expect that the greatest challenge during this course is interaction in teams and orderly contacts with customer representatives and RBS faculty. Doing work in a team may seem unnatural at first. We expect that many course participants have limited experience with teams.

1. **Choose and follow a collaboration approach that suits your team.**
   1. **Collaboration tools and practices.** Given the guidelines, prepare a collaboration framework for your team, explain it during a class session and follow it in all your subsequent activities related to the course.  
      *Instructors will provide the participants with a list of requirements for collaboration; it cannot be trivial - for example, “our team will always meet after the classes and we do not need to accommodate any off-site work”. It has to be fair to all team members and sufficiently transparent so that instructors can inspect your key decisions and statuses. Project-related communication outside the selected framework is also encouraged, but you will not be evaluated on what is not visible.*
   2. **Proper delegation and division of labor and responsibilities.** Given the overall objectives and the project management methods, ensure that every stage in your project plan is done collaboratively and the team has designated contact persons and presenters for each stage.  
      *All the class sessions involve auditions. Each team has to present their current status clearly and succinctly. The statuses of the team members should be communicated and visible at other times as well - it is not done just for the instructors’ sake.*
   3. **External communication.** Given the goals of the assignment, ensure that all the requirements and expectations are discussed with the customer and, if necessary, other project stakeholders.  
      *External communication is evaluated according to different criteria (compared to the communication inside the team). As before, it has to be visible to other team members and to the instructors in order to be evaluated. Communication can be evaluated either directly (what types of messages or artefects were exchanged) or indirectly - project work that has a chance to miss the customer’s needs means that the communication was not done properly.*

## Part 2: Managing Project Iterations

**Summary:** In this course we expect project management that (loosely) follows Agile or iteration-based development cycles. We assume that iterations roughly correspond to the class schedule - usually it is 2 weeks per iteration. (Very short iterations create unreasonable overhead of paperwork and reporting; longer than 2 weeks do not give an opportunity to learn from earlier ones.)

1. **Create an iterative project schedule and follow it during the semester.**
   1. **Value proposition.** Given the description of the customer’s problem, create a value proposition.  
      *(There may be various ways how to fix customer’s problem - not all of them involve software engineering or similar project activity. Many problems in people effectiveness and education are solved by changing the motivation, attracting the best people, and so on. The justification of the project and its viability is typically done at the very beginning of the project - it may be called “value proposition” or “business case”. Participants may need to come up with several approaches to the initially stated problem; then they can eliminate approaches that could work “in principle”, but do not lead to a workable value proposition - they may be too expensive, not a good fit for the customer’s context, rely on non-existent or untested technologies, etc.)*
   2. **Iteration planning.** Given a set of initial requirements from the customer and “value proposition”, produce a simple iterative plan that fits with the Agile methodology. *(Iteration planning typically happens* ***after*** *the main idea to solve the customer’s problem is already settled and it is proven that the proposed solution might work.)*
   3. **Risk planning.** Given the real (or artificially caused) risk situations, respond to them by adapting your current iteration and also by rescheduling your work.  
      *(Project management involves reaction to changing requirements and team responses to various risks. In more complex projects those risks appear naturally - team members may have health problems, some external dependencies might work differently as initially expected, customer does not provide the necessary information. In simpler projects we might need to cause “artificial” risk situations.)*

## Part 3: Creating value

1. **Create a solution that effectively addresses the problem stated by the customer and other stakeholders.** 
   1. **Innovative solution.** Given the problem description, come up with a workable idea that leads to substantial improvements compared to traditional approaches that have been used so far.   
      *Each software solution is a unique solution to a unique challenge. It is counterproductive to produce new software solutions, if they do not lead to innovative approaches to existing issues. Innovative idea is the starting point for all subsequent technical creativity.*
   2. **Subject-matter knowledge.** Given the customer’s input, reflect the specific expertise in your solution.   
      *For example, if your solution aims at solving some existing problem in education, it should use the best available knowledge and data about the topic. Since project developers usually are not experts in the area, they need to elicit all the relevant information and organize it in a new and more efficient way.*
   3. **Functionality of the solution.** Given the priorities of the functionality for your solution, ensure that the most important ones are implemented.   
      *Student projects only rarely reach all the major objectives. Successful value creation means prioritizing the work, ensuring that Minimum Viable Product (MVP) is still implemented and delivered.*
   4. **Knowledge transfer.** Given the intermediate or finished solution, the team has a well-defined and tested approach how to introduce it in the customer’s infrastructure.  
      *This is not necessarily a training, but the project team should have an approach how the solution will actually work in the production environment.*

## Part 4: Following standards

**Summary:** In some projects form is as important as substance. We try to enforce and evaluate course participants on their ability to create documentation and code that follows some best practices and are taught in some other RBS courses. Course participants need to be flexible and considerate of the needs of others. As known since early days of the TCP protocol: Be conservative in what you send, be liberal in what you accept.

1. **Create all the deliverables and the final product that conforms with the expectations of your customers and instructors.** 
   1. **English writing.** Given the style guidelines, ensure that the standards of the technical writing are followed.  
      *Deliverables in the project might be written in English even if the customer (municipality) and other stakeholders (high school students?) actually use Latvian. In this case we would try to enforce best practices of English writing.*
   2. **Avoiding fragile and untestable code in core functionality.** Given the best practices of test coverage for the computer code, implement automated tests and produce compliant code and data for your solution.  
      *Students who just start to code may be tempted to create functionality that is highly unstable and untested. It may break unexpectedly due to small changes. Up to a certain point this is unavoidable - we do not want to become “control freaks” who set impossibly high standards. Every software solution should have bugs. On the other hand, each product should have certain “core functionality” that is stable enough - and that can be tested at all times.*
   3. **Configuration management:** Given the environment where your solution should run, select, adapt and consistently use configuration management approach.   
      *Configuration management ensures that all team participants know, how the development and production environments can be installed and used. They check into the code repository only those portions of their solution that can be back-referenced to the functionality, they know how to run tests, how to make changes, how to ensure that only good versions of the code run in the production environment.*
   4. **Maintainability.** Given the context where the solution will be deployed, ensure its sustainability and ability to accommodate known data flows and changing customer needs.   
      *Solutions that end up to be brilliant but difficult to maintain can have unacceptable TCO. If this may happen, it is necessary to point this out to the participants.*