NATIONAL RESEARCH UNIVERSITY

FACULTY OF BUSINESS INFORMATICS

www.hse.ru

http://bi.hse.ru

# Case-study method in the master program “Software engineering methodology”

Pesockaya E.Y.

# Contents

## In Order of Appearance:

## By stages of software development life cycle:

## Topic of "Software Engineering Meth-odology":

# Introduction

“I am happy to announce that we have a new client — one of the leading banks has decided recently to change its core banking system and invited us to participate in **Automated Teller Machine** (ATM) software development” — Alexander announced to his colleagues. Alexander was a managing director in a software development company **SkillSoft**, and now the crewsize of his organization has grown up to 30 persons.

Alexander had been negotiating scope, budget and schedule with the leading Russian bank for more then 3 months before they came to the agreement.

13 years ago Alexander graduated from the University and started his career as a programer in a small Russian company. He was programming on Pascal and Basic, later on Visual Basic and then proceeded with C++. By now Alexander considered himself as a mature software developer, who lived out many troubles and success stories with customers and believed he had great expertise and skills in software development area.

Today morning he gathered software development company managers to announce the new project of developing software for the banking — ATM functionality

“Alexander, may I ask you a question” — continued Ivan, the Development Lead, — “If the Bank decided to use a new banking system, why should not they use ATM module, which should be a part of this new core banking system?”

“That’s a very good question” — answered Alexander. He explained to his colleagues, that the conventional ATM module is not suitable for the Russian bank because of some transactions specifics and Russian legislation. “It is possible to use this module, but the efforts for customization should be huge! This was the subject of my 3-month negotiation with the Bank and we concluded that it would be more reasonable to create ATM software from scratch and ensure further integration” — finished Alexander.

“And what is about scope and deadlines? What architecture and functionality are intended?” — asked Elena, System Architect and Testing Lead. Her colleagues actively nodded.

“We have 3-month period for software requirements gathering, planning, designing, coding testing and deployment!” — commented Alexander. “Regarding architecture and functionality you should organize a set of interviews with client team and analyze the requirements and constraints. We have very strict deadlines and limited budget that could not be enlarged.

Our further step is to organize working groups with the client and start to work on scope detailed definition and identification of requirements. We need to finalize requirements document in 2 weeks, so that in 3 weeks time we have received the client’s approval for requirements and started designing, then coding, testing and deployment!”

“Alexander, how to organize working groups? How do we know whom to interact?” – asked Irina, system Architect.

“Irina, this will be your responsibility to identify all interested and involved parties by tomorrow so that we can start to plan meetings immediately!” — said Alexander.

“Ivan should be appointed as a Project Manager of this ATM project. I am expecting the draft of a high-level project management plan with main life cycle phase’s and schedule from Ivan.

See you all the same time tomorrow, guys! Don’t forget I am waiting for the first results from you. Have a good day!” — Alexander finished the team meeting.

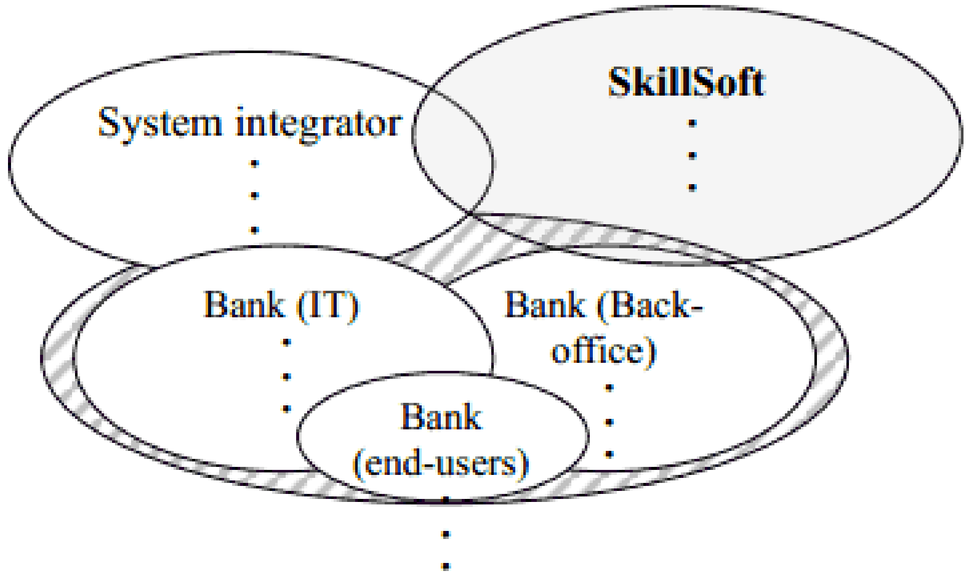
## Case 1. IT Project Stakeholders

After the team meeting finished Irina was dreaming of a coffee-cup to make her mind work. Irina was not as experienced as her colleagues, but was highly motivated and interested in a new ATM software development project.

She was nervous and upset due to a given task to identify all the interested and involved parties by tomorrow, but at the same time she felt proud of the responsibility! She had never been allowed to take self-consistent decisions — all her previous work was to effect given tasks, clearly defined to her.

"What shall I start with?" — Irina was thinking. There are at least three distinct groups that should be considered: end users, bank back-office staff responsible for running the software once it is in production, and the IT support staff who is responsible for aiding users with the software once it is in production. We should also interact with the system integrator, who is responsible for core banking system Implementation — as our ATM software should be fully integrated with the new system. "Who else?" — she sipped her coffee.

Irina decided to write down the illustration of her thoughts.



Irina thought that she also needed to identify the level of control that each group had over actual development and deployment. Can any group stop ATM deployment if we don't meet their specific requirements?

### Task:

**1.** Identify all interested and involved parties for the project. What are their interests and role in the Project?

**2.** Define working groups with involvement of SkillSoft specialists: project manger, business analyst, developers, system architects, tech writers, testers?

**3.** How many channels of communication you should use for each working group? Calculate channels of communication (use the formula: **$C = n \cdot (n - 1) \div 2$**, where **n** represents the number of people on the project team) and outline them.

# Case 2. Development Plan

Ivan has been working for the SkillSoft company for more then 5 years. He was the most mature and experienced software engineer (after Alexander, of course). And he was not surprised when Alexander appointed him as a project manager of ATM software development project.

This role was not new for Ivan. A week ago he finalized the previous huge project of developing software for unmanned petroleum station that could provide service with no operator, with the use of customer's credit card only.  They successfully finished all deployment tests and the system was considered as corresponding to the requirements and trustful. The pilot unattended petroleum stations already operate in the far Moscow region.

Ivan listed for himself the core software's life cycle activities:

• Planning / requirement analysis

• Design and implementation

• Testing, documenting

• Deployment and implementation maintenance.

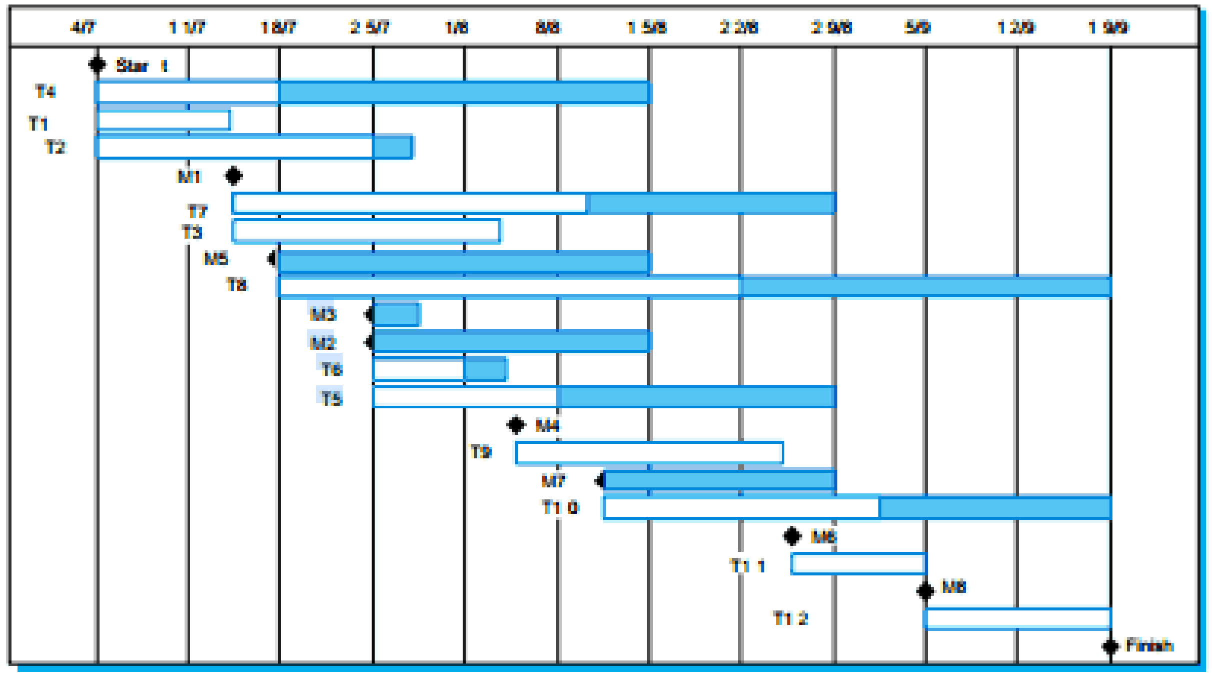
The question is whether should he add to the development plan the "software maintenance phase or not? He knew dick from his experience that software maintenance might be 2-3 times as difficult as making software design, development and deployment.

Then he came back to the team meeting where Alexander announced the deadlines: 3 months for all project activities. Ivan found the answer to his previous question on software maintenance and started to make a plan.

## Task:

**1.** Draw a high-level schedule for software development activities.

**2.** Use Gantt Chart for graphical representation to show high-level tasks and activities, timeline and responsible.



# Case 3. Case Tools Selection

Being a system architect, Elena was also responsible for technical support of her colleagues. This was a kind of hobby for her. She was always very curious about all new gadgets and specialized software, was fond of reading modern software journals and articles and has always kept her pace with new technologies and tools.

When she was working for another company, being engaged in previous projects she always felt lack of computer-aided software engineering (CASE) tools to support software development and evolution processes. She was aware of many useful tools, but could not use them.

When Elena came to SkillSoft, she was happy to learn that Alexander, the managing director, fully approves the idea of using computer-aided software in development process and she agreed to take ownership of choosing and purchasing the required CASE tools for the development purposes.

She believed that Case technology has led to significant improvements in the software process. Of course, software engineering requires creative thought — this is not readily automated. And software engineering is a team activity and, for large projects, much time is spent in team interactions. CASE technology does not really support these.

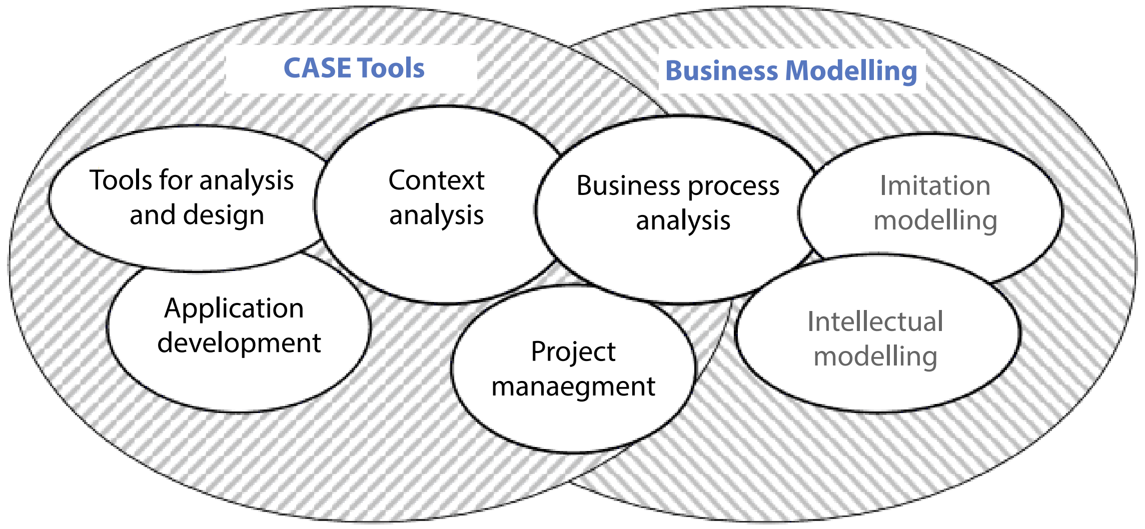
But still many things can be automated, e.g.

- Graphical editors for system model development;
- Data dictionary to manage design entities;
- Graphical UI builder for user interface construction;
- Debuggers to support program fault finding;
- Automated translators to generate new versions of a program, etc.

## Task:

Define set of CASE tools for various process and purposes. Use example (fill-in the appropriate tools). Feel free to add new process and tools to the given example.

Example:



## Case 4. Requirements Defenition

The ATM software development project has been running for one week.

Ivan, the Project Manager, accorded the development plan with the client, they agreed upon the core milestones and deadlines for each development stage.

Most of all interviews to define functional requirements were conducted.  But there were still some gaps that should be considered. For the interviews with the client team Ivan prepared questionnaires from the previous project and assigned two business analysts to conduct interviews.  They gathered a lot of information. But the problem was that the most of requirements were formulated using natural language.

To create a requirement document, business analysts should define:

1. Functional requirements
Statements of services the system should provide, how the system should react to particular inputs and how the system should respond in particular situations.

2. Non-functional requirements (product, organizational, external)
Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.

3. User requirements
Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.

4.System requirements
Detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so that may become a part to a contract between client and contractor.

5. Interface specification
Requirements that come from the application interface and reflect characteristics of that system.

### Task:

**1.** Based on the informal specification suggest functional, non-functional, user, system and interface requirements (2-3 for each category). Use common sense for writing requirements:

**A software system for an Automated Teller Machine needs to provide services on various accounts. ATM can service only one customer at a moment. Customer able to insert an ATM card  and enter personal identification number. If PIN is correct customer may choose a service. The customer services include operations on current account, operations on savings account, transferring money between accounts, managing foreign currency account, and change password. The operations on a current or savings account include deposit, withdraw, show balance, transfer money between any two accounts linked to the card and print out transaction records. The operation should be supported with a sound signal and should be finished with in a reasonable time (suggest appropriate time-limit for the operation).**

**2.** Write a) safety, b) functional reliability, c) security requirement specification for the ATM system.

# Case 5. Software Development: Object Oriented Approach

At the next team meeting Alexander decided to discuss the approach for software development.

As a model he chose an object-oriented approach with developing an object-oriented system model to implement requirements that were defined during last three weeks. Within this approach his development team should have developed an object model of the application domain and execute the coding using an OO programming language such as Java or C++. His team fully supported his idea as it ensured easier maintenance where objects may be understood as stand-alone entities. What they liked about object-oriented approach was objects as potentially reusable components.

We should use objects (the  entities) in a software system which represent real-world instances  and system entities as well as object classes, the templates for objects that may inherit attributes and services from other object classes. It was agreed to use UML as a core modelling language for class description.

In UML Objects and object classes participate in relationships with other objects and object classes. Generalised relationship in the UML is indicated by an association. Associations may be annotated with information that describes the association. Associations are general but may indicate that an attribute of an object is an associated object or that a method relies on an associated object.

Now the software developers should proceed with object-oriented design process and become an essential communication mechanism with the client team.

## Task:

**1.** Define ATM object classes / subclasses (e.g. ATM, Bank account, Transaction / Balance_inquiry, Withdrawal, Deposit, Money_transfer).

**2.** What factors to be taken into account in the design of a menu-based interface for walk-up systems such as bank ATMs? Consider the following factors:

a. System users may be disabled so will not be able to respond quickly to requests.

b. Users may not be able to speak the native language of the country where the machine is installed.

c. System users may be completely unfamiliar with technology and may make almost any kind of error in using the machine. The interface must minimise the number of possible errors and must be resilient to any possible error.

d. Some system users are likely to be confused by many options.

e. Different people may understand the meaning of icons in different ways.

f. If the system has navigation options, users are almost certain to become lost.

g. Most users will want to use the system for very simple functions (e.g. withdraw cash from an ATM) and will want to do this as quickly as possible.

h. There are many different ATM interfaces so each must be considered separately.

# Case 6. Defenition of Data Flow And Process Model

Ivan was really happy — he accorded the requirement document and now was able to give a task to system architects to proceed with further steps: definition of data flow for ATM and creation of sequence of actions to make sure he and his team understood the problem correctly.

He gathered his colleagues to discuss next steps and the results of the previous stage reflecting their and the client's compliance upon the main services and operations:
The ATM services include:

- operations on current account;
- operations on savings account;
- transferring money between accounts;
- managing foreign currency account;
- password change.

The customer is able to conduct the following operations with his account:

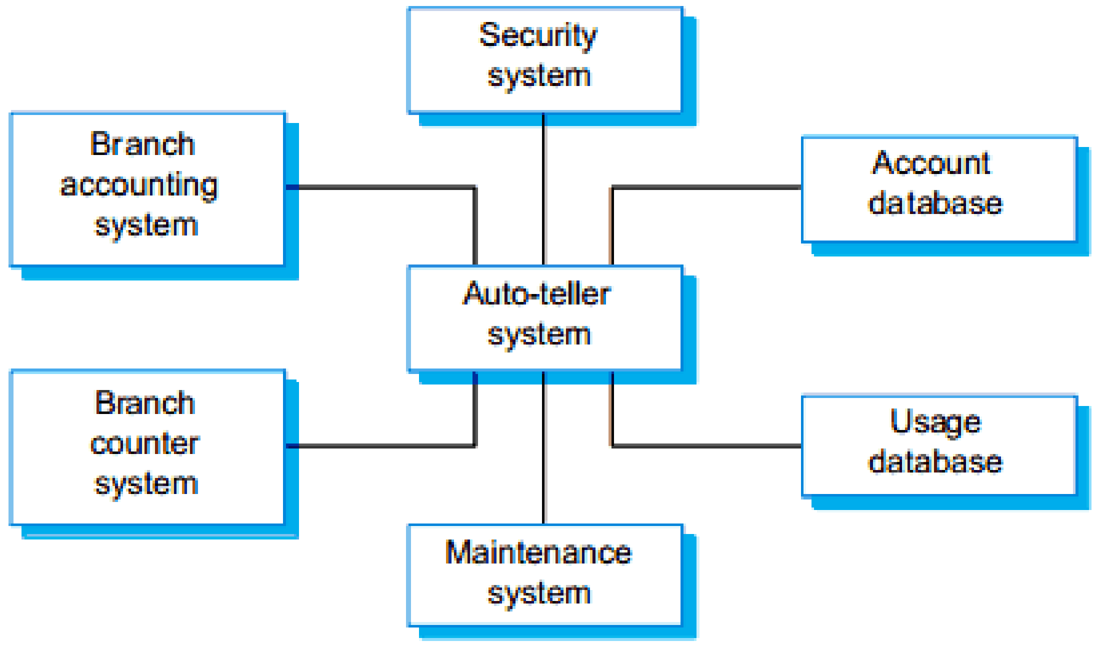
- deposit,
- withdraw,
- show balance,
- print out transaction records.

In order to reflect the graphical representation of data flow within an information system  he gave a task to his system architects to draw a data-flow diagram which shows processes and activities for modeling the data processing with an ATM bank involved when a customer withdraws cash from the machine (the main ATM operation).

## Task:

**1.** Create Data-flow diagram

**2.** Based on the data flow of operations  create a model of the data processing that might take place in ATM. Consider available context model:



## Case 7. Sequence of Actions Defenition

Ivan invited another system architect and asked to write a use-case for a withdrawal from an ATM and use Sequence diagram to show the sequence of event processing in the ATM system to model flow of control and illustrate typical scenarios.

System architect knew that sequence diagrams (or collaboration diagrams) in the UML are used to form interaction between objects. He decided to check the possible level of detalisation he should use for illustrating with Ivan. He also asked to provide him with main operations that he should describe using sequence of actions to show interactions between actors and the system objects that they use. The main operations were:

• System checks customer balance and provides the cash if available;

• System prints receipt and asks customer to remove their ATM card;

• Customer removes ATM card.

### Task:

**1.** Define withdrawal transaction use case:

• Pre-conditions,

• Post-conditions,

• Primary actor,

• Stakeholders (customer, bank, ATM administrator).

**2.** Create Sequence Diagram. Use the recommendation bellow:
Show person, system and database. A box inside the system box is titled, "Validate Card", "Handle Request", "Complete transaction".

Main activities:

• Customer inserts ATM card;

• System displays screen asking for PIN;

• Customer inputs PIN;

• System displays screen with banking options;

• Customer selects "withdraw cash";

• System displays screen requesting amount of withdrawal;

• Customer inputs amount;

• System checks customer balance and provides the cash if available;

• System prints receipt and asks customer to remove their ATM card;

• Customer removes ATM card.

# Case 8. Validation And Testing

Elena perfectly coped with system architecture. Using UML she build very comprehensive system models that was an easy subject to discussion and negotiation with the client team. Based on her design document with system architecture description, her colleagues produced the program code that needed careful testing. For validation and verification purposes Elena needed to check whether software conforms to its specification and at the same time software is processing what the user really requires. Her aim at this stage was to discover as many errors as possible, as suckers "free of defects" system is a myth!

First of all Elena will focuses on:

a) Defect testing to discover system defects. A successful defect test is one which reveals the presence of defects in a system.

b) Validation testing intended to show that the software meets its requirements. A successful test is one that shows that a requirement has been properly implemented.

She has built an expert body for testing purposes. Being in a role of Testing Manager she has taken responsibility for managing and control over software test project, supervising test engineers, defining and specifying a test plan. To the testing team she gave a task to define test cases, write test specifications, and run system tests. **"Please, remember — a successful test is a test that makes the system perform incorrectly and so exposes a defect in the system. Find as many faults as you can!"** — Elena told to Testers. Then she continued:

- **All functions accessed through menus** should be tested;
- **Combinations of functions accessed through the same menu** should be tested;
- **All user input-required functions** must be tested with correct and incorrect input.

For unit and integrations tests Elena engaged development engineers.
For usability and acceptance testing she planned to engage the client's users and volunteers that were frequently attracted to test beta versions in SkillSoft. Regression and stress testing will be conducted automatically with the help of special software (JUnit or another available system which supports automatic execution of tests). The only problem was their pressure for time as she needed to complete all the tests in two weeks!

## Task:

**1.** Suggest 3 user errors that might occur and propose safety requirements for ATM system.

**2.** Write test requirements for ATM to validate given business requirements:

- "ATM must do withdrawals";
- "Withdrawals are between $20 – $300";
- "Withdrawals are in $20 multiples".

## Case 9. Software Deployment And Risk Management

Ivan, SkillSoft Project Manager, spent a lot of days (and sometimes nights) developing a software for banking ATM with his team. For the last two weeks they have been working non-stop testing and debugging the software. Using iterative development approach they have made many iterations before their software product fitted its characteristics and requirements and was able to carry out all banking operations via ATM.

From the very first day Ivan harmonized organization of joint status meetings with Client and System Integrator (responsible for new core banking system implementation), but, since the project work began, they have succeed to organize only 4 meetings where the key IT decisions were discussed. The lack of system integrator involvement made the development process harder — SkillSoft had to read tones of manuals for the new system and came to many conclusions without any help and using their own experience and intuition.

Ivan asked Alexander to try all his best to organize meetings with system integrator in order to reduce integration risks and operating costs by architecting and presenting uniquely matched components. But unfortunately many errors, that could be avoided, were found only at the testing stage and it took many efforts to properly comply the functionality work with the other modules of banking system.

SkillSoft team already had their functionality in place, running the system in parallel with the existing system. Now it was time to perform a cutover and integrate ATM module with the new system. But with a straight cutover Bank would need to plan a downtime period for cutover, anywhere from a few seconds to a few hours, that was critical for client services.

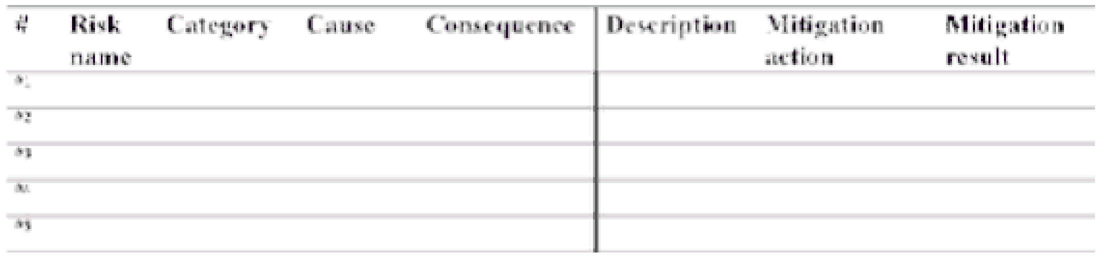
This subject grabbed two last nights for Ivan — as Bank rejected to make cutover during day due to the customers claims.  There were still some gaps between new system functionality and ATM module functionality. And Ivan asked his development and testing team to work overtime to meet deadlines.

A week later all bugs were found and avoided, the ATM software worked properly with the new system which was not still finalized. A client asked for software support and user documentation, and Alexander found out that they had very poor documentation results. He asked Ivan for explanations, but being a programmer, he understood that documentation is a typically risky matter at most projects with strict budget and deadlines. Production of effective documentation is often neglected until it is too late.

Today Ivan came to Alexander with another bad news. Week later after the launch of production one of the key stakeholders asks for some changes to be made in the ATM software. The stakeholder is asking for the change to be implemented immediately. Ivan believes that the changes in question will require much efforts and resources. Alexander should make the final decision whether to implement these changes or not.

### Task:

**1.** List 5 major risks of the deployment stage of the project. Fill in the table:



| # | Risk name | Category | Cause | Consequence | Description | Mitigation action | Mitigation result |
|---|---|---|---|---|---|---|---|
| №1 | | | | | | | |
| №2 | | | | | | | |
| №3 | | | | | | | |
| №4 | | | | | | | |
| №5 | | | | | | | |

**2.** Make a decision regarding implementing a new change after the system go in production. Prove your decision.

# Case 10. Evolution And Maintanance

4 months left after Alexander initiated the ATM software development project with leading bank it grew a very tough project client altered requirements for several times and there was a complex procedure of harmonizing change requests. The most complicated part was integration of the ATM module into the whole banking system which was completely out of order! Alexander made double work, actually. He ensured integration with the old banking system (replica) to check functionality, made tests and ensured its meeting the requirements. On the other hand he made the full integration of ATM module to the new core banking system and some new errors and system faults immediately appeared– his team had to work overtime and on weekends to ensure proper integration and absence of faults.

Due to the fact that the project's start was delayed for 1 week and exceeded the budget, Alexander agreed to conduct 2-days training for users to be able to effectively use ATM back-office software and understand how to operate it and support.

The next topic for negotiation with the Client was how to ensure further maintenance and update the ATM system. They discussed several options:

1. To create a SLA and maintenance contract for 1-2 years

2. To provide the Client IT services with detailed documentation and open source

3. To deal with system integrator (responsible for core banking system maintenance)

Alexander was expecting the 1st option from the Client — it was the best solution for SkillSoft. Alexander knew that maintenance costs usually overweighed development costs.

But here was no obvious option for the Client — as the Bank had already made huge investments in their software systems. Alexander told the Client, that maintenance does not prick commonly involve major changes to the system's architecture. Usually changes are implemented by modifying existing components and adding new components to the system. He suggested two types of maintenance that Skillsoft may provide:

- Maintenance to adapt software to a different operating environment. Changing a system so that it could operate in a different from its initial implementation environment (computer, OS, etc.);
- Maintenance to add to or modify the system's functionality. Modifying the system to satisfy new requirements.

The Client asked for time-out to think over a maintenance proposal and promised to come back to Alexander with the decision in 1-2 weeks.

## Task:

**1.** Describe the main types of software maintenance, explain why it sometimes difficult to distinguish between them?

- Corrective maintenance or fault repair;
- Adaptive maintenance or environmental adaptation;
- Perfective maintenance or functionality addition.