Lameck Onyango Oyare
MSc. Software Engineering

Reg.SCT313 – 3506/2017

ICS 3104: SYSTEMS ENGINEERING

Continuous Assessment II

You've been asked to build software to support a low-cost video editing system. The system accepts digital video as input, stores the video on disk, and then allows the user to do a wide range of edits to the digitized video. The result can then be output to DVD or other media. Do a small amount of research on systems of this type and then make a list of technology risks that you would face as you begin a project of this type.

Every aspect of a software development project could be influenced by risks that could cause project failure. It is common to say that risk is the price of opportunity, i.e. a project with a high number of risks has an opportunity on the global software market if the project is completed on time and within planned expenses In order to complete a complex software development project within planned boundaries, risks on the project should be well understood and mitigated.

Technology risk is the potential for technology shortfalls to result in losses. This includes the potential for project failures, operational problems and information security incidents. The following are common types of technology risks.

1. **Project size:** could be a risk, meaning, while building the video editing system, customers may start to realize what the capabilities of the system are and ask for more functionality, which can cause the size of the project to increase, as well as cost, and complexity.
2. **Shortage of technically skilled staff:** Lack of staff with the knowledge to implement the system.
3. **Compromising on designs:** In order to get stuck into the next 'real' tasks, developers tend to rush the design-process. This is a waste of programming hours, as designing is the most critical part of software development.
4. **Gold plating:** Developers sometimes like to show off their skills by adding unnecessary features. For instance, a developer might add Flash to a basic login module to make it look 'stylish'. Again, this is a waste of programming hours.
5. **Estimation and scheduling:** The unique nature of individual software projects creates problems for developers and managers in estimating and scheduling development time.
6. **Sudden growth in requirements:** As a project progresses, issues that are not identified earlier can create a last-minute hurdle to meeting deadlines. Try to think big early on in the project, and anticipate the worst-case or heaviest-use scenario.
7. **Breakdown of specification:** During the initial phases of integration and coding, requirements might conflict. Moreover, developers may find that even the specification is unclear or incomplete.

8. **Architecture risk:** Is the potential for an architectural design to fail to satisfy the requirements for a project. This includes capacity limitations, poor quality designs, flaws and inefficiencies that are either rejected by the sponsor or impede project work.
9. **Change Control:** A failure to control change to complex systems including practices such as change management and configuration management. is the process of capturing, evaluating, prioritizing, approving, scheduling and implementing change.
10. **Innovation Risk:** A special category of risk associated with experimentation and aggressive rates of change. Typically requires novel approaches to risk management such as designing activities to fail well.
11. **Unclear Project Scope**
12. **Insufficient resources:** Sometimes, the available resources (i.e. people, tools and technologies) are not enough to complete the project. In other cases; the system cannot be implemented using the current available technology where the project involves the use of new technology.

Others risks

- Poor definition of requirements
- Inadequate of requirements: Impossible requirements
- Continuous changing requirements
- No advanced technology available or the existing technology is in initial stages.
- The product is complex to implement.
- Difficult project modules integration.

REFERENCES

1. Sertić, H. (2002): *Applying Unified process on complex software system development,* Master Thesis, Economic Faculty, University of Zagreb, Zagreb

2. Booch, G.; Rambaugh, J.; Jacobson, I. (2001): *The Unified Software Development Process,* Addison-Wesley, New York

3. Karolak, W. (1995): *Software Engineering Risk Management,* Wiley-IEEE Press, San Francisco

4. Royce, W. (1998): *Software Project Management: A unified framework,* Addison-Wesley, New York

5. Larman, C. (2002): *Applying UML and Patterns,* Prentice Hall, Upper Saddle River

6. Capers, J. (1994): *Assessment and Control of Software Risks,* Prentice Hall PTR, Upper Saddle River NJ

7. Ould, M. (1996): *Strategies for Software Engineering: The Management of Risk and Quality,* John Wiley & Sons, San Francisco

8. Charette, R. (1989): *Software Engineering Risk Analysis and Management,* McGraw Hill, New York