Wiki Entry: Risks and risk mitigation. Read the articles by Verner et al (2014) and Anton & Nucu (2020) and then answer the following questions:

What are the main risks that the authors identify? How do these fit into the traditional SDLC model?

The software development lifecycle (SDLC) entails the following six stages (AWS, n.d.):

1. **Plan** - cost-benefit analysis, scheduling, resource estimation, and allocation

The risks here include:

- Differences in physical time and location can increase complexity and increased overhead, and communication challenges may be caused by cultural and organisational differences (Verner et al., 2014).

2. **Design** – covers requirements analysis and solution design

The risks in this stage are:

- Requirements change management issues and communication issues among team members and stakeholders (Verner et al., 2014), especially pertaining to global teams with a mix of off-shore resources and/or subsidiary teams

3. **Implement** – full development and coding stage

The risks here may pertain to:

- Asymmetry in service and systems/processes (Verner et al., 2014) thus causing incompatibility between vendor and client.
- Additionally, there may be issues in number of trained personnel available to do the implementation (Elzamly et al., 2016).

4. **Test** – may run in parallel with the development stage, but entails quality checking and testing of code

The risks here include:

- Issues in distributed development when applying Agile practices due to high level of required interactivity between peers (Verner et al., 2014).
- Asymmetry in function criticality may result in inappropriate time being assigned to testing activities where more time is required (Amland, 2000).

5. **Deploy** – solution deployed to users in production

The risks here cover:

- Evidence of previous delay in deployment and full delivery of software (Verner et al., 2014).
- Conflicting priorities between deployment team and stakeholder/client (Bannerman, 2008).

6. **Maintain** – team solves issues and manages versioning updates and other changes

The risks here include:

- Issues with configuration management, particularly causing an increase in time required to complete maintenance requests (Verner et al., 2014).

Which of the frameworks discussed in the Unit 3 lecturecast would you use to capture and categorise the risks?

I would choose the NIST Risk Management Framework (RMF).

Add a risk and a suggested mitigation to the module wiki.

Risk and Risk Mitigation

Risk: Insufficiently-tested code causes significant failures of the software, thus delaying delivery and incurring losses.

Risk Mitigation: Introduce practices of test driven Agile development (Verner et al., 2014) and risk-based testing to assign additional testing based on risk of failure (Amland, 2000).

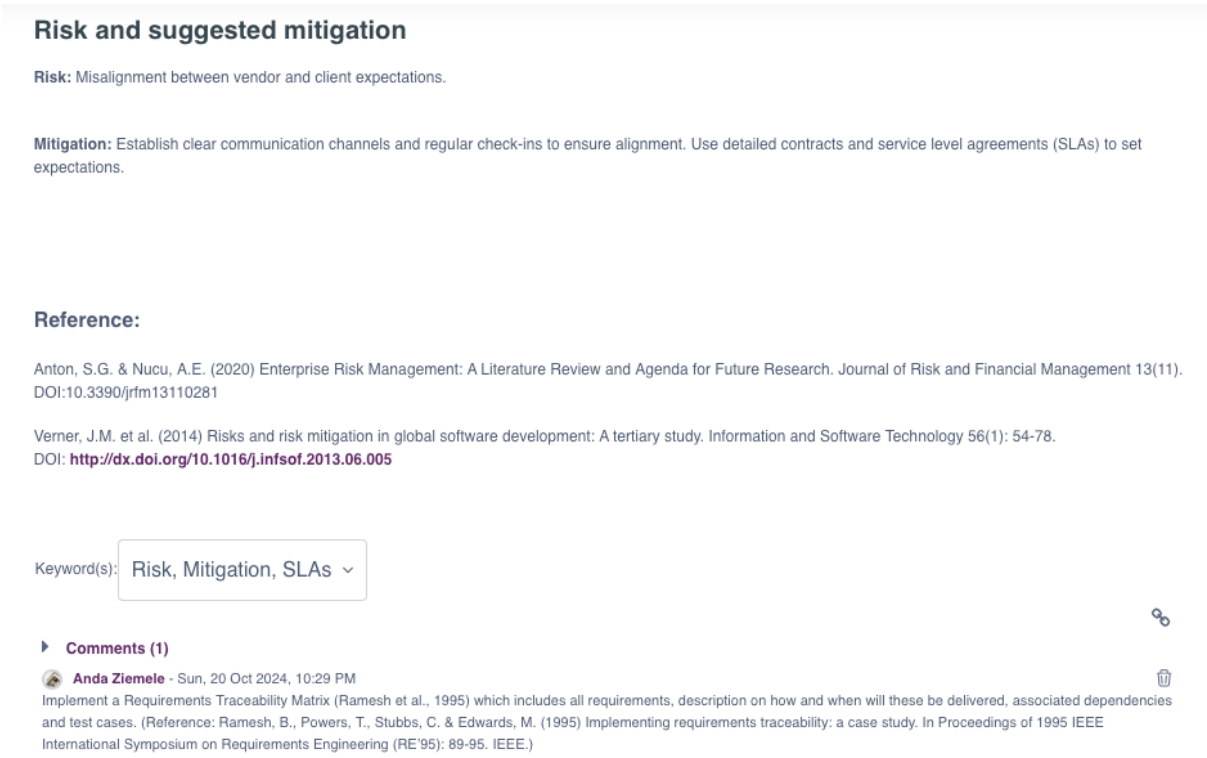
References:

Amland, S. (2000) Risk-based testing: Risk analysis fundamentals and metrics for software testing including a financial application case study. *Journal of Systems and Software*, *53*(3): 287-295.

Verner, J.M., Brereton, O.P., Kitchenham, B.A., Turner, M. & Niazi, M. (2014) Risks and risk mitigation in global software development: A tertiary study. *Information and Software Technology*, *56*(1): 54-78.

Figure 1. Screenshot of Wiki entry.

Choose an entry made by one of your colleagues in the wiki and comment on how you might mitigate it.



Risk and suggested mitigation

Risk: Misalignment between vendor and client expectations.

Mitigation: Establish clear communication channels and regular check-ins to ensure alignment. Use detailed contracts and service level agreements (SLAs) to set expectations.

Reference:

Anton, S.G. & Nucu, A.E. (2020) Enterprise Risk Management: A Literature Review and Agenda for Future Research. Journal of Risk and Financial Management 13(11). DOI:10.3390/jrfm13110281

Verner, J.M. et al. (2014) Risks and risk mitigation in global software development: A tertiary study. Information and Software Technology 56(1): 54-78. DOI: **http://dx.doi.org/10.1016/j.infsof.2013.06.005**

Keyword(s): Risk, Mitigation, SLAs

Comments (1)

Anda Ziemele - Sun, 20 Oct 2024, 10:29 PM

Implement a Requirements Traceability Matrix (Ramesh et al., 1995) which includes all requirements, description on how and when will these be delivered, associated dependencies and test cases. (Reference: Ramesh, B., Powers, T., Stubbs, C. & Edwards, M. (1995) Implementing requirements traceability: a case study. In Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95): 89-95. IEEE.)

References:

Amland, S. (2000) Risk-based testing: Risk analysis fundamentals and metrics for software testing including a financial application case study. *Journal of Systems and Software*, *53*(3): 287-295.

AWS (n.d.) What is SDLC (Software Development Lifecycle)?. AWS. Available from: https://aws.amazon.com/what-is/sdlc/ [Accessed 27 August 2024]

Bannerman, P.L., 2008. Risk and risk management in software projects: A reassessment. *Journal of systems and software*, *81*(12): 2118-2133.

Elzamly, A., Hussin, B. & Salleh, N.M. (2016) Top fifty software risk factors and the best thirty risk management techniques in software development lifecycle for successful software projects. *International Journal of Hybrid Information Technology*, *9*(6): 11-32.

Verner, J.M., Brereton, O.P., Kitchenham, B.A., Turner, M. & Niazi, M. (2014) Risks and risk mitigation in global software development: A tertiary study. *Information and Software Technology*, *56*(1): 54-78.