# What are the 5 overall causes of risk?

As highlighted by Roy et al. (2015), it is vital to identify key risk factors and risk types at each phase of SDLC, rather than conducting a risk assessment on the overall project level. The approach would aid in recognising the risks at a much early phase of development to curtail further spread of risk, causing serious damages and maintenance overhead. This risk management can be best achieved through a prioritised list of risks, after consideration of impact, probability and, frequency of occurrence (Shahzad & Iqbal, 2007).

## *Requirements analysis and definition phase*

Being the initial stage in the process, subsequent impact of any risk occurrences in this phase not only negatively impact project progress but also on the project outcome. Thus, risks reside in this phase are considered with highest severity compared to supplementary phases of SDLC. A study by Hijazi & Alqrainy (2014) concludes that the lack of strategic alignment/miscommunication among stakeholders; unclear/unrealistic requirements; inadequate estimation of project timelines; and unrealistic budgets are significant causes of risk. In the study, the outcome of correlation between each of these risk factors and project failure was above 70%. Roy et al. (2015) also highlight the importance of stability in requirements analysis and clarity in requirements definitions, as continuous changing requirements could cause technical risk at this phase.

## *Design phase*

Most of the risk factors in this phase are consequences of increased design complexity.  A recent study by Al-Shehab et al. (2021) highlights the risk of high technical complexity characterised by immature technology and highly complex tasks, while reconfirming previous research findings by Hijazi et al. (2014a). The study concludes “complexity of a project” as a high impact risk factor with above 70% likelihood of occurrence.

As a risk factor common to all SDLC phases, continually changing requirements also have a negative impact at the design stage, if the requirements have not been fully described at the start of the project (Hijazi et al.,2014b). For example, unclear direction for developers at this stage, may lead to develop a design for a system other than the intended one, especially if developers were not involved in the requirements analysis and definition phase.

## Implementation and unit testing phase

Inexperience technical staff (Roy et al.,2015) and lack of alignment in following coding standards and best practices in programming lead to complex/ambiguous coding (Hijazi et al.,2014a). Inability to understand and reuse codes will cause project delays and budget over-run.

Further, conflicts in time schedules may force programmers to neglect some non-functional requirements or abandon some core functionalities and curtail thorough testing to deliver the project on-time (Hijazi et al.,2014a).

## Integration and system testing phase

Improper testing strategies to recognise ambiguity in the developed product (Roy et al.,2015) and inexperienced testing team who might destroy the whole process by misuse of available tools and resources (Hijazi et al.,2014a), cause difficulties in verifying system integration and ensuring that the software meets the desired requirements.

## Operation and maintenance phase

All budgets are based on forward looking estimations and typically involve some degree of uncertainty. As the longest phase in the SDLC, most of its activities need to be repeated due to corrective maintenance. Such corrective maintenance may exceed the available budget, forcing operation and maintenance phase to be cut (Roy et al., 2015). Moreover, due to the longevity of the phase, any team turnover threats any project's phase, when adequate knowledge of the system's nature and function is not transferred among current teams (Hijazi et al., 2014a; Khan et al., 2014).

As a final note, it is important to not wait until the occurrence of risks, rather either mitigation strategies or contingency plans must be planned in advance.

## *Reference List*

Al-Shehab, A., Alfozan, T. & Gadelrab, H. (2021) Most severe risk factors in software development projects in Kuwait. Indonesian Journal of Electrical Engineering and Computer Science 21(1): 591. DOI: 10.11591/ijeecs.v21.i1.pp591-600 [Accessed 19 February 2021].

Hijazi, H., Alqrainy, S., Muaidi, H. & Khdour, T. (2014a) Identifying Causality Relation between Software Projects Risk Factors. International Journal of Software Engineering and Its Applications 8(2): 51-58.

Hijazi, H., Alqrainy, S., Muaidi, H. & Khdour, T. (2014b) A Framework for Integrating Risk Management into the Software Development Process. Research Journal of Applied Sciences, Engineering and Technology 8(8): 919-928. DOI: 10.19026/rjaset.8.1054 [Accessed 19 February 2021].

Hijazi, H. & Alqrainy, S. (2014) Managing Risks in the System Analysis and Requirements Definition Phase. International Journal of Computer Applications 99(3): 23-29. DOI: 10.5120/17352-7840 [Accessed 19 February 2021].

Khan, K., Qadri, S., Ahmad, S., Siddique, A., Ayoub, A. & Saeed, S. (2014) Evaluation of PMI's risk management framework and major causes of software development failure in software industry. International Journal of Scientific & Technology Research 3(11): 120 – 123. Available From: https://www.ijstr.org/final-print/nov2014/Evaluation-Of-Pmis-Risk-Management-Framework-And-Major-Causes-Of-Software-Development-Failure-In-Software-Industry.pdf [Accessed 19 February 2021].

Roy, B., Dasgupta, R. and Chaki, N. (2015) A Study on Software Risk Management Strategies and Mapping with SDLC. Advances in Intelligent Systems and Computing. 1(1): 121-138. DOI: https://doi.org/10.1007/978-81-322-2653-6_9 [Accessed 19 February 2021].

Shahzad, B., & Iqbal, J. (2007). Software Risk Management – Prioritization of frequently occurring Risk in Software Development Phases. Using Relative Impact Risk Model. 2nd International Conference on Information and Communication Technology. Available From: https://www.researchgate.net/publication/234556365_Software_Risk_Management_-_Prioritization_of_frequently_occurring_Risk_in_Software_Development_Phases_Using_Relative_Impact_Risk_Model [Accessed 19 February 2021].