Information System Failures And How To Prevent Them

Marcelo Fernandes
Wellington, New Zealand
Email: marceelofernandes@gmail.com
November 23, 2021

I. Introduction

Information System (IS) failures have been under the scrutiny of both the press and experts over the last few years in an attempt to reduce IT catastrophes. Despite numerous attempts to solve the issue, reporting on IS failures is still a difficult endeavour due to failures being multi-variable problems that often include human error factors, frequently complex to quantify. Regardless of difficulty, the rate of IS failures is still at a high point [1] and deserves attention in order to enable managers to better understand how to minimise risks that may lead to IS failures.

II. Importance

Firstly, why is it important to talk about IS failures? From a financial perspective, aggregated project failures culminate in wasted revenue, missed opportunity cost, and loss of jobs. From a social perspective failures culminate on loss of tax-payer money that could have been directed to aid important social advancements instead. A few of the frequently quoted examples used to illustrate such failures are:

- ***MasterNet (Bank of America) Accounting system*** [2]: $80 million in costs due to mismanagement and bureaucracy within the IT department. 9,600 jobs were cut as a result.
- ***KMart's IT system modernisation*** [3]: $1.4 billion investment failing after $130 million had already been spent, culminating in KMart declaring bankruptcy.
- ***Internal Revenue Service (U.S.A)*** [3]: $8 billion of taxpayers money spent on a system that failed due to mismanagement and planning.

III. Biggest Issues

It is important to firstly understand what factors contribute the most to IS failures and secondly, if it is possible to cross-examine failures across many organisations and establish common points between them. A study from 2008 [1] attempted to achieve this by going through a catalogue of 99 postmortem examinations on IS failures from 74 different organisations. The study summarises that the overwhelming majority of mistakes leading to IS failures was shared between two categories: process and people.

- Process (45%). Common failures in this category include:
  - Wasted time in the beginning of a project spent on many things except the development of the project itself, often called the "fuzzy front end" decision-making step [4]. This time is usually spent on budgeting, requesting approvals, general paper work, and identifying concepts and opportunities. An unsuccessful governance will stay in this phase of the project for months and then start the project already running on a tight schedule.
  - Humans generally produce overly optimistic predictions of when a given task will be completed [5]. This can potentially jeopardise the project by undermining thorough planning, as tasks will seem to be more trivial than they actually are.
  - Lack of risk analysis and management of risk. Typical risks in this category include: Too much scope in the project (scope creep), failing to secure employees to finish the project, and lack of planning for a possible sponsorship pull out (budget changes).
  - A rising number of companies outsourcing relevant parts of their business has led to a rising number of failures [6].
- People (43%):
  - Low motivation, which has been associated in other works with decreases in productivity and work quality [7].
  - Poor relationships between coworkers [8].
  - Management inaction on dealing with a problematic employee [9].
  - Adding human resources to a project at the end of its deadline, which is associated with a drop in the team's productivity [10].

The other categories of mistakes were: product (8%) and technology (4%). As counter-intuitive as it sounds to many in the technology industry, the evidence suggests that the technology that is used to deliver a project is seldom the primary cause of a failure, stressing that managers should focus mostly on processes and people.

IV. Recommendations for Avoiding the Biggest Issues

After the analysis provided through the examination of 99 failures [1], some thought was put forward to provide techniques to help organisations mitigate some of the most classic mistakes.

- ***Poor risk management***: Recommendations to prevent this issue are:
  - Identify the key risks by actively managing a list with the top 15 risks.
  - Active prioritisation and resolution for the most important risks through risk planning.
  - Active monitoring resolved issues as well as non-resolved ones in an analytical way so that future metrics can be easily examined.
  - Have someone in the team wear a risk officer hat and look for red-flags that could culminate in project failure during retrospective meetings.
- ***Poor estimations and schedule***: Some techniques to avoid this issue include:
  - Establishing a limit threshold for development time as smaller projects are simpler to estimate.
  - Create a team culture of breaking work down to limit the size and scope of tasks.
  - Use retrospective meetings as a way to capture the actual time it took to complete a task in order to use it in future estimations.
- ***Poor planning***: Rushing the development of a project in order to keep upper management satisfied can cause project managers to neglect the planning step and end up with a failed project. To mitigate this, it is paramount to:
  - Have unambiguous roles, duties, and responsibilities within the team.
  - Set up a project prioritisation list, so that side-way projects don't drive resources to other places.
  - Officially carry through project policies about implementation of procedures and governance.
- ***Poor quality assurance***: In order to meet deadlines for projects lagging behind, managers might be tempted to ask software developers to reduce time spent both on unit testing and on code reviews. However, it has been assessed that for each day in the early times of a project without quality assurance can cost between 3 to 10 days in future [11].
- ***Poor staff and/or work relationships***: A third of projects may experience some kind of staff or work relationship issue [1]. Getting the right people to develop a project from the beginning is paramount, as relying on programmers with low skills and/or experience may generate poor quality code and many technical debts, delaying the schedule and eventually creating staff turnover if the code-base becomes unmanageable.

V. Conclusions

The data analysed shows evidence that failures are likely to be people or process-based, instead of product or technology-based. Following that, project managers should aim efforts in such areas first, whilst also implementing the best practices mentioned above on a continuing basis to prevent such mistakes as much as possible. A proactive use of such practices along with a retrospective meeting plan to evaluate ongoing issues and risks in a project are the formula to minimise the chances of becoming a famous IS project failure.

References

[1] R. Nelson, "IT Project Management: Infamous Failures, Classic Mistakes, and Best Practices", MIS Quarterly Executive, vol. 6, no. 2, 2008 [Online]. Available: https://aisel.aisnet.org/misqe/vol6/iss2/4. [Accessed: 23- Nov- 2021]

[2] D. Frantz, "$80-Million Failure : B of A's Plans for Computer Don't Add Up", Los Angeles Times, 1988. [Online]. Available: https://www.latimes.com/archives/la-xpm-1988-02-07-mn-41322-story.html. [Accessed: 23- Nov- 2021]

[3] R. Charette, "Why Software Fails", IEEE Spectrum, 2005. [Online]. Available: https://spectrum.ieee.org/why-software-fails. [Accessed: 23- Nov- 2021]

[4] K. Eling, A. Griffin and F. Langerak, "Using Intuition in Fuzzy Front-End Decision-Making: A Conceptual Framework", Journal of Product Innovation Management, vol. 31, no. 5, pp. 956-972, 2013.

[5] R. Buehler, D. Griffin and M. Ross, "Exploring the "planning fallacy": Why people underestimate their task completion times.", Journal of Personality and Social Psychology, vol. 67, no. 3, pp. 366-381, 1994.

[6] L. Willcocks and M. Lacity, Global sourcing of business and IT services. Basingstoke England: Palgrave Macmillan, 2006.

[7] B. Boehm, "An Experiment in Small-Scale Application Software Engineering", IEEE Transactions on Software Engineering, vol. -7, no. 5, pp. 482-493, 1981.

[8] B. Lakhanpal, "Understanding the factors influencing the performance of software development groups: An exploratory group-level analysis", Information and Software Technology, vol. 35, no. 8, pp. 468-473, 1993.

[9] C. Larson and F. LaFasto, TeamWork: What Must Go Right/What Can Go Wrong. Newbury Park u.a.: Sage, 1990.

[10] F. Brooks, The mythical man-month. Boston: Addison-Wesley, 2005.

[11] C. Jones, Assessment and control of software risks. Englewood Cliffs, N.J: Yourdon Press, 1994.