Unit 2

Information Systems and their importance (Reflection)

Based on my initial understanding of what constitutes an information system, I further expanded on the understanding by adding hardware and software components. In unit 1, my understanding was narrow—people, data, and processes—and going through this unit's content broadened that view to be more inclusive of the technology components (hardware and software)

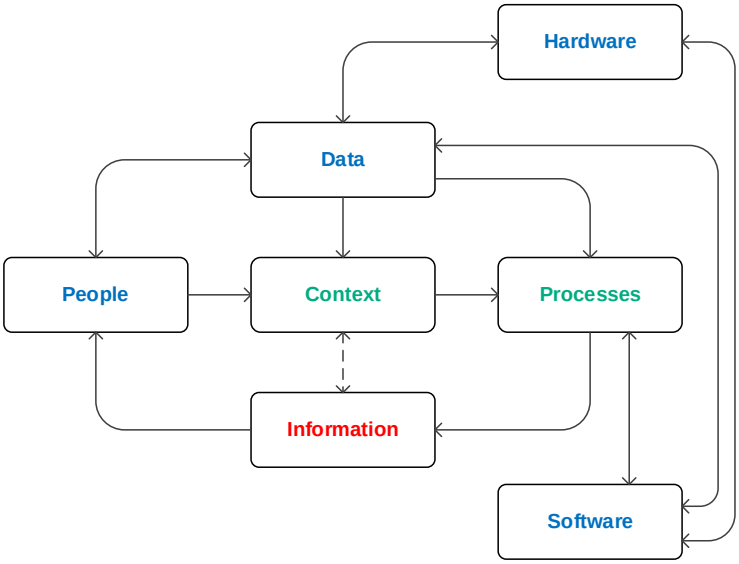

Figure 1 Components of an information system (created by the author)

Adding hardware and software components to an information system *increases* the surface area of potential failures within that system. I think of links within a chain and the saying that the chain is only as strong as its weakest link. I feel the same is true of information systems, whereby every additional component becomes a potential weak link. As a result, managing and designing information systems is complex as each technology component (hardware and software) requires evaluation against how well they meet the system's requirements, applicable legislation, cost, environmental impact, service agreements, and other relevant factors. For instance, cloud computing allows organisations to move their data from local on-

premises to cloud-hosted. However, such a move brings risk, including data security and availability. Plus, there is the added requirement to incorporate a mandatory internet connection into the system's design.

During this unit, I identified additional components (represented by dashed bubbles) of an information system beyond purely people, data, hardware, and software:

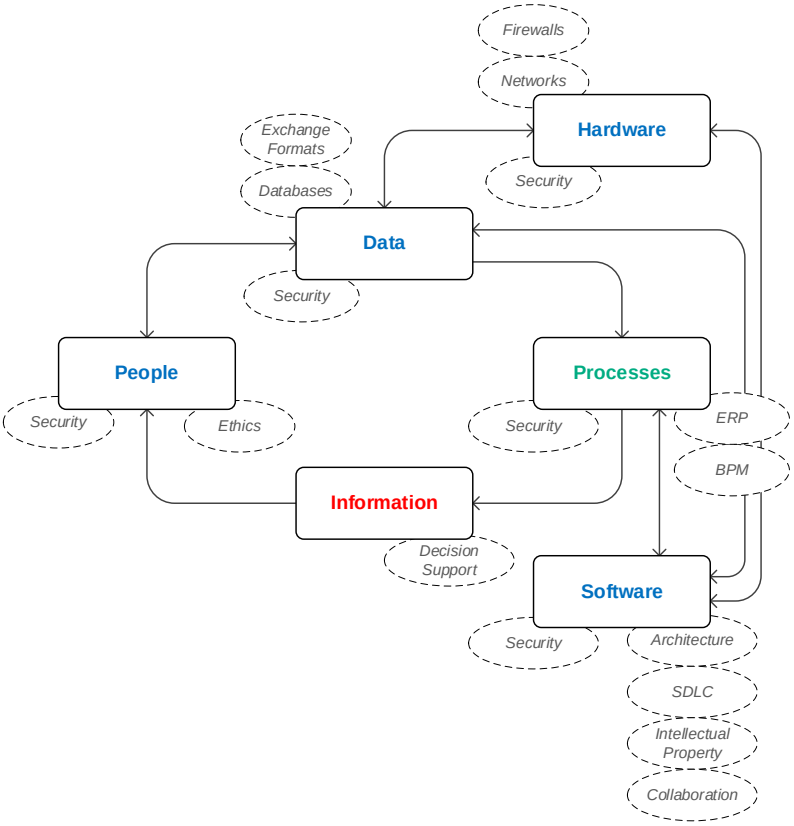

Figure 2 Components of an Information System (created by the author)

What struck me is the amazing depth of complexity contained by information systems as we zoom deeper into the details. In a way, I think this progressive disclosure of complexity is a beautiful example of what software/enterprise architects deal with in their careers: abstract away system details when not required but provide them when called for. From adding to the complexity perspective, each rectangle in Figure 2 acts as a potential source of failure. Therefore, it is impossible to build information systems free from defects due to the sheer number of interacting components. Also shown is that the security component is pervasive throughout an information system (NIST, 2018) because data is a fundamental aspect of any information system and requires protection from unauthorised use and access.

Continuing the collaborative discussion this week, what stood out as the main reasons for system failure is poor testing, lack of requirements, bad data assumptions and risk management.

1. **Testing.**

   Unsurprisingly, lack of testing seems evident in almost all the case studies presented for discussion. I say, “almost all”, because sometimes even the best managed systems fall foul due to power outages, broken hardware or circumstances beyond control and these circumstances cannot be tested. However, as I look at the case studies, lack of proper testing seems to be the number one culprit

2. **Requirements.**

   I considered whether developing a system is possible without all system requirements in place. While fellow students argued for the case that lack of requirements is a cause of failure, I do not believe this is true chiefly because it is impossible to capture *every* requirement of a system. Even writing *good quality* requirements is a difficult task that fades over time as team members settle for the path of least resistance. However, according to Saleh & Al-Zarouni (2004), the absence of non-functional requirements is one cause of information system failures thereby lending credence to the idea that requirements play a role in the success or failure of information systems.

3. **Assumptions.**

   Information systems can fail because of poor, unrealistic *assumptions* made about data. For example, assumptions that data values will never go out of allowed range; data will always be always present; data will be clean, and well-formatted, or that processes will present data in a timeous manner. These are poor assumptions. I considered a discussion about a limit on the number of 911 emergency calls allowed (Swanlow, 2021) where the system designers felt a limit of 40 million was sufficient. However, the limit was inevitably breached and lead to system failure. This reaffirmed the idea that had the designers leveraged good quality data to base their assumptions (PopulationPyramid, 2019) the system failure likely would not have occurred.

4. **Risk.**

   A lack of proper risk-management policies was raised in discussion as a potential source for information system failures. Mitigating risk considers prioritising code

> reviews, testing, and collaboration during the development phase of a Software Development Lifecycle. But I argue that Risk Management is part of a grander organisation-wide governance policy and not merely an isolated structure for managing only software development and deployment. As the number of companies increase their digital footprint, risk management becomes more a requirement, and no longer merely a need. (Tupa et al., 2017)

Investigating further, Farshidi et al. (2020) introduce system architecture as a prime component behind the success or failure of information systems. Architecture is about applying well-tested design patterns and managing complexities**.** It is highly likely that specific patterns, though good on paper and in theory, for instance, thick clients, are not suitable for devices with limited processing capability. Architects strive to lay a solid foundation in the beginning stages of information system design. These foundations determine the system's ability to cater to future requirement changes. Adopting the right architecture for the wrong scenario is bound to lead to failure; therefore, I agree with the position that architecture is an active factor involved in information system failures.

References

Farshidi, S., Jansen, S. & Werf, J.M van der (2020) Capturing software architecture knowledge for pattern-driven design. The Journal of Systems & Software. DOI: https://doi.org/10.1016/j.jss.2020.110714

NIST (2018) Risk Management Framework for Information Systems and Organisations: A System Life Cycle Approach for Security and Privacy. DOI: https://doi.org/10.6028/NIST.SP.800-37r2

PopulationPyramid (2019). Population Pyramids of the World. Available from https://www.populationpyramid.net/united-states-of-america/2014/ [Accessed on 16 May 2021]

Saleh, K. & Al-Zarouni, A. (2004). Capturing non-functional software requirements using the user requirements notation. The 2004 International Research Conference on Innovations in Information Technology: 222-230.

Swanlow, S. (2021) Object-oriented Information Systems May 2021. Available from https://www.my-course.co.uk/mod/hsuforum/discuss.php?d=254915.

Tupa, J., Simota, J. and Steiner, F. (2017) Aspects of Risk Management Implementation for Industry 4.0. *Procedia Manufacturing* 11:1223–1230, DOI: https://dx.doi.org/10.1016/j.promfg.2017.07.248.