Risk Assessment Report

Deciding upon the best ERP solution for Acme Manufacturing

1. Introduction

Acme manufacturing has decided to purchase an ERP (Enterprise Resource Planning) system to enable a more flexible supply chain management. Three solutions have been short-listed: a COTS (Commercial Off the Shelf), an Open-Source and an In-House solution.

In the following, this report aims to determine the most suitable option for Acme and provide a Disaster Recovery Plan for the recommended solution.

2. Methodology

In this study, secondary analysis of qualitative data provides an opportunity to maximise data utility, particularly with limited access to system users and lack of numerical data. Qualitative risk analysis is more credible in this study, given the impossibility to measure and quantify certain risks numerically. It is well suited to understanding the effect of IT-related innovations on organisational contexts (Darke et al.,1998).

In assessing the risk and deciding upon the best approach, the report uses the widely-recognised NIST 800–30 risk assessment framework. It is especially helpful to evaluate and convey IT risks in a concise language for business leadership to make informed decisions (Peacock, N.D.).

Risks for each short-listed solution are first identified, and then assessed based on their likelihood and business impact, using a risk-level matrix as outlined in SP 800-30 (Stoneburner et al., 2002).

3. Risk Identification

3.1 COTS Solution

Using a commercially available software solution provides a unique set of benefits and costs. The most apparent differences include an annual fee to a third party and relying on that third party for technical support. This allows the IT professionals to focus on other aspects of the system, such as cybersecurity and hardware reliability.

The biggest risk involves third-party technical support. A slow response to a problem that can't be solved in-house could significantly impact meeting the requirements for RPO and RTO. This risk can be mitigated by proper research into COTS and how effective their support is. Another risk is the possibility of the annual cost going up enough to make it no longer viable. This risk could be mitigated by negotiating a contract that covers how much the service fee is allowed to increase, but it is difficult to know if such an addition will be allowed. There are also a few risks shared with the other solutions. The most likely is user difficulty in adapting to the new solution. This will lead to lost time for the first few weeks the system is implemented, and perhaps longer if the UI is unintuitive. A commercial-grade

solution will most likely have a well-implemented interface, so the overall impact will be lower than with other solutions.

3.2 Open-Source Solution

The ability to achieve high vendor independence - possibility of in-depth customisation, while allowing freedom from upgrades - and potential cost reductions, such as in low licence and hardware are reasons to consider open-source-solutions (Johansson & Sudzina, 2008; Bajaj & Ojha, 2016; Sanchez et al., 2020), making it advantageous for SMEs.

Nevertheless, high-quality open-source-solutions are based mainly on having a skilled sustainable community for timely development, effective debugging, and upgrades (Aberdour, 2007; McAllister & O'Hara, 2016; Wen, 2018). Therefore, an adaption of Open-Source ERP could be useless or even harmful, if there is a lack of effective and timely community support for maintenance and integration problems (Ayala et al., 2012), given that 75% codebase contained security vulnerabilities and 49% were high-risk vulnerabilities based on the 2019 Open Source Security and Risk Analysis Report (Synopsys, 2020).

As open-source use grew in popularity, high use of unmanaged open-source components lead to licence conflicts, introducing significant legal and security risks that could cause financial loss (Duan et al., 2017). Accordingly, a key part of any open-source management strategy is to initially concentrate efforts on licence identification (Wu et al., 2015; Synopsys, 2020).

Maintaining an accurate, up-to-date codebase is also vital for high-quality, compliant, and secure solutions. Recent research highlights that 91% of open-source components were more than four years out-of-date, which lead businesses to the risk of attacks (Synopsys, 2020). Therefore, a strategy should be in place to adapt security updates rapidly and have a process for identifying and patching known issues on time to avoid higher costs due to re-work (Piiparinen & Peura, 2020).

3.3 In-House Solution

When developing in-house solutions, the development aspect of writing and testing the application code and the operations, such as deployment and system administration, have to be considered. As Bass (2018) describes, DevOps offers a set of practices that ensure code quality and stability whilst reducing time to release changes by automation. However, the traditional DevOps approach is not addressing one vital aspect: security. The DevSecOps approach builds on top of DevOps; additionally, it addresses application and infrastructure security from the start. This is done by automating security testing and integrating it transparently into the CI/CD pipeline (Gartner, 2018). Figure 1 describes a sample lifecycle for this approach.

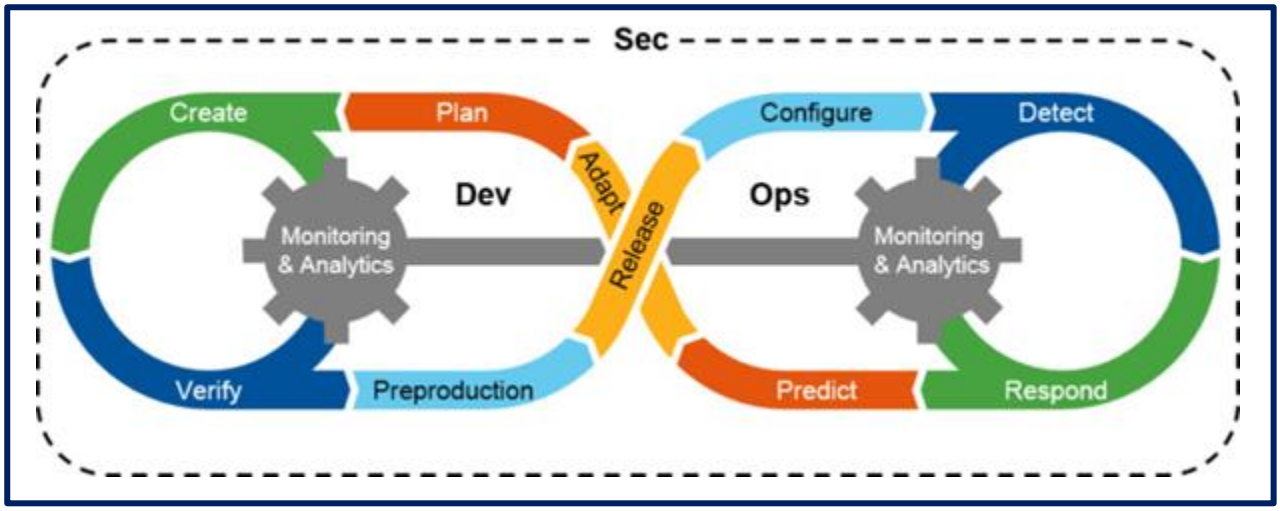

Figure 1: DevSecOps Continuous Delivery Lifecycle (Gartner, 2018)

The DevSecOps approach will be used to implement the in-house ERP solution for Acme manufacturing. Despite the benefits of the approach, there are risks to consider.

The success of the implementation may be impacted due to insufficient resources and expertise. Knowledge of automating processes and security architecture is a prerequisite for successful implementation. DevSecOps allows risk spreading across the whole team; however, the in-house solution at Acme is being built by only a student and internal IT department. If resources and expertise are spread too thinly, bottlenecks arise, causing delays and additional costs. Furthermore, a smaller team may not allow for sufficient knowledge sharing and code documentation, making team-turnover a critical cause for risk throughout the SDLC (Shahzad et al, 2010).

Inexperienced developers increase the likelihood of risk during the design and implementation phase of the SDLC. If an inappropriate architectural design (AD) is chosen, the system implementation might not be completed successfully. During the implementation phase, many risks are caused by insufficient code readability and reusability. If code is not well-documented, reviews, and extensions to the system will be hard to execute, causing long-term stability issues (Hijazi et al, 2014).

Due to a highly automated testing phase, the development team must write sufficient test cases to cover crucial business scenarios. Incomplete testing causes the system to be released with problems that impact its useability. If test cases inside the CI/CD pipeline are insufficient, future updates may further deteriorate the system.

4. Risk Assessment

Following the SP 800-30, a risk-level matrix should be developed to determine each option's level of risk. To do so, the likelihood of occurrence and the magnitude of impact for each risk must be classified (NIST, 2012).

Risk likelihood indicates the probability that a potential risk may occur within the business environment of Acme manufacturing. Figure 2 depicts the five risk likelihood levels.

Level	Likelihood of Occurrence
Very Low	Theoretically possible. May only occur under exceptional circumstances.
Low	Unlikely to expect the event to occur at some time, under normal business conditions.
Medium	Reasonable to expect the event could occur at some time, under normal business conditions.
High	Likely to expect the event to occur at some time, under normal business conditions.
Very High	Virtual certainty the event will occur at some time, under normal business conditions.

Figure 2: Classification of Risk Likelihood

The second factor, risk impact, measures the magnitude of effect on the business operations and its assets based on a qualitative assessment. Figure 3 depicts the five impact levels.

Level	Magnitude of Impact
Very Low	Little to no damage or operational impact. No reputational or financial impact.
Low	Localized or minimal damage or operational impact. Minor reputational and financial impact
Medium	Noticeable damage or operational impact. Harmful reputational and financial impact, but not enough to ruin the business.
High	Major damage or operational impact. Extensive reputational and financial impact, but not enough to ruin the business.
Very High	Critical, long-term damage or operational impact. Financial and reputational damage could be enough to ruin the business.

Figure 3: Classification of Risk Impact

Having allocated these levels to the identified risks, the final determination of mission risk can then be derived by assigning a probability to each likelihood- and a specific value to each impact level. By multiplying both, the level of risk can then be allocated (Stoneburner et al., 2002). Figure 4 depicts the developed risk-level matrix.

Acme Manufacturing Risk-Level Matrix		Impact				
		Very Low (5)	*Low* (25)	*Medium* (50)	*High* (75)	*Very High* (100)
Likelihood	***Very High*** (1.0)	5	25	50	75	100
	High (0.7)	3.5	17.5	35	52.5	70
	Medium (0.5)	2.5	12.5	25	37.5	50
	Low (0.2)	1	5	10	15	20
	Very Low (0.05)	0.05	1.25	2.5	3.75	5

Risk Scale: Severe (>70), Medium (>30 to 70), Low (>10 to 30) and Very Low (<10)

Figure 4: Risk-Level Matrix for Acme Manufacturing

By evaluating the risks identified in chapter 3, using the established risk-level matrix, it is apparent that both the open-source and in-house solution face medium to severe risks, whereas the COTS solution faces low to medium risks. Figure 5 depicts the complete list of results.

COTS (Commercial Off The Shelf) Solution			
Risk	**Likelihood**	**Impact**	**Assessment**
Slow response from third-party technical support	Medium	High	Medium (37.5)
User adaption might be difficult	High	Medium	Medium (35)
Increase of annual/subscription fee	Medium	Medium	Low (25)
Solution difficult to integrate in current business operations	Low	High	Low (15)
Open-Source Solution			
Risk	**Likelihood**	**Impact**	**Assessment**
Codebases contain high-risk vulnerabilities due to lack of skilled community support	High	Very High	Severe (70)
License conflicts or no license declaration for components	High	High	Medium (52.5)
Codebases contain at least one low-risk vulnerability due to lack of skilled community support	High	Low	Low (17.5)
Out-dated codebases due to reliance on community support	Very High	Very Low	Very Low (5)
In-House Solution			
Risk	**Likelihood**	**Impact**	**Assessment**
Insufficient resources and expertise for DevSecOps approach	Very High	High	Severe (75)
Team-turnover leading to bottlenecks and expertise loss	High	High	Medium (52.5)
Inappropriate architectural design (AD) due to inexperience	Medium	Very High	Medium (50)
Insufficient code readability and reusability due to inexperience	Medium	High	Medium (37.5)
Lack of full automated test coverage in CI/CD pipeline	Medium	High	Medium (37.5)

Figure 5: Classification of Identified Risks for Acme Manufacturing

5. Cost-Benefit Analysis

The selection of an open-source solution includes a codebase with numerous open-source components, causing medium-severe risks. As the open-source ERP only contains basic functionality, it demands an internal IT team with significant ERP expertise for customisation, implementation, and maintenance.

Having limited employee capacity, both the open-source and in-house solutions demand additional resources to be implemented effectively. Furthermore, an internally managed ERP project redirects Acme's operational focus away from its core business.

In conclusion, both in-house and open-source solutions provide notable cost savings and flexibility compared to a COTS solution; however, they pose drawbacks through medium-severe risks, additional resources, and distraction from core operations. Given the above analysis, it is recommended to choose the COTS solution, which helps Acme transfer maintenance accountability to a third-party supplier with a fee, while minimising risks to low-medium.

6. Disaster Recovery Solution Design

There are two main benchmarks for the disaster recovery solution: a recovery point objective (RPO) of 15 minutes and a recovery time objective (RTO) of four hours. This means that in the case of a database failure, the most recent backup must have been less than 15 minutes ago, and that backup must be able to be reconstructed and fully operational within four hours.

There are three common types of disaster recovery: hot site, cold site, and warm site. Hot site backups are updated and kept operational in real-time. This leads to very low RPO and RTO but can be very costly to implement and operate (Al-Sharidah and Al-Essa, 2017). Cold site backups are updated infrequently and only run at minimum capacity. This lower cost significantly, but can have RTO as high as a few days. Warm site backups are a compromise of the low recovery time of hot backups and the lower costs of cold backups. Backups occur frequently and are preconfigured to lower recovery time (Al-Sharidah and Al-Essa, 2017).

A solution that utilises warm backups is the best choice for meeting requirements at a reasonable cost for the suggested COTS solution. A full backup of the entire database will occur once per day. Additionally, every ten minutes a differential backup will record the files of the database that have changed. This will ensure that these frequent backups won't require an extensive amount of storage space. When a database failure occurs, it can be reconstructed by restoring the last full backup and then overwriting the files that were changed from the most recent differential backup.

Figure 6 depicts the sequence of actions to be taken to meet the agreed service level agreements. Figure 7 shows an example of how a differential backup is performed.

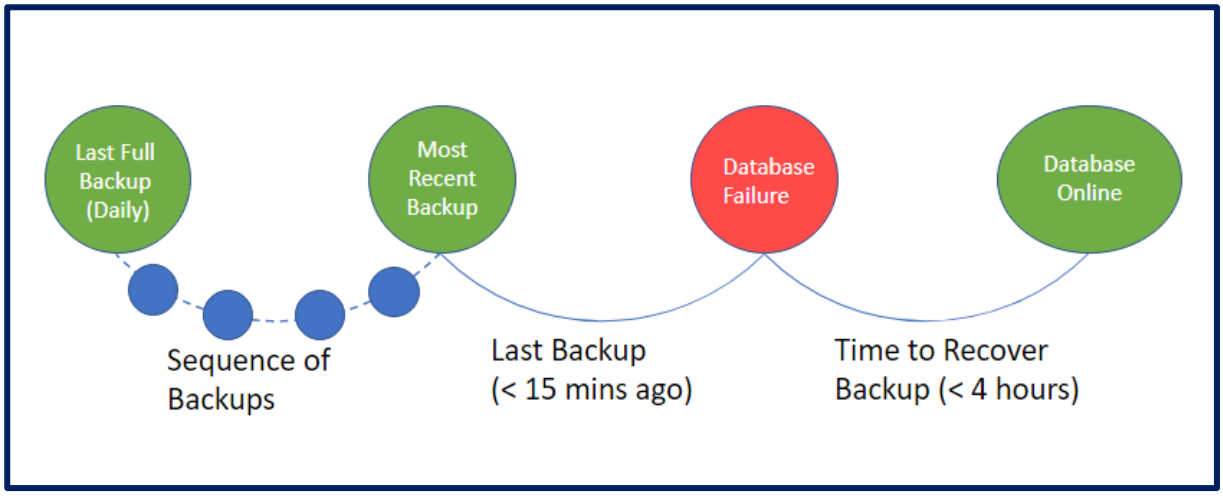

Figure 6: Sequence of Backups for meeting service level agreement

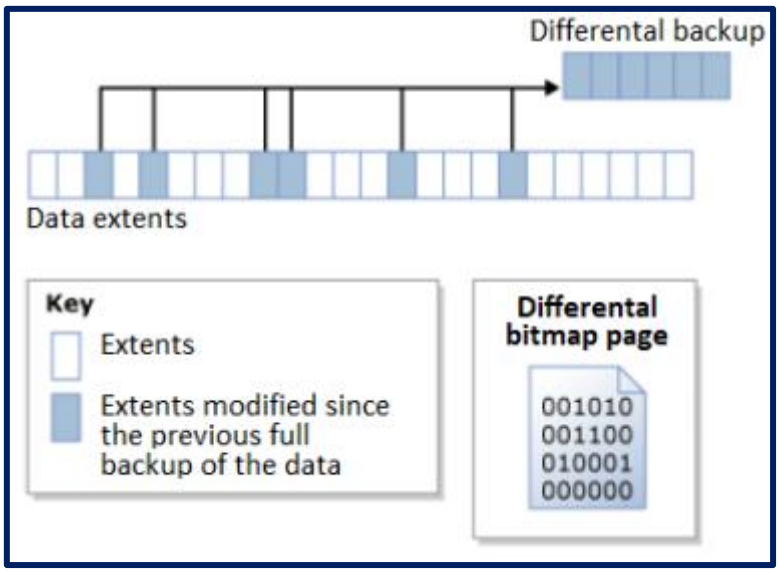

Figure 7: The principle of making a differential backup (Dudjak et al, 2018)

Reference List

Aberdour, M. (2007) Achieving Quality in Open-Source Software. IEEE Software. 24(1): 58-64. DOI: 10.1109/MS.2007.2.

Al-Sharidah, A. & Al-Essa, H. (2017) Toward cost effective and optimal selection of IT disaster recovery cloud solution. *2017 9th Computer Science and Electronic Engineering (CEEC)*. Colchester. IEEE Software. 43-48.

Ayala, C. et al. (2012) OSS Integration Issues and Community Support: An Integrator Perspective. *IFIP Advances in Information and Communication Technology*. 378: 129-143. DOI: 10.1007/978-3-642-33442-9_9.

Bajaj, S. & Ojha, S. (2016) Comparative analysis of open source ERP softwares for small and medium enterprises. *3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. New Delhi, India, 16-18 March. IEEE Software. 1047-1050.

Bass, L. (2018) The Software Architect and DevOps. *IEEE Software*, 35(1): 8-10. DOI: https://doi.ieeecomputersociety.org/10.1109/MS.2017.4541051 [Accessed 27 February 2021].

Darke, P., Shanks, G. & Broadbent, M. (1998) Successfully completing case study research: combining rigour, relevance and pragmatism. Information Systems Journal. 8 (4): 273-289. DOI: https://doi.org/10.1046/j.1365-2575.1998.00040.

Duan, R., Bijlani, A., Xu, M., Kim, T. & Lee, W. (2017) Identifying Open-Source License Violation and 1-day Security Risk at Large Scale. *In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. Dallas, Texas, October. New York: Association for Computing Machinery. 2169–2185.

Dudjak, M., Lukic, I., Kohler, M. (2018) Survey of Database Backup Management. Available from: https://bib.irb.hr/datoteka/934634.180407_OTO2018_Zbornik_za_tisak.pdf#page=153 [Accessed 04 March 2021].

Gartner (2018) A Risk-Adaptive Approach. Available from: https://www.gartner.com/teamsiteanalytics/servePDF?g=/imagesrv/media-products/pdf/Forcepoint/Forcepoint-1-4YCDU8P.pdf [Accessed 27 February 2021].

Hijazi, H., Alqrainy, S., Muaidi, H. & Khdour, T. (2014) Risk Factors in Software Development Phases*. European Scientific Journal* 10(3): 213-231. Available from: https://www.researchgate.net/publication/266144501_Risk_Factors_in_Software_Development_Phases [Accessed 27 February 2021].

Johansson, B. & Sudzina, F. (2008) ERP systems and open source: An initial review and some implications for SMEs. *Journal of Enterprise Information Management* 21(6): 649-658. DOI: 10.1108/17410390810911230.

McAllister, A. & O'Hara, S. (2016) Toward Effective Management of Large-Scale Software. *2016 IEEE/ACM 3rd International Workshop on Software Engineering Research and Industrial Practice (SER&IP)*. Austin, Texas, 17 May. IEEE Software. 16-22.

NIST (2012) *Guide for Conducting Risk Assessments: NIST Special Publication 800-30 Revision 1.* Gaithersburg: National Institute of Standards and Technology. Available from: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-30r1.pdf [Accessed 2. March 2021].

Peacock, J. (N.D.) What is NIST SP 800 30. Cybersaint Security. Available from: https://www.cybersaint.io/blog/what-is-nist-sp-800-30 [Accessed 7. February 2021].

Piiparinen, J. & Peura, P. (2020) Sustainable Software Development & DevOps. Available from: https://www.researchgate.net/publication/345984579_Sustainable_Software_Development_DevOps [Accessed 01 March 2021].

Sanchez, V., Neira, P., Galindo, J. & Benavides, D. (2020) *Open Source Adoption Factors—A Systematic Literature Review*. Available from: https://www.researchgate.net/publication/341260496_Open_Source_Adoption_Factors-A_Systematic_Literature_Review [Accessed 24 February 2021].

Shahzad, B., Al-Mudimigh, A., & Ullah, Z. (2010) Risk identification and preemptive scheduling in software development life cycle. *Global Journal of Computer Science and Technology* 10(2): 55–63. Available from: https://www.researchgate.net/publication/234721439_Risk_Identification_and_Preemptive_scheduling_in_Software_Development_Life_Cycle [Accessed 27 February 2021].

Stoneburner, G., Goguen, A. & Feringa A. (2002) *Risk Management Guide for Information Technology Systems: SP 800-3*. Gaithersburg: National Institute of Standards and Technology. Available from: https://dl.acm.org/doi/pdf/10.5555/2206240 [Accessed 2. March 2021].

Synopsys (2020) 2020 Open Source Security and Risk Analysis (OSSRA) Report. Available from: https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/2020-ossra-report.pdf [Accessed 27 February 2021].

Wen, S. (2018) Learning secure programming in open source software communities: a socio-technical view. *In Proceedings of the 6th International Conference on Information and Education Technology (ICIET '18)*. Osaka, Japan, January. New York: Association for Computing Machinery. 25–32.

Wu, Y., Manabe, Y., Kanda, T., German, D. & Inoue, K. (2015) A Method to Detect License Inconsistencies in Large-Scale Open Source Projects. *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. Florence, Italy, 16-17 May. IEEE Software. 324-333.