Metamycelium: a domain-driven distributed reference architecture for big data systems

Pouya Ataei[1][0000−0002−0993−3574] and Alan Litchfield[2,3][0000−0002−3876−0940]

[1] Auckland University of Technology, Princeton NJ 08544, USA
[2] pouya.ataei@aut.ac.nz
http://www.springer.com/gp/computer-science/lncs
[3] School of Engineering, Computer and Mathematical Sciences
{abc,lncs}@uni-heidelberg.de

Abstract. The abstract should briefly summarize the contents of the paper in 150–250 words.

Keywords: First keyword · Second keyword · Another keyword.

1 Introduction

The ubiquity of digital devices and proliferation of software applications have augmented users to generate data at an unprecedented rate. In this day and age, almost all aspects of human life is integrated with some sort of software system, that by large, is processing data, and executing the necessary computations.

According to Internetlivestats.com [27] in one second 3,135,050 emails are sent, 1,151 Instagram photos are uploaded, 6,738 Skype calls are made and 147,084 GB of traffic has gone through the internet. That is, in the last minute, 825.04 terabytes have been transferred through the internet.

The rapid expansion and evolution of data from an structured element that is passively stored to something that is used to support proactive decision making for business competitive advantage, have dawned a new era, the era of Big Data (BD). The era of BD began when velocity, variety and volume of data overwhelmed traditional systems used to process that data [3][4].

BD is the practice of crunching large sets of heterogenous data to discover patterns and insights for business competitive advantage [24][13].

Since the inception of the term, ideas have ebbed and flowed along with the rapid advancements of technology, and many strived to harness the power of data. Nevertheless, BD is not a magical wand that can enchant any business processes and many have failed to absorb the complexity of this new field. According to a recent survey by MIT Technology Review insights in partnership with Databricks, only 13% of organizations excel at delivering on their data strategy. Another survey by NewVantage Partners highlighted that only 24% of organizations have successfully adopted BD [22].

Sigma computing report indicated that 1 in 4 business experts have given up on getting insights they needed because the data analysis took too long [9].

Along the lines, McKinsey & Company [?] and Gartner [30] demonstrated that approximately only 20% of organizations have fully adopted BD. These statistics unveil the truth that succesful adoption of BD system is scarce.

Among the challenges of adopting BD, perhaps the most highlighted are "cultural challenges of becoming data-driven", "BD architecture", "data engineering complexities", "rapid technology change", and "lack of sufficiently skilled data engineers" [26][23]. In the past, organizations relied on a few technology giants to account for their analytics and storage needs, whereas today's ecosystem of BD encompasses far-reaching plethora of technologies ranging from visualization to high-level sql-like scripting languages to logging, stream processing and distributed storage.

On the other hand, in react years, more and more companies are shifting to cloud native architectures for improved efficiency, cost reduction, which in turn led to creation of new roles such as chief data officers (CDOs) and chief analytic analytics officer (CAOs) to channel the organizational BD capabilities towards fruition and competitive advantage.

Taking all into consideration, how can one embark on such rather sophisticated journey? what is best initial point for adopting BD ? what can be a good logical approach to address the ever-increasing complexity of BD systems? how can the organization develop a BD system that can effectively handle data of various workloads such as business analytics, real-time stream processing, bulk batch processing and machine learning ?

The initial design, development and deployment of a BD system is only the beginning of BD journey. As system grows larger, data providers and data consumers increase, data variety expands, data velocity extends, and metadata become increasingly more challenging to handle. This means, only a handful of hyper-specialized data engineers would be able to understand the system internal, resulting in silos, burnt out and potential miscommunication.

This creates a perfect ground for immaute architectural decisions that result in hard-to-main and hard-to-scale systems, and raise high-entry blockage. Since ad-hoc BD designs are undesirable and leaves many engineers and architects in the dark, novel architectures that are specifically tailored for BD are required. To contribute to this goal, we explore the notion of reference architectures (RAs) and presented a domain-driven distributed software RA for BD systems.

2 Why Reference Architecture?

To rationalize why we have chosen RAs as the suitable artefact, we first have to clarify two underlying assumptions:

- a sound software architecture is integral to successful development and maintenance of software systems
- there is a substantial body of knowledge in the field of software architecture to support the development of an effective RA

In essence, software architecture is an artefact that aims to satisfy business objectives through a software solution that is evolvable, const-efficient, maintainable, and scalable. In addition, it allows for capturing design issues when they are still cheap. Whereas this practice can be applied to any class of systems, it's particularly highlighted when it comes to design and development of complex system such as BD [3].

Despite the known complexity of BD systems, good news is that a RA can be developed, analyzed and designed to incorporate best practices, techniques and patterns that will support the achievement of BD undertakings. This can help engineers and architect better absorb complexity of BD system development and make it tractable.

This approach to system development is not new to practitioners of complex system. In software product line (SPL) development, RAs are utilized as generic artifacts that are instantiated and configured for a particular domain of systems [10]. In software engineering, IT giants like IBM have referred to RAs as the 'best of best practices' to address complex and unique system design challenges [8]. In other international standardization, RAs have been repeatedly used to standardize an emerging domain, a good example of this is BS ISO/IEC 18384-1 RA [] for service oriented architectures [14]. RAs are used for political speech and even NASA space data systems [19].

Based on the premises discussed above, RAs can be considered effective for addressing the complexity of BD system development for the following reasons:

1. RAs adhere to best practices, patterns, and standards
2. RAs can endow the architecture team with increased openness and interoperability, incorporating architectural patterns that ensue desirable predefined quality attributes
3. RAs can serve as the locus of communication, bringing various stakeholders together
4. RAs can be effective in identification and addressing of cross-cutting concerns
5. RAs can be some of the best approaches to complex system development such as BD, capturing design problems when they are still cheap
6. RAs can take the role of blueprint and summary in the portfolio of software architects and data engineers, resulting in better dissemination of knowledge
7. RAs can manifest the implicit knowledge of software architects as explicit actionable models

3 Research Methodology

There are several approaches to systematic development of RAs. In one effort, Cloutier et al [8] demonstrated a high-level model for development of RAs through collection of contemporary architectural patterns and advancements. Clouterier's approach is driven from the Stevens Institute of Technology forum for systems engineers and tend to explicate RAs from a system engineering point of view. In another effort, Bayer et al [5] introduced a method for creation of RAs

for product line development called PuLSE DSSA. This methodology is inspired by the works of Kazman et al [15] and in specific their architecture evaluation method; SAAM. Therefore it tends to pivot around scenarios for creation of candidate architectures which results in the intended RA.

Stricker et al [28] presented the idea of pattern based RA for service based systems and used patterns as first class citizens. This study, inspired by the works of Buschmann [7], and Gamma et al [12] portrays the RA as amalgamation of various patterns that are structured into different categories namely; high-level patterns, abstract patterns, and implementation patterns. This study is part of the NEXOF-RA [6] project.

Along the lines, Nakagaw et al [17] presented a four-step approach to design and development of RAs. This methodology benefits from an ecosystem of complementary artifacts that facilitate the process of RA development, such as the RAModel [18] (a reference model), and FERA [25] (a checklist for evaluation of RAs). In a more recent study, Derras et al [10] presented a four-phase approach for practical RA development in the context of domain engineering and software product line. This approach is influenced by ISO/IEC 26550 [29].

Nguyen et al utilized reverse engineering and reference model for creation of agent systems reference architecture [20], and Norta [21] followed a pattern based approach similar to that of Stricket et al. Galster and Avgeriou [11] proposed a 6 step methodology that is founded on two major concepts; 1) empirical foundation and 2) empirical validity.

Analysis and study of these methodologies for design and development of RAs have highlighted several recurring themes. While some of the these studies are older and some are more recent, observed commonalities are similar. All these methodologies share the same fundamental pillars that 1) RAs should capture the essence of best practices, 2) RAs can be built by absorbing current best practices, standards, architectures, RAs, and systems, 3) RAs should sit at the right level of abstraction, and 4) RAs should be communicated effectively.

Taking all these into consideration, we found 'Empirically-grounded reference architectures' methodology proposed by Galster and Avgeriou as the suitable methodology for the purposes of this study. This decision is to two factors; 1) this methodology is the most adopted approach for RA development, and 2) the focal elements surrounding the methodology is in-line with the objectives of this study.

Nevertheless, this methodology suffered from a few limitations and we had to augment several areas of it with other approaches in order to arrive at the desired rigour and relevance. For instance, we could not find comprehensive guidelines on how to collect empirical data in step 3 of the methodology. We did not know what kind of theory we should collect and how should we model the data collected. Lack of systematicity and transparency on this area, would undermine our overall approach, therefore we employed Nakagawa et al's information source investigation guidelines and the idea of RAModel [18].

Another area were we faced limitation was evaluation. The methodology did not provide detailed instructions on evaluation of the RA, thus, we looked

for a more systematic and stronger evaluation approach. For this purposes and inspired by the works of Angelov et al [2] citeangelov2014extending, we first instantiated a prototype of the RA in a real world practice and then used the 'Architecture Tradeoff Analysis Method (ATAM)'[16] to evaluate the artefact.

The methodology chosen for this study is constituent of 6 steps which are respectively; 1) Decision on type of the RA, 2) Design strategy, 3) Empirical acquisition of data, 4) Construction of the RA, 5) Enable RA with variability, 6) Evaluation of the RA. It is essential to clarify that the phrase 'empirically grounded' refers to two concepts; 1) RA should be based on well-established and proven principles, and 2) RAs should be evaluated for validity and applicability. These concepts do not only belong to the works of Galster and Avgeriou; other researchers such as Derras et al [10] and Cloutier et al [8] have promoted the same ideologies.

Lastly, it is worth mentioning that this research methodology is iterative, implying that the results yielded from the last step (evaluation) will determine the subsequent iteration until the artefact reaches saturation (research objectives).

Decision on type of the RA Precursor to development of the RA, it's effective to determine the type of it. The type is an important factor as it illuminates on the structure, the data to be collected and objective of the RA. To determine the type of the RA, we used the classification framework provided by Angelov et al [1] and the SLR conducted by Ataei et al [4].

References

1. Angelov, S., Grefen, P., Greefhorst, D.: A classification of software reference architectures: Analyzing their success and effectiveness. In: 2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture. pp. 141–150. IEEE (2009)
2. Angelov, S., Trienekens, J.J., Grefen, P.: Towards a method for the evaluation of reference architectures: Experiences from a case. In: European Conference on Software Architecture. pp. 225–240. Springer (2008)
3. Ataei, P., Litchfield, A.: Neomycelia: A software reference architecturefor big data systems. In: 2021 28th Asia-Pacific Software Engineering Conference (APSEC). pp. 452–462. IEEE (2021)
4. Ataei, P., Litchfield, A.T.: Big data reference architectures, a systematic literature review (2020)
5. Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., Widen, T., DeBaud, J.M.: Pulse: A methodology to develop software product lines. In: Proceedings of the 1999 symposium on Software reusability. pp. 122–131 (1999)
6. Birukou, A.: Nexof-ra
7. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-oriented system architecture: a system of patterns, chapter 2.3 (1996)
8. Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E., Bone, M.: The concept of reference architectures. Systems Engineering **13**(1), 14–27 (2010)
9. Computing, S.: Bridging the gap between data and business teams (2020), https://www.sigmacomputing.com/resources/data-language-barrier/

10. Derras, M., Deruelle, L., Douin, J.M., Levy, N., Losavio, F., Pollet, Y., Reiner, V.: Reference architecture design: A practical approach. In: ICSOFT. pp. 633–640
11. GALSTER, M., AVGERIOU, P.: Empirically-grounded reference architectures. Joint ACM
12. Gamma, E., Helm, R., Johnson, R., Johnson, R.E., Vlissides, J., et al.: Design patterns: elements of reusable object-oriented software. Pearson Deutschland GmbH (1995)
13. Huberty, M.: Awaiting the second big data revolution: from digital noise to value creation. Journal of Industry, Competition and Trade **15**(1), 35–47 (2015)
14. Iso, I.: Information technology — reference architecture for service oriented architecture (soa ra) — part 1: Terminology and concepts for soa. International Organization for Standardization p. 51 (2016), https://www.iso.org/standard/63104.html
15. Kazman, R., Bass, L., Abowd, G., Webb, M.: Saam: A method for analyzing the properties of software architectures. In: Proceedings of 16th International Conference on Software Engineering. pp. 81–90. IEEE (1994)
16. Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., Carriere, J.: The architecture tradeoff analysis method. In: Proceedings. Fourth IEEE International Conference on Engineering of Complex Computer Systems (Cat. No. 98EX193). pp. 68–78. IEEE
17. Nakagawa, E.Y., Guessi, M., Maldonado, J.C., Feitosa, D., Oquendo, F.: Consolidating a process for the design, representation, and evaluation of reference architectures. In: 2014 IEEE/IFIP Conference on Software Architecture. pp. 143–152. IEEE (2014)
18. Nakagawa, E.Y., Oquendo, F., Becker, M.: Ramodel: A reference model for reference architectures. In: 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture. pp. 297–301. IEEE (2012)
19. NASA: Reference architecture for space data systems (2008)
20. Nguyen, D.N., Usbeck, K., Mongan, W.M., Cannon, C.T., Lass, R.N., Salvage, J., Regli, W.C., Mayk, I., Urness, T.: A methodology for developing an agent systems reference architecture. In: International Workshop on Agent-Oriented Software Engineering. pp. 177–188. Springer (2010)
21. Norta, A., Grefen, P.: Developing a reference architecture for inter-organizational business collaboration setup systems. Beta, Research School for Operations Management and Logistics (2006)
22. Partners, N.: Big data and ai executive survey 2021 (2021), https://www.supplychain247.com/paper/bi_data_and_ai_executive_survey_2021/pragmadik
23. Rad, B.B., Ataei, P.: The big data ecosystem and its environs. International Journal of Computer Science and Network Security (IJCSNS) **17**(3), 38 (2017)
24. Rada, B.B., Ataeib, P., Khakbizc, Y., Akbarzadehd, N.: The hype of emerging technologies: Big data as a service (2017)
25. Santos, J.F.M., Guessi, M., Galster, M., Feitosa, D., Nakagawa, E.Y.: A checklist for evaluation of reference architectures of embedded systems (s). In: SEKE. vol. 13, pp. 1–4 (2013)
26. Sivarajah, U., Kamal, M.M., Irani, Z., Weerakkody, V.: Critical analysis of big data challenges and analytical methods. Journal of Business Research **70**, 263–286 (2017)
27. Stats, I.L.: Internet live stats (2019), https://www.internetlivestats.com/
28. Stricker, V., Lauenroth, K., Corte, P., Gittler, F., De Panfilis, S., Pohl, K.: Creating a reference architecture for service-based systems–a pattern-based approach. In: Towards the Future Internet, pp. 149–160. IOS Press (2010)

29. WG, I.: Iso/iec 26550: 2015-software and systems engineering–reference model for product line engineering and management. ISO/IEC, Tech. Rep (2015)
30. White, A.: Our top data and analytics predicts for 2019 (2019), https://blogs.gartner.com/andrew_white/2019/01/03/our-top-data-and-analytics-predicts-for-2019/