DOI: 10.2507/33rd.daaam.proceedings.xxx

WORKFLOW AUTOMATION PATTERNS IN NEXT GENERATION MICROSERVICES ENVIRONMENTS

Bernhard Angerer, Markus Stopper & Branko Katalinic

This Publication has to be referred as: Katalinic, B[ranko];Park, H[ong] S[eok] &Smith, M[ark] (2022). Title of Paper, Proceedings of the 33rd DAAAM International Symposium, p.xxx-xxxx, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-xx-x, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/33rd.daaam.proceedings.xxx

Abstract

This paper explores the requirements, design and implementation challenges of workflow automation in the light of modern microservices based software architectures. It thereby points to specific design patterns and approaches which function as a guidance or rule book to successfully overcome the challenge of mapping workflows to multiple microservices with its different sizes and granularities in terms of providing functionality and the ability of recombination. In this mapping the top down view looks at end-to-end business processes in a holistic way and meets the bottom up view in which domains need to be defined under technical aspects like transactional security and the construction of interfaces to facilitate the communication and coordination between the microservices.

Keywords: Microservices, Workflow Automation, Business Process Modelling, Bounded Context, Domain Driven Design.

1. Introduction

It is now state of the art to design and architect systems as an ensemble of services which run independently in virtual machines. These services then are glued together through a communication mechanism – typically through a message-bus which is provided by a separate server or through http/Rest communication endpoints which reside within the service. The objectives are better scalability, better modularity and maintainability as well as enhanced robustness of the overall system [1]. Concurrently the world of business process modelling (BPM) aims to deploy BPM engines which allow for a runtime execution of workflows thereby giving great flexibility in the world of workflow automation. Both worlds are now meeting each other for the first time and are supposed to converge and interact. A design approach is needed.

2. Description of the Problem

Up to this point the world of microservices is under intense debate and there is no general design approach existing which offers a reliable and incremental path of how to bring BPM engines and microservices together. In theory the creation of software through the combination and interaction of services creates a situation of flexibility because components had been identified which then can be mapped to the various “planes” (different abstraction levels) within the BPM flow charts. Since the business process modelling notation is standardized and is offering explicit graphical notations for specifying business processes it is perfectly complementary to the world of microservices. Moreover, it

could greatly utilize and help in the process of designing and operating microservices because it would bridge the world of process experts and software architects. In reality these two worlds speak very different languages on a human level as well as on a technical level.

The analysis, design and implementation of microservices are heavily determined by technical constraints. Transactional security and persistence in workflows or state-machines which encompass several services hosting in-process and out-of-process communication are instantly facing a variety of new challenges which do not exist in a single process or monolithic system. A classic situation is a flight booking scenario where subsequent business transactions (the booking of a hotel or rental car) need to be cancelled as well if the main transaction (the flight) gets cancelled. In these situations so-called "compensate actions" need to be implemented where "long-lived" transactions which spawn several services need to be rolled back [3]. Another challenge in the design of microservices but especially when linking microservices and BPM is functional decomposition. Software architect usually decompose the system by functions [4]. This works well for the implementation of technical components like an email or a printer service. In the world of business transactions and workflows this can be described as an anti-pattern. The different nature of (long living) business transaction will inevitably collide with technological need of a transaction creating a reliable dialog with a specific resource (e.g. a database or a device).

3. Suggested Solution

A generic design pattern must be established which enables the mapping and linking of BPM workflows with specific services provided by one or several microservices. Moreover, in order to achieve agility and flexibility this technique must inherently support an incremental process where refactoring and redesign are first class citizens. The following characteristics need to be met:

- Orchestration needs to be at the core of the microservices design
- "Manager" services as docking facilities into the BPM world
- Coarse granularity of the services
- Volatility based decomposition of the services

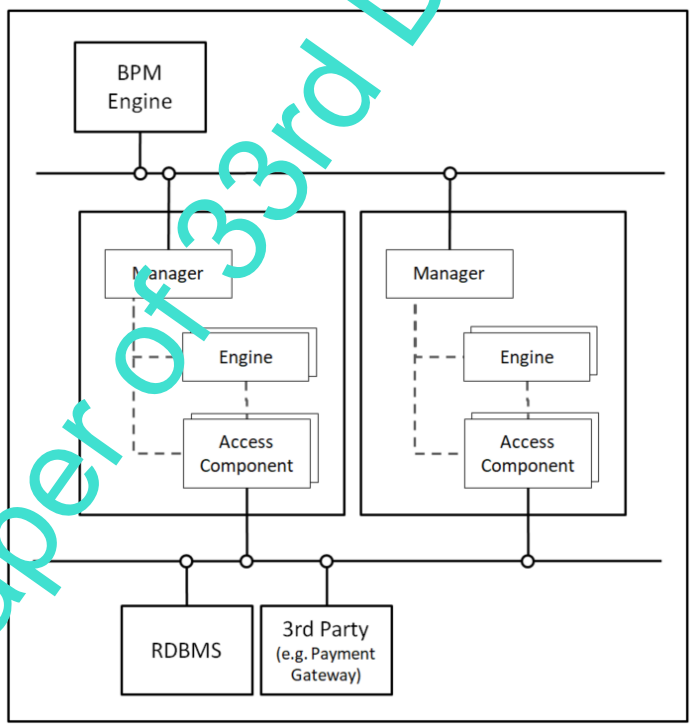

Fig. 1. The principle of the workflow automation pattern in microservices architectures

Orchestration aims to achieve automated configuration, management and coordination of computer systems, applications and services. In the proposed method orchestration needs to be implemented on a macro- (run-time orchestration through the BPM engine) as well as on a micro- (design-time orchestration due to implementation details of the microservice) level. Figure 1 shows the principle of this approach through the decomposition of a service into managers, engines and access components. Access components simply abstract a specific resource like a database or a 3rd party component. Engines contain re-useable business logic. Managers implement the needed functionality through orchestrating the functionality provided by one or many engines and access components. These three types of

components constituting a microservice reside within the same process and transactional context. The manager components are responsible of exposing the functionality of the microservice to the outer world through a network interface (e.g. a Rest API) hence they are the only "speaking partner" for the BPM engine. This overall approach leads to a coarse granularity of microservices and therefore greatly reduces complexity and works in favour of the system's communication topology [5].

The proposed design pattern also entails the decomposition method which functions as a guideline on which functionality should be wrapped into a manager (and therefore be contacted by the BPM engine) in the first place. Functional decomposition will collide with the business process context of the BPM workflows. The solution is the identification of these parts of the system which are most likely to undergo repeated change. The so-called "volatility based decomposition" identifies those domains in the system and tackles them with the manager-, engines-, access-component patterns as described above. It clusters all the areas of high volatility within the manager components. This way new functionality can be introduced more easily through design-time orchestration as well as run-time orchestration with the help of BPM.

4. Further Challenges

With microservices architectures the industry stepped into the world of distributed systems and thereby facing a whole new set of challenges when designing, implementing and operating enterprise systems. The decoupling of the semantics of distributed computing from those of the problem domain has many aspects and facets. In the context of the design patterns described in this paper the following points stand out in terms of further research which needs to be conducted:

- The boundary of BPMN's execution semantics
- Visibility into the manager, engine and access components interplay

With the version 2.0 the Business Process Model Notation (BPMN) introduced "execution semantics" to further describe the inner workings of BPM elements [6][7]. Alongside comes the capability to extend BPMN itself as well as extended capabilities of dealing with events. Further thoughts in this context are outlining a so-called Service Contract Implementation (SCI) Layer which consists of BPM planes for stateful and stateless integration processes [8]. This clearly overlaps with the domain of the manager components and the objective of volatility based decomposition. A clear distinction is needed which defines the boundaries of both worlds.

On the contrary there are no standards which could bring visibility and traceability into the design and operation of the manager- engine- and access-components. Currently the methods of object oriented programming with its SOLID principles are the only methods existing in terms of data-descriptive structures. On this aspect additional research is needed to facilitate the creation of correct domain models with its interplay to the BPM layer to achieve process automation in a flexible and sustainable way.

5. Conclusion

The interconnection of BPMN engines with microservice environments is postulating the next logical step in process automation. In particular the human machine interface provided by BPMN when developing and operating microservices is brought to the next level and enables new kinds of capabilities which are untapped so far. The key to this is a holistic course of action which revisits every layer in the technology stack. A pure top down approach like historically implied by the BPM community is not sufficient.

This paper presented design patterns which form a practical approach for the BPM as well as the middleware and microservices community to fruitfully work together. Those patterns are needed to bridge the gap between both worlds and future BPM engines as well as microservices environments may actively support those patterns. In return this will help both communities to fashion opinionated approaches in their design methodologies.

6. References

[1] Jamshidi P.; Pahl C.; Mendonça N.; Lewis J. & Tilkov S. (2018). Microservices: The Journey So Far and Challenges Ahead, IEEE Software, https://ieeexplore.ieee.org/abstract/document/8354433, DOI: 10.1109/MS.2018.2141039

[2] Rademacher F.; Sorgalla J. & Sachweh S. (2018). Challenges of Domain-Driven Microservice Design: A Model-Driven Perspective, IEEE Software, https://ieeexplore.ieee.org/document/8354426, DOI: 10.1109/MS.2018.2141028

[3] Stefanko M.; Chaloupka O. & Rossi B. (2019). The Saga Pattern in a Reactive Microservices Environment, Scitepress, https://www.scitepress.org/Papers/2019/79187/79187.pdf, DOI: 10.5220/0007918704830490

[4] Tyszberowicz S.; Heinrich R.; Liu B. & Liu Z. (2018). Identifying Microservices Using Functional Decomposition,

https://www.researchgate.net/publication/326901590_Identifying_Microservices_Using_Functional_Decomposition

[5] Beschastnikh I.; Liu P.; Xing A.; Wang P.; Brun Y. & Ernst M.; (2020). Visualizing Distributed System Executions, https://dl.acm.org/doi/abs/10.1145/3375633, https://doi.org/10.1145/3375633

[6] Lemrabet Y.; Clin D.; Bigand M. & Bourey P.; (2010). From BPMN 2.0 to the Setting-Up on an ESB – Application to an Interoperability Problem, Springer, Berlin, Heidelberg, https://doi.org/10.1007/978-3-642-15961-9_85

[7] Fdhila W.; Gall M.; Rinderle-Ma S.; Mangler J. & Indiono C. (2016). Classification and Formalization of Instance-Spanning Constraints in Process-Driven Applications, Lecture Notes in Computer Science(), vol 9850. Springer, Cham. https://doi.org/10.1007/978-3-319-45348-4_20

[8] Schäffer E.; Stiehl V.; Schwab P.; Mayr A.; Lierhammer J. & Franke J. (2021). Process-Driven Approach within the Engineering Domain by Combining Business Process Model and Notation (BPMN) with Process Engines, Procedia CIRP, https://www.sciencedirect.com/science/article/pii/S2212827121001037, https://doi.org/10.1016/j.procir.2021.01.076.