International Journal of Latest Research in Engineering and Technology (IJLRET)
ISSN: 2454-5031
www.ijlret.com || Volume 03 - Issue 07 || July 2017 || PP. 48-52

Transformer Platform for Business Process Management Automation

By Alex Osadchyy, D. Mgt and Yevgen Osadchyy, Ph.D
Taras Shevchenko National University of Kyiv, 2017

Abstract: A new design methodology for business process management (BPM) systems, named as Business Process Transformer Platform (BPTP), is introduced and applied to designing monitoring and management mechanisms in BPM. The new methodology argues thatby compacting a number of functions into a single modulethat has a simple analogy it is possible to develop complex software systems using the analysis and synthesis means of the simple systems.

The methodology provides a framework for simplified engineering of complex systems whilemaintaining the transparency of the system behavior through making context-specific representations of the simple model and maintaining those in multiple views.

The application of the BPTP methodology tothe design of a business process monitoring system, was studied. A series of case studies show that the dynamic behavior of the business process management system is represented by a combination of processes, resources, events, rules, activities, orchestration, modelling and analytics transformers, based on which the processes such as marketing, finance, accounting, and product/service delivery, can be designed quickly in one iteration and then enhanced through multiple revisions.

Keywords: programming model, complex systems, business process, transformer, analogy

Introduction

Recent decades demonstrated an increase in new software development methodologies and frameworks. They are fueled by the global competition were companies seek to faster deliver products to the market and increase their sophistication and value. For example, agilemethodologies provide advantages of shorter value delivery cycles, greater flexibility, less documentation overhead, and immediate customer feedback (Cao, Mohan, Xu, & Ramesh, 2009).Microservices approach to architecture enabled companies like Netflix to grow and scale their products effectively along different axes, and to take advantage of processes like continuous delivery (Thones, 2015). However, applying traditional decomposition approaches to components on complex products lead to creation of vast amount of components that complicate the interactions and make holistic understanding of the product difficult.

Creating new architectural methodology and introducing it is demanding. Complexity that can be easily imagined and maintained over years deems the known solution of a simple analogy. For example, amoebic organism analogy is used by companies to manage effectively large organization dynamics (Osadchyy, 2016).Referenceto principles of amoeba analogy systemcan enhance the effectiveness and efficiency of managing most challenging distributed organizations such as offshore software development. Architectural frameworks such as microservices, deal with complexity by encapsulating a unique process and a business goal in one component that communicates through a well-defined, lightweight interface. It simplifies the understanding of a system on a level of components, however when zooming out to a higher level, the big picture with numerous components and heterogeneous connections becomes blurry. Known from other industries multifacetedapproach would seem to help addressing that deficiency. Looking onto the system architecture from different perspectives will maintain the simplified understanding on both high and low levels.

Transformers is a science fiction movie that is gaining popularity as an analogy approach in designing in various industries. It’s application in software development remains undervalued. The ideology of building transformer intellectual toys is characterized by the fact that the child is offered a completeset of basic constructive elements, with the possibility of mutual displacement. With their complex functionality, toys have almost no small parts and there is no need to disassemble them completely to turn into something else. Their success lies in the fact that the idea of a simple transformation of the device image was first proposed and implemented. Other successful attempts have been done to create fully functional transformer devices (TD) in orthopedics, construction and sport, the basic structural elements of which are multi-functional modules (Osadchyy, 2004, Anisimov et al., 2016). Their main advantage is the possibility of multipurpose use. Soon it became evident that even simplest devices can have the ability to transform. Technologies that ensure the life cycle of TD, or the multipurpose use of any devices were called transformer (TT).TT is the results of applying extensive intellectual efforts in designing a device or a system, and are derivatives of human intellect. Every solution that puts priority on concentration of the information in a small volume justifies application of the designing efforts and creating transformers.

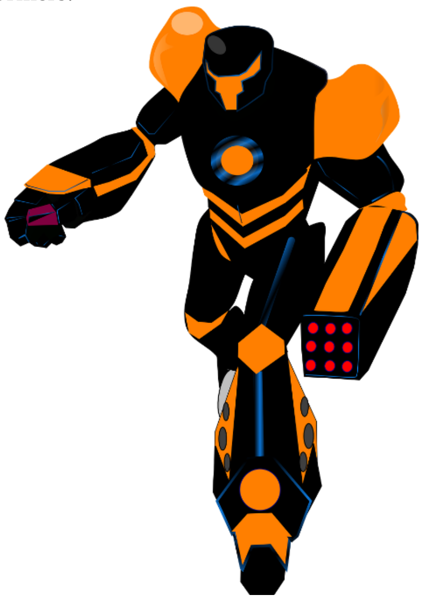

Figure 1. Transformer toy. Science fiction analogy of transformer technology.

As a software development technology, transformers are already applied in building data-parallel programming models. In development of distributed systems, existing programming models still require developers to master various APIs and deal with sophisticated details such as exception handling. Once developed for initial requirements and vision, these designs become monolithic and hard to extend in order to meet changing needs.Tu, et al. (2010) designed a framework that blends multiple parallel computation models such as Dryad-like data flow, MapReduce, and All-Pairs in one layer that rests on top of common runtime system and communicates via 2 simple primitives send() and receive(). Developers following the transformer architecture focus on separated high level views such as data partitioning, running multiple tasks, and fault recovery. While the framework takes care of concurrency, remote procedure calls, serialization, network programming, etc. It is extended via either modifying existing models or adding new ones. From the engineering point, each component such as controller, can be further optimized and cleaned. Transformer approach in Tu's et al. (2010) programming model is a multifaceted approach, where facets are represented by models.

Transformer Platform

A new design methodology for business process management (BPM) systems, named as Business Process Transformer Platform (BPTP), is introduced and applied to designing monitoring and management mechanisms in BPM. The new methodology argues that by compacting a number of functions into a single module that has a simple analogy it is possible to develop complex software systems using the analysis and synthesis means for the simple systems. The target environment is a typical enterprise that builds process-based applications and automates routine tasks. These mission critical applications need to be monitored, analyzed and managed to ensure continuous improvement and adaptation of the processes.

The methodology provides a framework for simplified engineering of complex systems while maintaining the transparency of the system behavior through making context-specific representations of the simple model and maintaining those in multiple views. BPTP platform consists of several layers depicted on Figure 2: business process management, container, interaction, transaction and package. Package layer represents solution of business products deployed in the enterprise. Companies often combine products such as ERP, CRM of different vendors, in-house developed, and outsourced systems. Transaction layer aims at cross-product state, session, transaction and data management. On top of it, are higher level interactions typically achieved by the enterprise bus such as routing the documents, transforming information, sending messages and connecting various services. Container layer is responsible for unifying lifecycle and accessibility of the components. Where components in modern architectures are services or microservices that are loosely coupled and serve single business purposes.

Business Process Transformer Platform (BPTP) rests on top of container and all other layers. It wraps and underlies hundreds and thousands of services and their connections to ensure switching of the modes and alignment with the view(facet) that is active. Each service is a multi-session component. Transformer platform allows to extend its flexibility to exist in different modes at the same time. Each mode could be the same service of different versions, service in intense monitoring and debugging mode, service in development and operation. Teams of business people, developers and IT, can work on the same services in different modes without interfering with each other. It is similar to container technology, where the same service is deployed to different container of development, staging or production environment. However in BPTP the separation happens on the level of services thanks to their transformers nature.

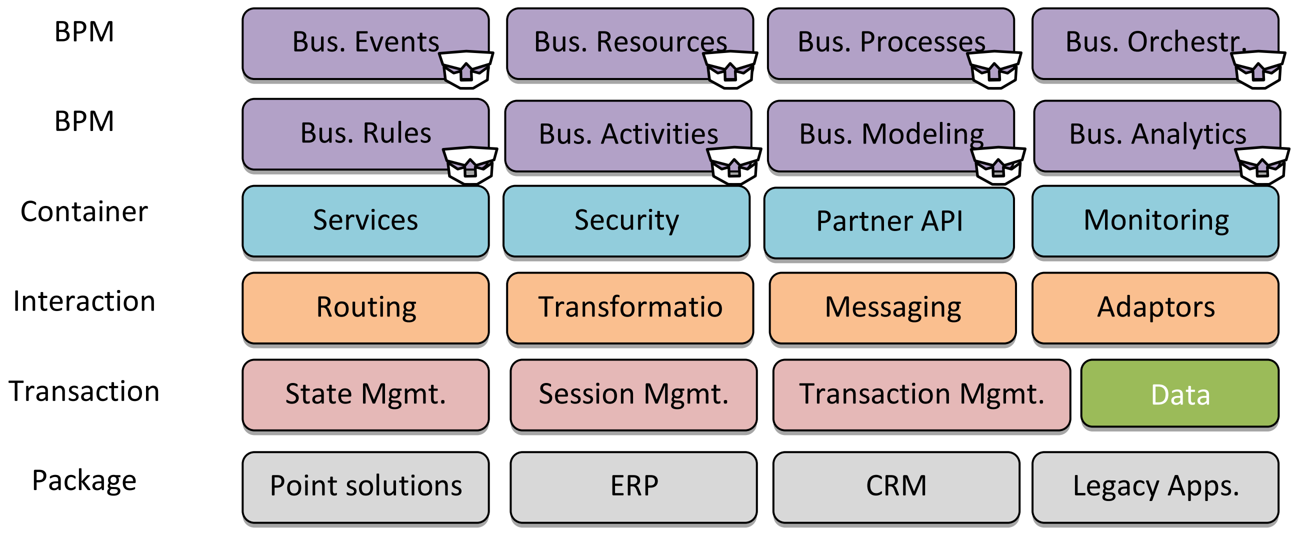

Figure 2. Enterprise Architecture. Service Middleware.

Case studies

The application of the BPTP methodology to the design of a business process monitoring system, was studied. A series of case studies show that the dynamic behavior of the business process management system is represented by a combination of processes, resources, events, rules, activities, orchestration, modelling and analytics transformers, based on which the processes such as marketing, finance, accounting, and product/service delivery, can be designed quickly in one iteration and then enhanced through multiple revisions.

Facet 1 in Figure 3 represents business process management in operation. A customer interacts with the system and creates a business event, such as places an order. Business events service which is running in operation mode responds to the order and interacts with business rules service, which is also in operation mode. Based on defined rules, business activity service is contacted to perform some activity, such as shipment. Other services of business resources, processes and orchestration interact according to their active mode in Facet 1.

In facet 2 of Figure 3 an IT person monitors and analysis business processes. All services in facet 2 are in active mode of monitoring. Business modelling service interacts with business processes and activities services to get their health status. Business analytics service contacts business orchestration service to collect statistics on transactions in order to perform analysis and display to the IT person on dashboard. Transformer platform takes care of the separation of services in different modes. As well as ensures they can co-exist in parallel and interact in their facet spaces without interference.

Finally, facet 3 of Figure 3 is a view of a developer who is investigating an issue in business process service. By running business analytics requests on the service, the developer indirectly involves business orchestration and processes services. They all are running in the isolates space of facet 3 and have corresponding transformations intact. The developer resolves the issue and replaces fixed business process service in face3. It is debugged and tested to verify the correct operation. When the work is done, facet 3 is deactivated. Next time business processes service is activated for the next facet, a new fixed version of the service will be used.

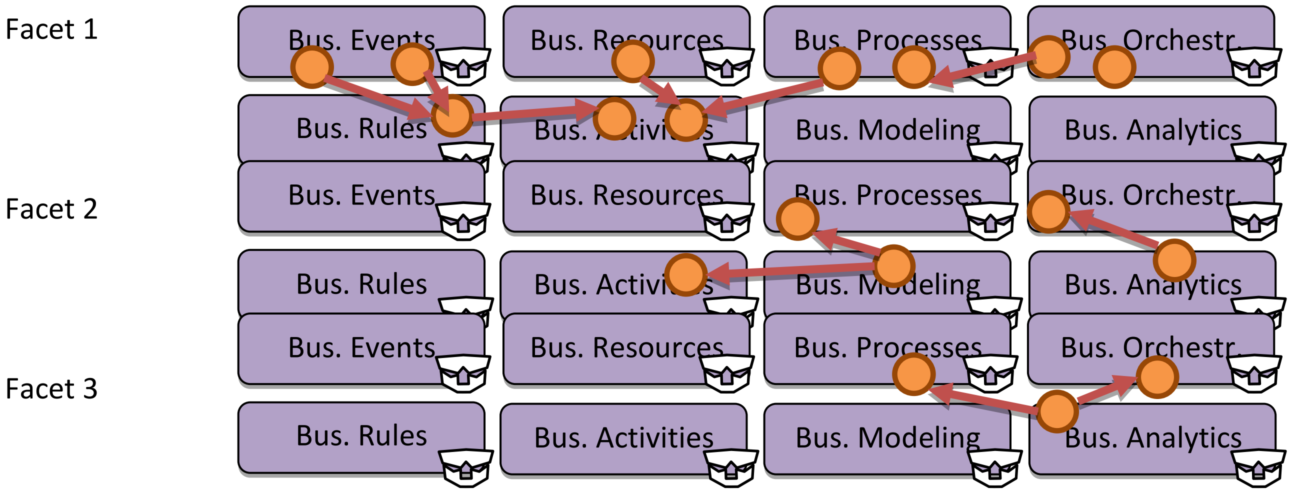

Figure 3. Transformers with activated facets. Facet 1 is in operation. Facet 2 is in development. Facet 3 is in monitoring.

Transformer platform technology is a generation away from microservices technology and several generations apart from the traditional monolithic design. Table 1 shows comparison of the three architectural approaches. Transformer technology gives a high-level view on the system via scenarios, such as facet for IT monitoring specific business processes. Microservices are useful to decompose the system into loosely coupled business components. However, level of abstraction is lower. Monolithic design consists of thousands of connected objects and is on the lowest level of abstraction among the 3. Transformer approach allows to focus on modules of multiple services at the same time thanks to grouping and compacting the functionality in one module. Multifaceted architecture and single environment are other advantages that simplify development, operation and maintenance of complex business process systems.

Table 1. Transformer vs Microservices.

	Transformer	Microservices	Monolithic
Level of abstraction	Scenarios	Components	Objects
Functional density	Many per module	One per component	Partial per object
Architecture	Multifaceted	Single	Lacking
Multiple versions and modes	Same environment	Separate environments	Separated in time

Conclusion

Business Process Transformer Platform (BPTP) is a concurrent design, development, operation and maintenance methodology that brings the challenging task of dealing with complex systems to the next level of efficiency. By increasing concentration of functions in a single module that has a simple analogy of transformers it is possible to develop complex software systems using the analysis and synthesis means of simple systems. Such modules can combine not only operational functions, but development and versioning via co-existence of the services in different modes and versions.

The application of the BPTP methodology to the design of a business process monitoring system, was studied. A series of case studies demonstrated that the dynamic behavior of the business process management system is represented by a combination of processes, resources, events, rules, activities, orchestration, modelling and analytics transformers, based on which the processes such as marketing, finance, accounting, and product/service delivery, can be designed quickly in one iteration and then enhanced through multiple revisions.

References

[1]. Anisimov, A., Osadchyy, Y., Gorbunov, O. (2016).Backbone exoskeleton and transformer technology to ensure it's lifecycle. Journal of Taras Shevchenko National University of Kyiv, pp. 39-40. Retrieved from http://dsr.univ.kiev.ua/projects/services/index.php?PAGEN_1=2

[2]. Cao, L., Mohan, K., Xu, P., & Ramesh, B. (2009). A framework for adapting agile development methodologies. *European Journal of Information Systems, 18*(4), 332-343. doi:10.1057/ejis.2009.26

[3]. Krumeich, J., Weis, B., Werth, D., & Loos, P. (2014). Event-driven business process management: Where are we now? Business Process Management Journal, 20(4), 615-633. Retrieved from https://search.proquest.com/docview/1658146435?accountid=35812

[4]. Osadchyy, A. (2016). Software Outsourcing Beyond Traditional Benefits: Grounded Theory Study. ISBN-10: 3659896926. Germany: LAMBERT Academic Publishing.

[5]. Osadchyy, Y. (2004).Transformer technologies of building machines and mechanisms. Science World Journal, pp 167.

[6]. Ragusila, V. (2016). Mechatronics by analogy and application to legged locomotion (Order No. 10140567). Available from ProQuest Dissertations & Theses Global. (1821420266). Retrieved from https://search.proquest.com/docview/1821420266?accountid=458

[7]. Thones,J.(2015). Microservices, IEEE Software, 32(1), pp. 116. doi:10.1109/MS.2015.11

[8]. Transformer toy image. Free commercial use. Retrieved from https://pixabay.com/en/robot-black-orange-transformer-296534/

[9]. Tu, B. et al. (2010). Transformer: A New Paradigm for Building Data-Parallel Programming Models, IEEE Micro, vol. 30, no. , pp. 55-64, doi:10.1109/MM.2010.75