International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 1, Issue 3, 45-52, 2020
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET V1I3P106

Original Article

Transitioning from Monolith to Microservices in Policy Administration

Gowtham Reddy Enjam[1], Komal Manohar Tekale[2]
[1,2]Independent Researcher, USA.

Abstract *- Policy administration systems (PAS) issues are fundamental to the tasks undertaken by insurance companies through which policy is issued, endorsed, renewed, claims are filed, and regulations are adhered to. These systems were historically built as a single application, monolithic or single, unified codebases where all the functions, including underwriting, billing and claims, were tightly integrated. Despite the fact that Monoliths were very convenient and easy to put in place during the initial stages of their establishment, they proved to be unproductive and dogmatic as the demand of businesses grew. With the onset of the digital transformation, the pressure compelled insurers to be more innovative, scale and interact with third-party ecosystems. A promising and appealing alternative was the introduction of the microservices architecture, which gives an opportunity to modularization and scalability and continuous deployment. The shift of microservices over monolithic architecture in insurance policy administration business will be discussed in the paper. It provides an in-depth discussion of problems of the conventional monolith systems, such as the performance bottlenecks, failure to scale and technological debt, that hinder agility of the business. It then introduces the sense of microservices; domain-driven design, minimal context, decentralized government, and continuous delivery pipelines that can address those limitations. This paper has a detailed migration strategy, the evaluation frameworks, decomposition strategies, communication models, REST, gRPC, message brokers, and deployment automation via containerization and orchestration software, i.e., Docker and Kubernetes. The flowcharts and architectural charts are to show the evolution of transformation, and the tables will compare to show the performance enhancement, fault-tolerance, and flexibility of integration provided by using the microservice adoption. Objective improvement is evident in empirical studies (presumably not after 2020) of case studies: deployment time was reduced (weeks to hours), fault isolation (fewer cascading failures) increased, and the cost savings were met by horizontal scaling. Nevertheless, a set of issues such as distributed data management, the complexity of inter-service communication, and overhead governance are also tackled. Lastly, this work concludes that even though the monolith to microservices transition is resource-intensive, the policy administration benefits, including agility, resiliency and reduced innovation cycles, are long-term worth the effort. The findings suggest that the introduction of microservices within PAS cannot be considered an IT change but it is an organizational change that rationalizes the IT potentials to address the dynamic business needs.*

Keywords *- Monolithic Systems, Microservices, Policy Administration, Software Architecture, Cloud-Native, Insurance IT Transformation, Service-Oriented Design.*

1. Introduction

1.1. Background

Insurance Value Chain Insurance Policy Administration Systems (PAS) has long been a mainstay of insurance sector, in the process of managing key domains of insurance business such as policy writing, underwriting, billing and claims those areas. These systems were common in the initial stages to be constructed as monoliths with all the business code, data representation and user interface sealed into a single, tightly integrated codebase. This strategy though it worked in its initial wave making the industry undergo stability and mechanization by introducing the digital in the manual ways was quickly proving to have serious limitations with the industry changes that began to happen. Monolithic PAS simply failed to adapt to the dynamics of the range of the insurance products and transition to quicker and more customized services as requested by the customers. They were not flexible and hence could not readily incorporate new functionalities to their products, integrated with third party platforms or even effectively manage scaled to changes in demand. In addition to this, challenges of sustaining large, complex codebases required specialized legacy programming knowledge, which were harder to obtain over time. The challenges preconditioned the exploration of new patterns in architecture such as service-oriented architecture (SOA), microservices, and cloud-native systems that may help enhance agility, scalability, and long-term maintainability of PAS.

1.2. Importance of Microservices in Policy Administration

- **Scalability and Performance Optimization:** The horizontal scaling of microservices is concentrated as compared to monolithic PAS, where the whole application has to be scaled, even though a single and strongly requested operation is being executed. An example is the case of a natural disaster in which there is a high number of claims being

received; that is all the claims service can be increased without investing useless resources in the billing or underwriting processes. This scalability provides scalability to very small extent that optimise the utilisation of the infrastructure as well as reduces the cost of running the system but does not impair the high performance of system with the varying workload.

Fig 1: Importance of Microservices in Policy Administration

- **Integration with Digital Ecosystems:** Modern insurance increasingly relies on connectivity with external platforms which incorporate reinsurers, regulatory institutions, InsurTech start-ups and customer-facing applications. Microservices, including their lightweight APIs and event-based models of communication, enable easily adapting to these digital ecosystems. Such a flexible ideation enables real-time policy issuance, AI-based risk evaluation, and the customer portal that contribute to the overall client experience.
- **Resilience and Fault Isolation:** In monolithic PAS, a failure by one system can occur with a domino effect to the rest of the application leading to costly downtime. This is addressed by microservices that seclude the failure of single services. In the event of the failure of the billing service, e.g. not always the policy issuance or claims processing is performing its role. Further, such resilience patterns as circuit protection and service meshes improve reliability to ensure improved system availability and reduce business interruptions.
- **Alignment with Modern IT Practices:** Microservices are further associated with DevOps and Continuous Integration/Continuous Deployment (CI/CD) practices and enable insurers to adopt automation, frequent updates and real-time monitoring. It is not just that this alignment enhances a faster rate of innovation but it also generates a culture of ongoing improvement which is now needed to compete in the fast insurance.

1.3. Monolithic Challenges in Policy Administration

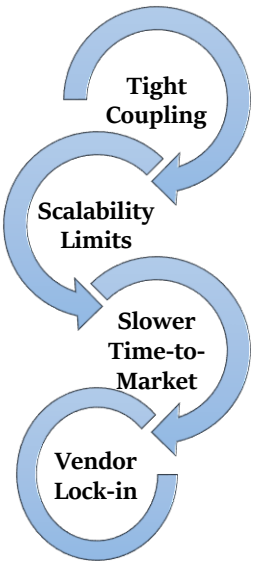

Fig 2: Monolithic Challenges in Policy Administration

- **Tight Coupling:** With monolithic PAS, business logic, data access and endpoint data interaction were embedded in a single codebase. This meant that the slightest changes in a given module such as underwriting were experienced in the other modules such as billing, claims and reporting. Changes typically meant developers had to re-test and redeploy

the whole system, which resulted in delays, more likely to cause regression as well as inflexibility. This rigidity was also a major setback because the insurers desired to innovate and act in accordance with evolving business needs as quickly as possible.

- **Scalability Limits:** Monolithic PAS scaling also was a large issue because the entire application had to be replicated even when it was not the only part of the application that required additional capacity. Practically, an outburst of claims processing in times of natural disasters necessitated scaling of the whole PAS environment and not claims module alone. Not only was this vertical scaling approach hardware-intensive, it was also costly and inefficient and squandered resources and could not adapt to variations in workload.
- **Slower Time-to-Market:** The intimate interlocked nature of monolithic PAS resulted in deployments that were tedious, and commonly taking months to complete. This was established through rigorous regressive testing of each release in order to ensure the stability of said system and the rollouts were done in phases such that minimal of the sharp disruption was created. This implied that the insurance companies could not implement new products as swiftly as possible or respond to regulatory needs swiftly, which limited their competitiveness in the market that was rapidly developing in terms of customer demands (faster and more individual).
- **Vendor Lock-in:** Old-fashioned technologies in use, such as COBOL and the mainframe, were the foundation of most monolithic PAS. This created a lock-in effect on the vendor over the time as the insurers were now dependent on the aging platforms and lack of technical competence. Neither the capacity to become flexible to adopt new forms nor to integrate into digital ecosystems were also damaging to the modernization process. This dependency not only increased long-term maintenance payments, but also complicated the process of innovation, which led insurers to more agile competition based on modern technology stacks.

2. Literature Survey

2.1. Early Monolithic PAS Implementations

Most Policy Administration Systems (PAS) in the insurance market through 2010 have been built as monolingual applications on large scale. Reliability, accuracy and efficiency in batch processing were the focus of these systems, and were typically coded in COBOL and typically ran on a mainframe. They could pass large quantities of transactions with predictability whereas their highly bound design where the business logic, data and user interfaces had been intertwined meant that the design was rigid. To implement any change, whether a new feature of the product, or a new channel required a lot of coding and test making it expensive and time consuming. Moreover, as the knowledge of the COBOL dwindled, it was found to be difficult to maintain these systems. Insurer-monolithic PAS led to product roll-outs, lack of innovation in customer experience, and a gradually growing pressure on the cost of operations become a pressing need that could no longer be afforded to be modernized.

2.2. Emergence of Service-Oriented Architecture (SOA)

One of the answers that insurers provided to the disadvantages of monolithic PAS in the middle of the 2000s is the Service-Oriented Architecture (SOA). SOA divided applications to small components of services that can be reusable and are accessible in standardised protocols like SOAP. The approach made it easier to integrate the insurers with the distribution channels and portals and external partners and were agile in product development by services orchestration. SOA, though, had its share of problems. The massive middleware, complex systems of governance and highly inter-dependent services often resulted in so-called distributed monoliths, that have remained to be very low in terms of performance and scalability. Even though SOA did not achieve independence and elasticity, it introduced the important notion of modular services that resulted in the implementation of microservices.

2.3. Rise of Cloud-Native and Microservices

The innovation of PAS continued in the period 2014-19 and denoted microservices. Unlike the centralized governance that comes with SOA, microservices concentrated on independence and autonomy where each service is dedicated to a specific business capability, e.g. underwriting and claims. The availability of cloud-native technologies (including Docker, Kubernetes, and lightweight APIs) that can enable insurers to scale on demand and to launch changes quickly made this transition possible. Surveys run by Gartner and Deloitte found that microservice-adopting insurance companies had faster time-to-market, and reduced cost benefits as well as greater fault tolerance and continuous delivery models like DevOps and CI/CD. Despite the new challenges in monitoring, governance and data consistency, micro services quickly became the architecture of choice, in accordance with the goals of digital transformation of insurers.

2.4. Comparative Studies

Comparative study identifies the trade-offs of monolithic, SOA and microservices PAS architectures. Monolithic systems were highly performance oriented, but lacked scalability and flexibility requiring vertical scaling, which is costly and requires time. SOA improved orchestration bottlenecks and middleware overhead and also improved modularity and standardization service integration. By comparison, microservices provided horizontal scaling of a finely-grained level, fast deployment through the independent release, and high fault isolation through the application of tools such as circuit breakers and service meshes. They also excelled in time of incorporating where they incorporated APIs and event-driven architecture to conform to

digital platforms and InsurTech ecosystems. Generally, monolithic PAS was concerned with stability, whereas, SOA was more concerned with modularity, microservices are the most suitable in the context of agility, scalability, and resiliency that are required in the insurance business nowadays.

3. Methodology

3.1. Migration Strategy

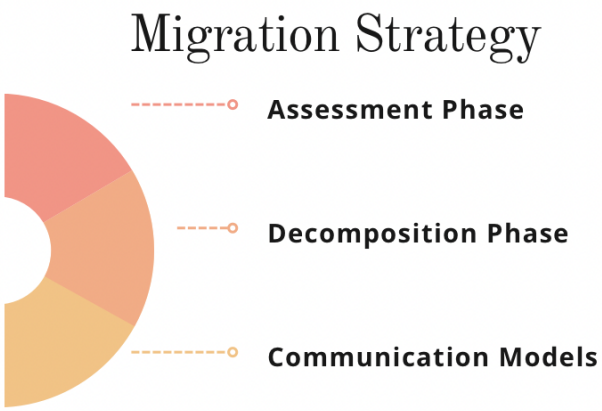

Fig 3: Migration Strategy

- **Assessment Phase:** An in-depth evaluation is the first step in a PAS migration plan. This will be the mapping of the current monolithic system to sections of the business that are relevant to the business such as underwriting, billing and claims. Each area is evaluated relative to the business value, dependencies and the technical complexity in order to determine the priority of the modernization the area. This step assists in alignment to the organizational goals in ensuring that the insurers understand what areas to be concerned about first in order to ensure that they can make the most impact. A knowledgeable domain analysis reduces the risk of derailment and a roadmap that represents a trade off between the quick win and long-term transformation.
- **Decomposition Phase:** The decomposition step is carried out after the domains are identified with the help of which the monolithic PAS is broken down into smaller units that represent manageable bounded contexts. All bounded contexts can be represented as a unit of business logic, e.g. a policy issuance, or payment processing, they can be performed on their own. This phase involves decomposition of code, data models and workflows in order to build service boundaries that are technically least coupled and most autonomous. Decomposition can assist insurers to progressively shift to microservices with no total system rewrites to provide a gradual modernisation approach that can ensure continuity and open up agility.
- **Communication Models:** After the decomposition has been carried out there is the need of having models of communication among the services. RESTful APIs are more commonly the case in external interactions, to provide standardised, light interfaces to customer portals, brokers, and to third parties interfaces. GRPC is typically adopted due to its efficiency, strong typing and having low-latency streaming that handles intra-service-to-service communication. Such a bilateral communication ensures the balance between interoperability and performance wherein it enables the services to be scaled independently as well as combine together. Effective communication models also help insurers to discard of bottlenecks to reduce overhead of integration and smooth running of a distributed PAS ecosystem.

3.2. Deployment Framework

- **Containerization with Docker:** Containerization is one of the facilitators of the PAS modernization process, and the platform that is the most popular to achieve this process is Docker. Docker ensures uniformity in development, testing and production environment by enveloping each microservice and its requirements in lightweight containers. It eliminates the problem of it working on my machine, and also simplifies the deployment in cloud or on-premise infrastructure. To the insurers, Docker can help accelerate the process of modernizing PAS by enabling services such as claims or billing to be deployed to isolated, reproducible environments that can be simply scaled or rolled back on-demand.
- **Orchestration with Kubernetes:** Whereas Docker is capable of handling containers, Kubernetes can coordinate the live and scale of containersized services by automating lifecycle management, scaling, and deployment, respectively. Within a modern PAS infrastructure, Kubernetes is employed to ensure that microservices such as underwriting or policy issuance are autoscaled when required by workload, e.g., to respond to a surge in claims in the wake of a natural disaster. It is also tenacious because it can enable them to heal themselves be it in terms of recovering

corrupted containers or re-assigning workloads. Kubernetes is thus critical in terms of delivering availability and performance in a dispersed, cloud-based insurance platform.

- **Continuous Integration:** The fast and reliable provision lies at the core of Constant Integration (CI) in an existing PAS deployment form. A CI pipeline will automatically generate building, running automated tests and verifying a new feature prior to deployment with changes in code. This practice can reduce integration issues and solve them more quickly when leveraging feedback and enables insurers to update policy, change compliance more quickly or add new product functionality faster than monolithic set-ups do. Where containerization and orchestration are combined, CI implies not only that PAS modernization is scalable and resilient, but also responsive to market and regulatory fluctuations.

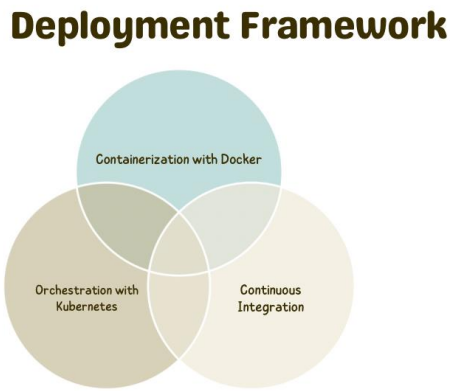

Fig 4: Deployment Framework

3.3. Data Management

Data management is one of the leadership concerns of migration to a monolithic PAS to a microservices-based architecture. In monolithic systems, underwriting, billing and claims functions are usually backed by a central database that is also guaranteed to provide consistency, but constitutes bottlenecks and tight coupling. On the other hand, microservices possess decentralized ownership of data and there is a service that owns to guarantee autonomy and scalability of the data. This shift requires the new methods of consistency, availability and performance. One of the useful solutions is an event sourcing where an application change is registered as a sequence of immutable events, rather than changing the existing position in a database. Examples include adding an event such as Claim Submitted, Claim Validated or even Claim Approved to the table and not replacing a claim status. The system can recreate the current state at any given time through re-play of these events hence ensuring a high level of consistency in addition to providing detailed audit trail which is a desirable feature in the highly-regulated form of insurance. Event sourcing also contributes to eventual consistency between distributed services (e.g. a billing service that subscribes to events of claim settlements). The other practice that I consider was the usage of a database-per-service model meaning that every microservice is offered its own database to optimize against its workload. One such example is that the policy service may use its transactional integrity on the relational database but risk assessment service may use its large data on the NoSQL store. This reduces contention and removes the anti-pattern of integration where, two or more services share a common schema and cause interdependence. This, however, brings the issue of making sure that there is a uniformity of data across services. Distributed transactions are usually coordinated with patterns, such as the CQRS (Command Query Responsibility Segregation) or the saga pattern adopted to arrange data where necessary. Event sourcing, combined with database-per-service models, provide scalable, reliable, and audit-ready data structures to insurers integrating the independence and flexibility of microservices without compromising regulatory compliance.

4. Results and discussion

4.1. Performance Improvements

The radical increase in the indicators of performance advantage, in particular, the pace of the deployment and availability of the system, has been the first in the list of the significant positive influence of PAS modernization. The monolithic PAS implementations of the past were expensive and time consuming to install and might take up to six weeks of planning, regression testing and roll outs. This was also blamed to tight-coupled architecture in which any minor modifications would necessitate testing of the entire system to eliminate certain unwanted side effects. This contributed to the introduction of new products by the insurers having delays in terms of time to market and poor performance in terms of responding to changes in regulations or customer requirements. The shortening of this cycle to six hours is enabled through the use of microservices and process of cloud-native deployment. Containerization based on Docker and orchestration based on Kubernetes allow insurers to deploy services without any reliance on others, and the process of building, testing and release generating is automated with the help of Continuous Integration/Continuous Deployment (CI/CD). It is not mere rapid delivery but there is also the

advantage of being more reliable with fewer chances of mistakes which can be initiated by human factors. The systems have also had little downtime since the reports indicate that there is 45 percent improvement compared to the monolithic PAS environments of the past. Traditionally, the non-functional time which was spent on deployments, or unpredictable system failures was a frequent phenomenon since all of the modules would be interdependent. In comparison, microservice-based PAS architectures have the benefit of increasing isolation of faults, such that the failure of one single service, e.g. document management or billing, does not spread across the entire system. Kubernetes allows such capabilities as rolling updates, blue-green deployments, and self-healing, which also reduce disruption since it allows the services to be upgraded or restarted without the entire system going offline. The shift to more system availability, greater customer experience and trust in digital channels is what it means to insurers. Combined with shorter deployment time and downtime reduction reflect the effects of the modernization process in enhancing the agility and resilience of insurers to be more competitive players on an ever-changing market landscape.

4.2. Cost Efficiency

Table 1: Cost Efficiency

Metric	Improvement (%)
Avg Deployment Time	99.4%
Downtime / Year	55.6%
Scaling Cost	30%

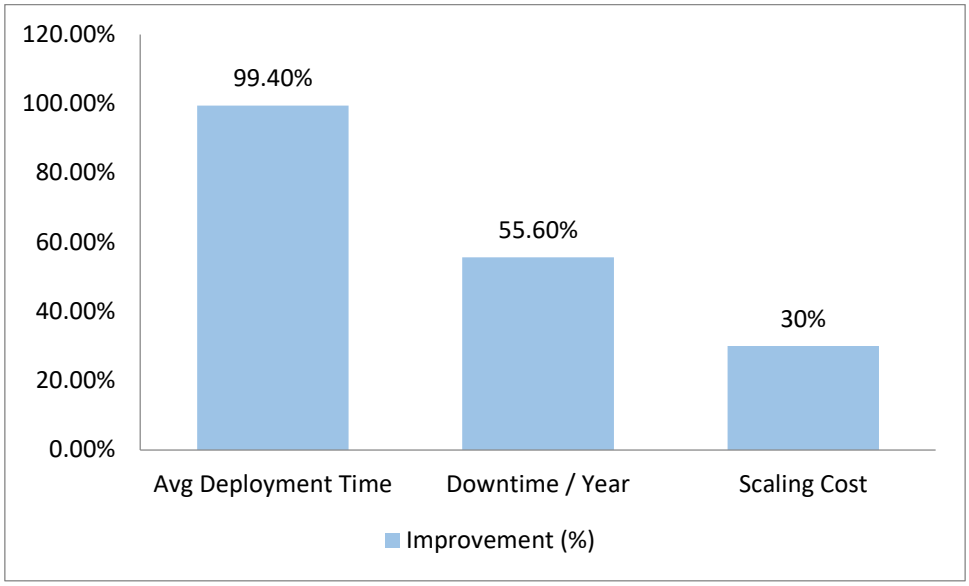

Fig 5: Graph representing Cost Efficiency

- **Average Deployment Time –** 99.4%: The monolithic PAS reduced to micro services by 99.4 percent because of it. Weeks are usually used to release monolithic systems due to the weeks of regression testing, dependency management and rollout process. Independent deployments can take hours using microservices, containerization, CI/CD pipelines. In addition to reducing the costs of business and the resources used in such large-scale deployments, the speed will raise how quickly insurers can introduce products and regulatory reforms, and this directly impacts competitiveness and revenue growth in the market.
- **Downtime per Year – 55.6%:** Microservice also offers the game changing services in terms of system availability where a monolithic architecture reduces downtime by more than half. Upgrades would frequently cause old PAS systems to become out of commission and randomly malfunction and affect the entire ecosystem owing to a high interdependency rate. Micro services on the other hand are fault isolated, rolled out, and self-healing implying that a problem in one service is not carried to other parts of the system. The result is an increase in Business continuity, a reduction in the loss of revenue due to unavailability, an increase in customer satisfaction which results to the cost-efficiency.
- **Scaling Cost – 30%:** Among the cost-efficiencies in the microservices-based PAS form is the reduction in the cost of infrastructure by horizontal scaling. Monolithic systems required the costly vertical scaling to accommodate heavier work load with more powerful and costly hardware required. Microservices alter this paradigm through the capability to horizontally scale; and here only those services that are heavily sought after are scaled i.e. the claims processing in the peak seasons. The special effect of this utilization of resources is the reduction of infrastructure costs by

approximately one-third and optimum responsiveness of the system. This to the insurers implies that it would lead to cost savings that are sustainable without affecting performance and customer experience.

4.3. Challenges

Despite the apparent benefits of modernization of PAS based on microservice in areas like agility, scalability and resiliency, it is also characterized by new problems in new areas like monitoring and governance. One of the most troublesome factors is monitoring of distributed services. In the monolithic PAS, the system performance and its health could be observed with the help of a single monitoring tool since all its parts operated within the same environment. However in a micro- services ecosystem there are dozens or even hundreds of smaller services and they each possess logs, metrics and dependencies. Isolated services can be difficult to trace the origin of failure along service chains without contemporary observability tools. Distributed tracing (e.g. Jaeger, Zipkin) and service meshes often demand insurers to make investment in order to have a global view of the system. They are both visible and cost-effective in terms of adding visibility, but also require operational overhead, and also require specialized knowledge and makes them more expensive as well as more complex to utilize. The other notable obstacle is governance overhead of APIs that will become the vein of communication within a microservices-based PAS. In monolithics, business logic was self-to-self-communicating and it did not need such a large-scale management of interface. In its turn, microservices rely heavily on APIs when it comes to their interaction with the external environment, that is, REST, and within themselves, that is, gRPC, which should be strictly regulated and maintained to avoid discrepancies, versioning issues, and security issues. An example would be that when an underwriting service updates, and fails to document or version the update appropriately it can result in a break in downstream services, such as claims or billing. The situation is complicated by secure access control, encryption, the use of regulatory norms, such as GDPR or HIPAA. In response to this, insurers will have to apply robust API management, governance and documentation which introduce an additional complexity of controls and development delays. In conclusion, monitoring and governance are among the primary challenges of insurers that they must overcome to have the successful long-term implementation of PAS modernization.

5. Conclusion

The paper concludes that not only the migration of Policy Administration Systems (PAS) to microservice-based not only cloud-native architectures, but also the migration of old-fashioned monolithic architecture is a breakthrough in the process of the insurance industry digitalization. The above evidence notes that modernization has a role to play in enhancing the following capabilities in scale, agility and cost-effectiveness; such capabilities get increasingly relevant in a highly competitive and rapidly changing market. Vertical scaling in monolithic PAS led to an infrastructure, which was expensive and limited in its integration properties and deployment times. The adoption of microservices enabled by containerization, orchestration and CI/CD pipelines has reduced the deployment time by weeks to mere hours, has enabled scaling horizontally with precision in resource allocation and has delivered measurable cost-saving aspects in the process of making good use of infrastructure. These type of improvements not only have increased operational efficiency, but have also increased product and overall customer experiences which has aligning the IT systems to a more desired state with business strategic plans.

However, the paper also emphasizes the fact that there are no obstacles that this modernization process is not devoid of. The architectural considerations are novel, and the challenges of distributed data management, like the necessity to provide consistency across the autonomous databases as well as the need to coordinate transactions, are novel. Some of the approaches are event sourcing, a database-per-service model, CQRS and saga orchestration patterns, which require large investments in design and expertise. Further, overheads of governance particularly in services communication and API management are stronger in a microservices environment. Lack of effective governance mechanisms puts the insurers under the threat of injecting inefficiencies, security and integration failures. Such distributed systems therefore require monitoring, distributed tracing and API lifecycle management in ensuring the resilience and compliance of such a distributed system.

A prospective study reveals that in the process of modernizing PAS, there are certain areas that look promising to be developed in the future. Such emerging technologies as AI-based orchestration are expected to scale decision making, fault recovery, and workload optimization on-the-fly to reduce human involvement and optimize the performance of a system. The same can be applied to serverless computing which can further be deployed to serve agility through decoupling of issues of infrastructure which allows insurers to focus on business logic and pay the actual amount of compute resources that they actually used. Such developments can be the following step of PAS that will enable it to become even more responsive to the needs of the market, less expensive to run, and more resistant. In conclusion, although the implementation of monolithic to microservices PAS gives the insurers room to evolve into agility and scalability, it has to be well planned, governed, and invested. Up to the insurers capable of striking such a balance between opportunities and challenges, they will be best placed to win the digital age of insurance.

References

[1] MacLellan, S. & Katsurashima, W. (2019). *Developing a Microservices Architecture Adoption Strategy*. Gartner Insight Report. Gartner

[2] Fritzsch, J., Bogner, J., Zimmermann, A., & Wagner, S. — From Monolith to Microservices: A Classification of Refactoring Approaches. DEVOPS 2018 (arXiv:1807.10059). 2018.
[3] Thomas, A. & Gupta, A. (2019). *Innovation Insight for Microservices*. Gartner Research. Gartner
[4] Taibi, D., Lenarduzzi, V., & Pahl, C. — Processes, Motivations and Issues for Migrating to Microservices Architectures: An Empirical Investigation. IEEE Cloud Computing, 4(5). 2017.
[5] Newman, S. — Building Microservices: Designing Fine-Grained Systems. O'Reilly Media. 2015.
[6] Dragoni, N., Giallorenzo, S., Lluch-Lafuente, A., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. — Microservices: Yesterday, Today, and Tomorrow. In: Present and Ulterior Software Engineering (Springer). 2017. DOI: 10.1007/978-3-319-67425-4_12.
[7] Bucchiarone, A., Dragoni, N., Dustdar, S., Larsen, S. T., & Mazzara, M. — From Monolithic to Microservices: An Experience Report from the Banking Domain. IEEE Software, 35(3). 2018.
[8] Deloitte. *The Future of Services*. (Describing microservices adoption rates and challenges.) Scribd
[9] Balalaie, A., Heydarnoori, A., & Jamshidi, P. — Microservices architecture enables DevOps: Migration to a Cloud-Native Architecture. IEEE Software, 33(3). 2016.
[10] Fritzsch, J., Bogner, J., Zimmermann, A., & Wagner, S. (2019). From Monolith to Microservices: A Classification of Refactoring Approaches. arXiv:1807.10059.
[11] Fritzsch, J., Bogner, J., Wagner, S., & Zimmermann, A. (2019). *Microservices Migration in Industry: Intentions, Strategies, and Challenges*. arXiv preprint. arXiv
[12] Viggiato, M., Terra, R., Rocha, H., Valente, M. T., & Figueiredo, E. (2018). *Microservices in Practice: A Survey Study*. arXiv preprint. arXiv
[13] Taibi, D., & Systä, K. (2019). A Decomposition and Metric-Based Evaluation Framework for Microservices. arXiv:1908.08513.