Microservice Architecture for Insurers

Solutions for building flexible and nimble IT Architecture using microservices

Abstract

Many insurers are going through a “digital transformation” phase and, as a result, have similar competing challenges — reacting to the ever-changing demands of the market, maintaining a customer-first focus, and ensuring the integrity and consistency of all systems which are either built across many years or are gathered as part of mergers and acquisitions over time.

Insurance IT systems are unique and interwoven in nature, and present a common set of challenges and impediments. This paper will investigate many of these challenges and provide solutions based on real-world examples.

The key tenants of the solutions presented herein, are to remain pragmatic, given the insurer’s current technology investments and budgetary constraints, to reduce system complexity and corresponding maintenance efforts, and to avoid common pitfalls and anti-patterns that crop up in microservices solutions.

Author

Sabyasachi Chowdhury, *Principle Architect Technology with Cognizant’s Insurance vertical, has vast experience architecting and implementation large legacy transformation initiatives for insurers across the globe. Have spent considerable amount to time solving current architecture impediments and business process roadblocks to help Insurers march towards a more digital focused ecosystem along with maintaining their proven and stable legacy IT landscape.*

Sabyasachi.Chowdhury@cognizant.com

Thanks to Satish Venkatesan (Leader, Enterprise Architecture group – Insurance and Retirement Services) and Gene Loparco (Chief Architect – Insurance) for valuable inputs and contribution to this paper.

Traditionally, insurance carriers have provided marginally better service to their insured's and agents by simply adding features to existing products, hone premium pricing or by streamlining claims processes.

But there is a fundamental shift towards deploying systems of intelligence to automate, improving technology systems by adopting cloud technologies along with an inherent need for customers, agents and employees to be able to interact with them across different channels and devices, when and wherever from they want to.

It is evident that a state of art customer experience is not only a function of a human centric front end layer, but also on the responsiveness of the core layers of the architecture, availability of backend systems, consistency of data and agility within the organization as a whole.

Insurers generally tend to lag behind other industries when it comes to any of these and adopting newer technologies and current digital go-to-market processes due to risks associated with early adoption and also because of an innate tendency to accrue technology debt. Most of the Insurance companies are buried deeply under large legacy systems (monoliths) and manual processes mostly due to inter woven business processes and mergers and acquisitions.

Insurance enterprise impediments

Insurance is a highly regulated and process-oriented industry with its unique series of challenges. The business requirements involve multiple actors and are mostly interwoven, which also increases the need of constant data flow in every direction.

Another key challenge with the Insurance IT landscape is the constantly changing rules governing these processes, which possess significant challenges to automation. Most of the core insurance IT systems and solutions are designed as monoliths to enable handling of all the dimensions and aspects of a complex business process. Most of these monoliths are either home grown or are commercial off the shelf products implemented over years.

In order to achieve all that, the insurers face significant impediments even today and some of the most common ones are highlighted below.

IT landscape of monoliths and heterogeneous systems

- ✓ policy and claims systems evolved over years and shaped up as monoliths, because business requirements involved multiple actors and are mostly interwoven, which also increases the need of constant data flow in every direction.
- ✓ the rules that govern business processes are constantly changing, resulting in significant automation challenges

- ✓ complex business processes and lifecycle management spanning across multiple domains caused systems to sit as close as possible, adding up to the size and complexity of these systems
- ✓ batch oriented systems depending highly upon data feeds
- ✓ mergers and acquisitions over years have led to accumulation of technical debt and heterogeneous systems landscape.

Inconsistent user experience

- ✓ strong LOB alignment and ownership leading to implementation of singular executable or applications leading to LOB centric portals
- ✓ FNOL, 3rd Party Claimants & Injured Workers interact over phone
- ✓ only adjuster can reach to claimants via phone

Data remains an application asset

- ✓ insurance companies have always been data centric, but always lacked the sophistication of converting raw data into marketable value
- ✓ lack of data-driven innovation, pervasive data governance and data integration initiatives like MDM, ODS, RDM or DataHubs.

Low Availability

- ✓ legacy business services causing too much dependence on other system availability
- ✓ low availability due to traditional long maintenance cycles and lack of continuous delivery mechanism.
- ✓ lack of cloud adoption and dependence on infrastructure which are on premise.

Lack of right grained services and customer centricity

- ✓ services were mostly focused around what the system can provide rather than what the customer wants.
- ✓ services were mostly relegated to software professionals seeking to resolve interface and data sharing problems associated with incompatible software systems.
- ✓ lack of API strategy and API centric architecture and platforms

Lack of Agility

- ✓ lack of enterprise agility towards adoption of newer technologies and tools, technology evaluation and adoption takes months.

- ✓ agile and DevOps practices are mostly targeted towards small to medium initiatives in silos, primarily towards mobile and web based developments. However, they typically depend on core back-end systems.
- ✓ practices like test driven development and continuous Integration are not popular due to challenges posed by demands for changing requirements and to speed to market.
- ✓ large costly release cycles spanning over weeks - due to ever-increasing demand towards speed to market and changes to underlining rules and regulations, most of the changes are often collected over time and distributed through infrequent software release cycles.

At the same time most of the Insurer are rich in data and have developed core competencies around data, but majority of the data are buried under legacy system of records or data warehouses for reporting purposes. Insurers do perform some amount of analytical and predictive analysis on the data but that is more driven towards pricing, risk management, providing a better claims experience and improved product features.

Solution patterns

It is difficult to find a direct correlation between these solution options and any of the impediments listed above, mostly because they complement each other and cannot be fully achieved or implemented without the existence of the other.

Migrate towards a Microservices based architecture

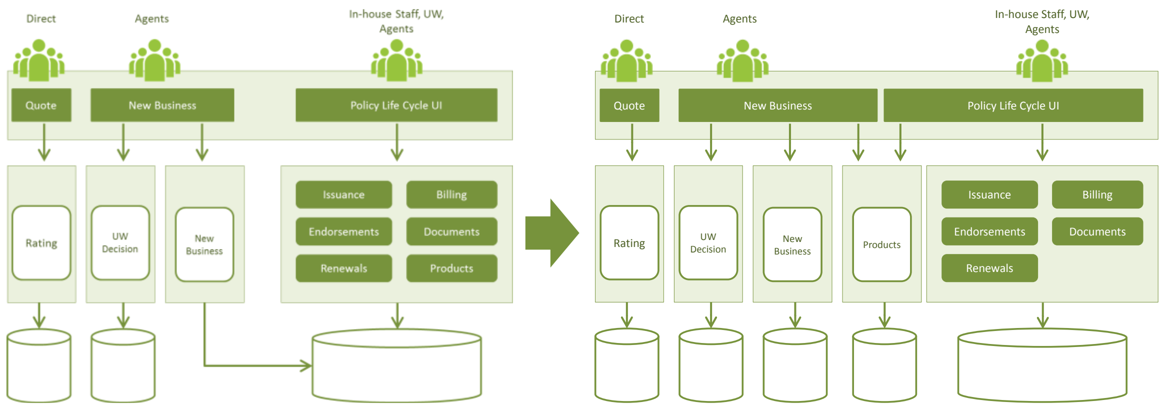
Conceptually, microservices don't differ much from the Service Oriented Architecture (SOA) approach commonly used within the insurance industry; the objective remains the same: to decouple portions of the larger application into cohesive, individual modules, capable of being deployed and distributed as separate applications wherein each component can be maintained independently.

James Lewis and Martin Fowler describe microservices as "an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery"

Most importantly, microservices are all about the notion of a "bounded context" and a "share-nothing" architecture, where each service and its corresponding data are compartmentalized and completely independent from all other services, exposing only a well-defined API. This bounded context is what allows for quick and easy development, testing, and deployment with minimal dependencies. This is the core philosophy of DDD – Domain Driven Design and is the most important factor for defining and curving out microservices.

A good example for that in the context of an insurer is the rating service, because they depend on just rates and rules and have minimum to no dependency on any external or parallel service, because of the fact that the rating service can be defined within a bounded context, it becomes one of the prime candidates across insurers for a first step towards microservices adoption.

In a real life scenario, migration from a monolithic application to microservices bases architecture should be done in a progressive manner by adopting the functionality first and data last approach as depicted in the diagram below -



As depicted in the diagram above, microservices can be curved out within a bounded context with their own set of data store as well as may still connect back to the core system as an interim solution. This helps insurers to migrate their legacy monolithic applications to a microservices based architecture in a progressive manner and at the same time contain the legacy monolith application and turn it hollow over time.

Development of a microservices based application is always a decentralized effort, allowing smaller teams (famously referred to as “2 pizza team” – a team which can be fed by 2 pizzas) internal or otherwise – to oversee individual segments of the application through their development lifecycle and through release. This helps development of applications around singular business processes rather than overwhelmingly large development efforts.

Leveraging the existing capabilities and be part of the API Economy

The term API economy refers to the opportunities associated with productizing the exposure of your business functions mostly developed as microservices through APIs. Consider that your API is a consumable product, and an insurer should market and position their products correctly for maximum profit.

Traditionally, APIs allow different software applications to communicate and offer services to one another. Earlier, software products would expose highly technical services to each other that wouldn’t make sense to businesspeople. As API technology became more standardized and software applications evolved to work with each other across the internet, insurers began using APIs to offer business services in software form. This was perhaps the biggest leap forward in creating the API economy.

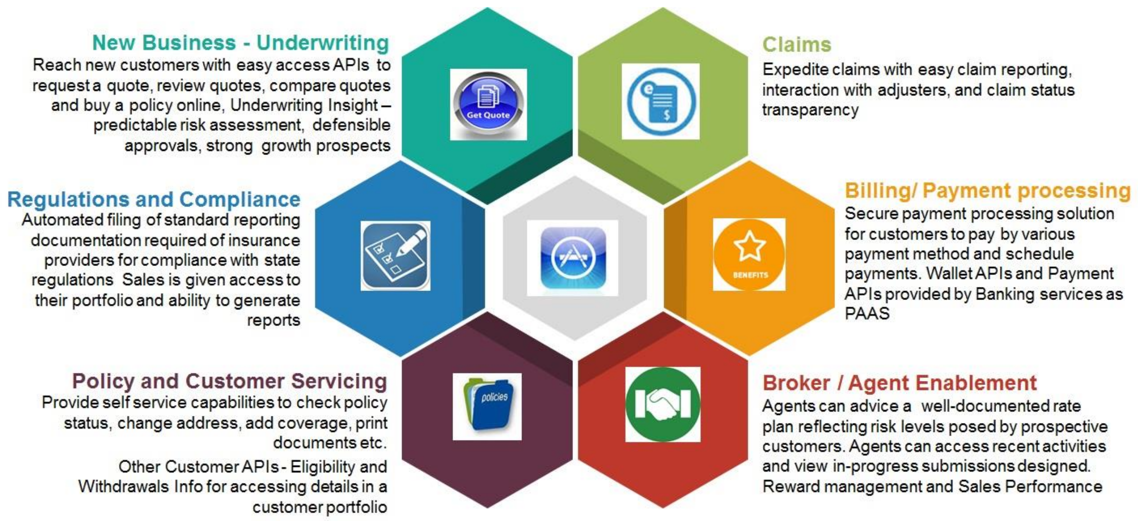
For insurers embarking on the digital transformation journey it is imperative to offer a richer set of data-driven, customer-facing services, enabled by a customer-centric approach of doing business or risk of being marginalized. Below are some of the key areas an insurer should focus to define the overall digital and API strategy -

- an omni channel enabled buying journey, that augments traditional channels with robust self-service options, direct purchasing and a single customer experience across online, mobile and now social channels.
- ability to leverage data and analytics across the entire value chain, including product innovation, marketing and sales, new business, servicing, claims, and operations
- APIs to make it easy to do business with partners(providing APIs to partners allows insures to offer insurance options to their customers directly)
- Get quote or rate APIs exposed for 3rd party aggregators
- First notice of loss (FNOL) automation and self-service options for claims and claims status
- Ability to directly interact with the claim adjuster and service professionals though chats and workflows.
- Get quote or rate for external comparative rater integrations like PL Rater and EZLynx

"According to Gartner, APIs make it easier to integrate and connect people, places, systems, data, things and algorithms, create new user experiences, share data and information, authenticate people and things, enable transactions and algorithms, leverage third-party algorithms, and create new product/services and business models"

Walgreens is a good example. Like many retailers, the company provides a photo printing service that takes digital photos and turns them into physical photographs.
https://developer.walgreens.com/

Whereas below are some of the core business areas and their potential to either expose of consume APIs within the insurance lifecycle -


New Business - Underwriting
Reach new customers with easy access APIs to request a quote, review quotes, compare quotes and buy a policy online, Underwriting Insight – predictable risk assessment, defensible approvals, strong growth prospects
Get Quote
Claims
Expedite claims with easy claim reporting, interaction with adjusters, and claim status transparency
Regulations and Compliance
Automated filing of standard reporting documentation required of insurance providers for compliance with state regulations Sales is given access to their portfolio and ability to generate reports
Billing/ Payment processing
Secure payment processing solution for customers to pay by various payment method and schedule payments. Wallet APIs and Payment APIs provided by Banking services as PAAS
BENEFITS
Policy and Customer Servicing
Provide self service capabilities to check policy status, change address, add coverage, print documents etc.
Other Customer APIs - Eligibility and Withdrawals Info for accessing details in a customer portfolio
policies
Broker / Agent Enablement
Agents can advice a well-documented rate plan reflecting risk levels posed by prospective customers. Agents can access recent activities and view in-progress submissions designed. Reward management and Sales Performance

APIs have a deep design relationship with microservices, in nutshell the business APIs exposed are developed as microservices to provide scalability, responsiveness, management of deployment, monitoring and most importantly communication between every other microservice in the architecture as well as with the applications and Web sites they power and the databases from which they draw real-time information, essential to their functioning.

Agility towards changes and new opportunities through the adoption of CI/CD and PaaS

Most Insurers are already performing Continuous Integration (CI) in some form or the other, those adopting CI have further leveraged agile ideologies to use microservices as a driver to achieve more frequent software releases, which has led to the practice of continuous delivery (CD). CD uses a quality focused ideology to build potentially shippable product increments, which speeds the deployment pipeline, achieving a result that culminates in bringing changes to production as quickly as possible. CD is a business function and too many frequent releases to production may cause disruptions and confusion, hence many Insurers still follow a stringent process of approvals and gatekeeping for any CD pipeline.

Microservices are an attractive CI/CD pattern because of their enablement of speed to market. With each microservice being developed, deployed and run independently microservices allow organizations to “divide and conquer”, and scale teams and applications more efficiently.

Many Insurers are adopting agile and CI/CD in targeted areas such as mobile and web. Those areas, however, typically depend on core back-end systems for data and business logic. If those systems are not also nimble then the agility of the business will always remain limited. Insurers must therefore bring Agile and CI/CD to all their platforms— including the mainframe—if that’s where their core data and business logic reside.

Data as the new differentiator

Insurers have traditionally competed with product features, however in today’s customer focused world there is a need of easy to comprehend products for better customer experience. The right analytics on data would lead an Insurer to think about the right products, go to market strategy and better customer experience. Data plays a central role in those value propositions.

Source of such data can be from the Internet of Things like weather sensors on the ground, telematic devices in vehicles, networked sensors in commercial buildings and homes, etc. And beyond IoT social media behaviors generate new types of data that can be used both in real time and for historical risk-rating purposes. Some of these sources can be tapped into directly or are also made available through aggregators.

Insurers need sophisticated technology to convert these data to meaningful information and use of technologies like Big Data and Analytics and/or artificial intelligence. These sophisticated techniques allow insurers to discover otherwise hidden patterns, trends and correlations across complex data sets.

Key Considerations

As part of the digital transformation journey, it is imperative to focus on improved customer engagement models, innovations, automation (like 0 touch and state through processing) and improved decision making by leveraging data, but the scope should not only be restricted to digital front ends alone, instead should also be towards building an efficient and nimbler IT ecosystem/architecture from the backend up to the system of engagement layer.

The right level of abstraction from core legacy system by the use of microservices based architecture, domain driven design principles and an agile operating model supported by a data centric architecture is the recipe to get nimbler and flexible.

But when it comes to the core insurance systems or system of record

- most policy administration and claims management systems have evolved over years and have shaped up as monoliths, because the business requirements involved multiple actors and are mostly interwoven, which also increases the need of constant data flow in every direction.
- complex business processes and lifecycle management spanning across multiple domains caused systems to sit as close as possible, adding up to the size and complexity of these systems.

But there has always been a need for speed to market and frequent changes due to changing rules and regulations, which inherently mandated the need for flexibility to move changes to production at a faster pace and frequency.

However, like SOA, companies developing microservices are finding themselves struggling with mainly defining the approach, where to start, service granularity, data migration challenges, organizational and cultural change, and distributed processing challenges. Hence, understanding the future needs, current business drivers, overall organizational structure and technology environment is important before jumping on to the bandwagon due to some of the antipatterns and pitfalls associated with this architecture pattern.

Some of the key antipatterns and pitfalls that are encountered with the insurance ecosystem due to the level of complexity within the current landscape are –

Level of Service Granularity (“Grains of Sand” Pitfall)

Choosing the right level of granularity for your services is critical to the success of any microservices effort. Service granularity can impact performance, robustness, reliability, change control, testability, and even deployment.

For example, if you have 3 services each for add, update and delete customer, it may be justified to club them to a single service called maintain customer.

Data migration antipatterns

When insurer plan to migrate from a monolithic application to microservices architecture the first goal is to split the functionality of the monolithic application into small, single-purpose services and the second goal is to then migrate the monolithic data into small databases (or separate schemas) owned by each service.

Most of the insurers believe that data should be a corporate and not an application asset. Given that, you can appreciate the risk involved and the concerns crop up with continually migrating data. Data migrations are complex and error-prone—much more so than source code migrations. Understanding the risks involved with data migration and the importance of "data over functionality" is the first step in avoiding this antipattern.

Example, insurers are embarking onto the journey of microservices by carefully selecting use cases, which can be broken into smaller more independent components like rating and rules components. But when it comes to breaking a monolith like a claims processing system or for that matter a policy administration system, it is advisable to follow the path of functionality first and data last.

Reporting and data consistency antipatten

With microservices the services and corresponding data are contained within a single bounded context. How to obtain reporting data in a timely manner and still maintain the bounded context between the service and its data is the challenge.

Remember, the bounded context within microservices includes the service and its corresponding data, and it is critical to maintain it.

At the same time maintaining data consistency across these systems not only for the purpose of reporting but also for supporting end customer queries and needs is also of upmost importance.

The REST pitfall

REST is by far the most popular choice for exposing APIs and accessing microservices and communicating between microservices. The REST pitfall is about using REST as the only communication protocol and ignoring the power of messaging to enhance your microservices architecture. With messaging and asynchronous requests the service caller does not need to wait for a response from the service when making a request, referred to as "fire-and-forget" processing.

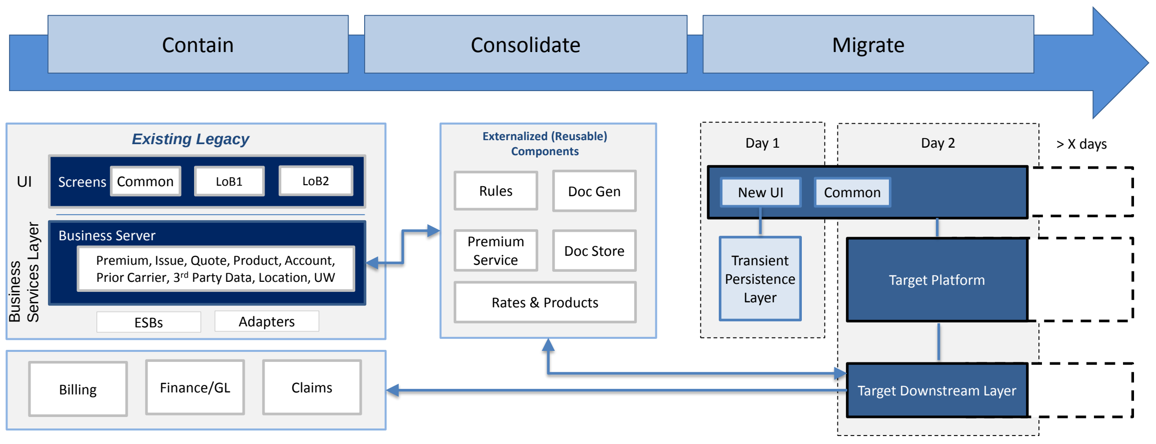
Examples can be submission of a final and accepted quote, FNOL, policy submission and also for communication between microservices like the rating and policy admin system, policy and payment system etc.

Adding to the maintenance nightmare

Microservices is about creating lots of small, distributed single-purpose services each owning their own data. As these services supports the notion of bounded context and share nothing architecture, where each service is compartmentalized and completely independent, hence adding to maintenance overhead.

Approach

Marching towards a Microservices based architecture helps an Insurer move away from some of the existing impediments and challenges with current systems and processes, but a methodical and pragmatic approach is what is most needed.



Some of the key considerations based on current challenges, solution options and antipatterns are listed below -

- Adapt to the principles of Domain Driven Design, and patterns like the CQRS, event sourcing and bounded context.
- Identify the right candidates to be built as Microservices and move to cloud
  - What architecture characteristics are most important?
  - Can it function as an independent component within a bounded context?
  - Does that component have direct dependency with other components or do they share persistence?
  - Do these components change frequently?
  - Is there a need to scale?
  - What are the primary business drivers?
- Assemble a "modern stack" from existing capabilities (cloud first, scale, working software, CI/CD, eliminate existing technical debt)
- Opportunistically modernize existing capabilities and incrementally move to the new stack
- Build new components as microservices for the capabilities that are deemed non-go-forward, which means contain existing legacy applications and make changes or enhancements to them only if not possible as Microservices.
- Expose the right grained APIs as products and independent of consumers' implementations or in other words build them generic and self-contained to the extent possible.
  - Start with more coarse-grained and then split it further if needed.
- Reuse existing services if they already meet performance SLAs and are already right grained; wrap them as REST APIs if needed, use REST for all new services.
- Microservices are not an alternate to SOA, continue investing in SOA and build upon it. Which means continue to build on the SOA patterns and encourage the culture of externalizing and reuse within the enterprise.
  - Examples of rules and rating components which can be externalized and built as microservices.
- Components which are dependent on external data feeds and requires critical processing should be treated asynchronously like batches, also move to cloud as that will help with elasticity.
- Design you applications and Microservices independent of any native cloud platform following the 12 factor app principles- this will help migrate to cloud and also reduce vendor locking from a cloud perspective.
- Initiatives like ODS, MBM and RDM should continue, if implemented on a RDBMS database, layer them with the likes of a MongoDB for faster queries by flattening the data
  - but balance between data availability and too many data migration between systems and databases

    - data migrations are error prone and complex and will become more complex over time.
    - treat data as a corporate and not an application asset
- Continuous Integration (CI) is a must have, build and invest in an efficient CI pipeline and framework. Build the ability to perform Continuous Delivery (CD), CD being a business function should be controlled as per the business need.

Some of the ready candidates to be moved to microservices based architecture could be

- premium generation or rating services
- FNOL and APIs for claims statuses
- product engines
- rule based services for rating, new business or underwriting
- or for that matter the whole Quote or New Business component

Summary

There is an inherent need for the Insurers to do some course correction which includes moving away from the current monolithic systems or contain the existing monolithic apps to get them hollow, simplify existing convoluted business process, use batch oriented systems only when necessary and move away from a culture of traditional waterfall based project execution and releases.

In order to achieve all that it is important to start small and experiment, instead of going with a mindset of breaking a monolith to smaller apps, identify the right candidate which are self-contained and migrate progressively.

An overarching enterprise strategy and vison supported by architecture guidelines and principles towards microservices based architecture, APIs and data centricity is essential. Tt is also important to welcome the right level of changes to be more agile and reduce the size of the failures.