**Microservice Architecture for Insurers**

Solutions for building a 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, whether they have been built over many years, or just recently acquired.

Insurance IT systems are unique 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 microservice 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*](mailto: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.*

In recent years, many insurance carriers have been content with adding features to existing products, honing premium pricing, and streamlining claims processes. While these efforts have served a purpose, they have typically resulted in nominal user experience and convenience improvements for both customers and agents.

In today’s world, differentiation and customer satisfaction are driven by the user experience. The user experience does not refer to a single channel, such as the web, as is typically the case. Instead an optimal user experience is defined as part of an omni-channel solution. Such a solution is built upon cloud technologies and systems of intelligence.

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 in adopting new technologies and processes, due to the associated risks and an aversion to accruing too much technology debt. As a result, most insurance companies are stuck with large legacy systems (monoliths) and manual processes. Interwoven business processes, and mergers/acquisitions only exacerbate the problem.

**The Challenges**

Insurance is a highly regulated and process-oriented industry with a unique set of challenges:

**Current Landscape of Monoliths and Heterogeneous Systems**

* business requirements involve multiple actors and are highly interwoven, resulting data flowing in every direction
* the rules that govern business processes are constantly changing, resulting in significant automation challenges
* most of the core insurance IT systems and solutions are designed as monoliths to enable handling of all dimensions and aspects of a complex business process
* mergers and acquisitions over years have led to accumulation of technical debt and heterogeneous systems landscape
* monoliths can be both home grown and commercial off the shelf products, implemented over years
* batch oriented systems depending highly upon data feeds

**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.

Most insurers have rich sets of data that they have developed core competencies around. But much of that data is buried in legacy Systems of Record (SoR) or in data warehouses used for reporting. Insurers do perform some amount of analytical and predictive analysis on their data but it is often driven by pricing, risk management, improved claims experiences, or improved product features.

Now, in order to enhance the user experience, insurers must provide customers with useful information, opportunities and solutions. To do so, systems must support the flow of data to and from multiple channels, using different formats, and must use new tools like Big Data to accept, manage and analyze that data.

A course correction is necessary in order to stay competitive. Insurers must contain, or move away from, monolithic systems, simplify business processes, use batch-oriented systems only when necessary, and move away from a culture of traditional waterfall-based project management and releases.

*Conveys Law: "Any organization that designs a system will inevitably produce a design whose structure is a copy of the organization's communication structure."*

**The Solutions**

These course-corrected goals can be achieved by using the solutions that follow. Solutions are grouped into four major areas of focus: Microservices, APIs, Continuous Integration and Deployment, and Intelligent Data.

**Adopting 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”*

Microservices are distinguished by their adherence to a “bounded context”, and a “share nothing” architecture. Each microservice and its corresponding data are compartmentalized and completely independent from each other. This bounded context is what allows for quick and easy development, testing, and scaling.

In the insurance industry, a prime candidate for a microservice is a rating service. Such a service performs a specific function and has no external dependencies.

Migrating from a monolithic application to a microservice-based architecture can be done in a progressive manner by following a “functionality-first, data-last” approach, as depicted in the diagram below.



Development of a microservice-based application is always a decentralized effort, which allows small teams to oversee individual segments of the application from development through release. This approach fits in well with an agile development process. It promotes the building of the application as individual components, based on business processes, rather than as a single, large development effort, wherein responsibilities are not clearly delimited.

**Leveraging 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.

*According to* [*Gartner*](https://www.gartner.com/smarterwithgartner/welcome-to-the-api-economy/)*, “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”*

Traditionally, APIs allow different software applications to communicate and offer services to one another. In the past, 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 rich set of data-driven, customer-centric services. Following are some of the key areas an insurer should focus on, to define an 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 social channels.
* the 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 (i.e., providing APIs to partners allows insures to offer insurance options to their customers directly)
* “get quote” or “rate” APIs exposed for 3rd party aggregators, or for external comparative rater integrations like PL Rater and EZLynx
* 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

*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/*](https://developer.walgreens.com/)

The aforementioned key areas cut across all core business areas. Each area has the potential to either expose of consume APIs within the insurance lifecycle. The following diagram depicts how each core business area may utilize the API economy.



APIs have a deep design relationship with microservices. Business APIs are deigned with omni-channel consumers (clients) in mind. The information sent or received helps define the microservices, and allows the clients to interact with interfaces that are optimized for delivery of data that meets their specific needs.

**Impact of Microservices on CI/CD**

Most insurers are already performing Continuous Integration (CI) in some form or another. Some insurers have taken automation further by implementing Continuous Delivery (CD) as well. CD uses a quality-focused ideology to build potentially shippable product increments, achieving a result that culminates in bringing changes to production as quickly as possible. But because CD may result in frequent large deployments to production, 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 discrete functionality. Rather than using CI/CD to deploy a new version of an entire application or package of services, each microservice may be updated individually by a CD process. This allows 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 each other based on product features. However, in today’s customer-focused world, user experience has now become a key differentiator for insurers. Data can go a long way towards creating a unique and powerful UX for customers. The right analytics of data can be used to inform an Insurer in areas such as product offerings, go-to-market strategy, and an overall improved customer experience.

Such data may come from myriad of sources. The Internet of Things (IoT) might provide information from weather sensors on the ground, telematic devices in vehicles, or networked sensors in commercial buildings and homes. Social media may provide behavioral information that could be used both in real time and for historical risk-rating purposes. Some data sources may be tapped into directly, while others may be made available through aggregators.

Insurers need sophisticated technology, such as Big Data/Analytics and artificial intelligence, to convert collected data into meaningful information. These sophisticated techniques allow insurers to discover otherwise hidden patterns, trends and correlations across complex data sets.

**Application of the Solutions**

As part of the digital transformation journey, it is important to focus on improved customer engagement models, innovations, automation (like 0 touch and state through processing), and improved decision-making. But the journey should not be restricted to digital front ends alone. It should also encompass the building of an efficient and nimble IT ecosystem/architecture from the backend through to the Systems of Engagement. To do so, it is imperative to select the right level of abstraction for the architecture, and to enable the architecture through the use of microservices and an agile operating model supported by a data centric architecture.

Most policy administration and claims management systems have evolved over years and, as a result their complexity and interdependencies have resulted in solutions based on monolithic solutions. Despite this, there has always been a desire for improving time to market and accommodating the frequent changes that are unique to the insurance industry, due to ever changing rules and regulations.

Regardless of this desire, insurers are struggling with microservices – the overall architectural approach, where to start, service granularity, data migration challenges, organizational and cultural changes, and distributed processing challenges. Hence, understanding the future needs, current business drivers, overall organizational structure, and technology environment are important to understand, so as to avoid common anti-patterns and pitfalls, such as:

**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 combine them to a single service responsible for maintaining a customer.

**Data Migration Anti-patterns**

When migrating from a monolithic application to a microservices architecture, the first goal should be to split the functionality of the monolithic application into small, single-purpose services. The second goal typically is to migrate the monolithic data into small databases, or separate schemas, each associated with a service.

Most insurers believe that data should be a corporate, rather than an application asset. But 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 Anti-pattern**

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.

**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.

Understanding the typical impediments and antipatterns that surround an insurance enterprise, there is an inherent need for an overarching enterprise strategy and vison which should be supported by a well drafted architecture guidelines and principles towards Microservices based architecture, APIs and data centricity. Beyond these architectural patterns, it is also important for an organization to make the right level of changes to be more agile and reduce the size of the failures. Below are some of the key steps and recommendations towards building a Microservices based architecture, which on the other hand helps an Insurer move away from some of the existing impediments and challenges with current systems and processes –



* 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.
* Adapt to the principles of Domain Driven Design, and patterns like the circuit breaker, CQRS, event sourcing and anti-corruption as deemed suitable.

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

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.

**Summary**

Summarize everything that this whitepaper has talked about into one or maybe two paragraphs.