Insurance Architecture - Impediments, solution options and recommendations for building a flexible and nimble IT architecture to support digital transformation initiatives

Why read this?

Most insurers are going through this phase of “digital transformation” and have similar challenge which is how to react to the ever changing demands of the market, be customer focused and also maintain the integrity and consistency of the 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, hence this paper talks about the common impediments, solutions on how to surmount these with some real examples. At the same time emphasizes on the need to remain grounded and not add to future complexities and maintenance nightmares by elaborating on some anti-patterns and pitfalls seen across these new architecture patterns and 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 Eugene Loparco (Chief Architect – Insurance) for valuable inputs and contribution to this paper.

Insurance like every other industry in the market is being disrupted by this rapid and pervasive adoption of digital technologies ranging from the smartphones to the Internet of Things (IoT). This technological tsunami is the driving force behind Insurance companies adopting to new business models like use of the weather sensors combined with customer data to provide crop insurance, use of in-vehicle telematics for auto insurance, smart home sensors for property insurance etc.

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 streamline claims processes. Instead 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 their customers, agents and employees to be able to interact with them across different channels and devices, when and wherever from they want to.

Insurers generally tend to lag behind other industries when it comes to 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. Mostly of the Insurance companies are buried deeply under large legacy systems (monoliths) and also manual processes and mostly due to inter woven business processes and primarily because of mergers and acquisitions.

Let's look at some of the key impediments faced by Insurers within their IT landscape even today.

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 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.

Due to ever-increasing demand towards speed to market and changes to underlining rules and regulations, most of the changes need rebuilt, retest, and redeployment, as a result, those 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. Instead the need is to equip and seek data from multiple channels in different formats and use new tools like big data to accept, manage and analyze that data to provide customers with useful information, opportunities and solutions.

In order to move out of such monolithic legacy systems and rigid and age old business processes, the Insurers face some significant impediments even today, and some of the most common ones are -

Legacy Platform

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

- *Strong LOB alignment and ownership leading to implementation of singular executable or applications covering all aspects of the business process and data, mostly known as monoliths*
- *Mergers and acquisitions over years have led to accumulation of technical debt and heterogeneous systems landscape.*

Customer Centricity

Lack of customer-centric approach to doing business.

- *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 Nimbleness

It's a cultural shift to move from the current practices to being agile in true sense.

- *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.*

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.

Most commonly adopted solution patterns to surmount these impediments.

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.

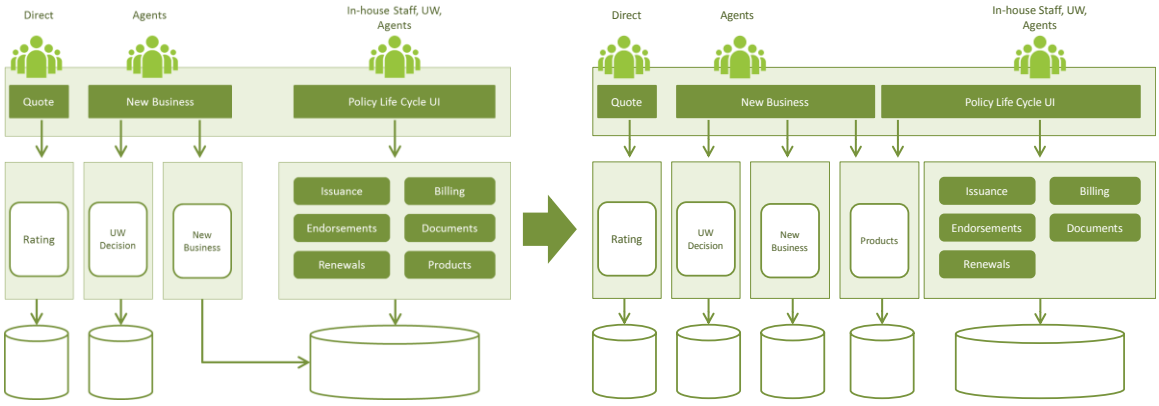
James Lewis and Martin Fowler describe microservices as "an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently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

Monolithic applications are incredibly difficult to keep updated with a frequency that matches market demand. Updating a single portion of the application will often entail retesting and redeploying the entirety of the application.

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.

A good example for that in the context of an insurer is a rating service, because they depend on just rates and rules and have minimum to no dependency on any external or parallel service.

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 shown in the diagram below -



Few more examples and candidates for microservices could be the product and underwriting rules component, first notice of loss for claims and the reference data model, which may include the insurance product model, as most of them can be defined within a bounded context with minimal dependency on any external components and may change frequently as needed.

Development of a microservice-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 approach can facilitate an extremely agile development process – building pieces of the application around singular business processes rather than overwhelmingly large development efforts wherein responsibilities are not clearly delimited.

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 as 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.

“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”

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.

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.

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. Some of the key areas of focus, based on our experience helping define the overall digital and API strategy for insurers are

- 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)
- Straight-through processing (STP) can be made possible through simplified products, automated underwriting for new business
- 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

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

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.

PaaS offerings should be considered to be used as deployment strategy and some of them are:

- Pivotal Cloud Foundry: is good choice when using Spring stack technology because it has a native integration
- Red Hat OpenShift: could be an alternative when using some Red Hat technology. It also uses docker and kubernetes to containerization
- SalesForce Heroku: It abstracts the features implementations such as containers or logging. It's a good choice when building applications using the Twelve Factor App methodology

Other choices to be considered are Amazon Elastic Beanstalk and Google App Engine.

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.

How to apply these solutions to build the foundation but still remain grounded and relevant?

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, but also towards building an efficient and nimbler IT ecosystem/architecture from the backend up to the system of engagement. The right level of abstraction from core legacy

system by the use of microservices based architecture and an agile operating model supported by a data centric architecture is equally important.

When it comes to the core insurance systems or system of record as they are called, 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.

With the recent changes in the industry where organizations are performing deployments and releasing new features to users every hour, insurers are also marching towards leveraging similar mind set, tools and process to get to a stage where changes can be made easy and release features more frequently from what it is today.

However, like SOA, companies developing microservices are finding themselves struggling with things like service granularity, data migration, organizational change, and distributed processing challenges. Hence, understanding the business needs, 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.

- 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 asset and not an application asset. Given that, you can appreciate the risk involved and the concerns raised 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 a 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 all the impediments, solution options and antipatterns associated with a microservices based architecture and for that matter API centricity, below are some of the key steps which an Insurer must take and look at -

- 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.
- Microservices are not an alternate to SOA, continue investing in SOA and build upon it (Microservice is SOA made scalable). Which means continue to build on the SOA patterns and encourage the culture of externalizing and reuse within the enterprise.
- Identify the right candidates to be built as Microservices,
    - 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?
- Some examples for miroservices could be
    - Rating services
    - FNOL for claims
    - Product and Underwriting Rules
- Start with more coarse-grained and then split it further if needed.
- Avoid data migration, data is a corporate asset and not and application asset, data migrations are error prone and complex and will become mpre complex over time.
- Adopt to the principles of Domain Driven Design, but stay away from going towards analysis paralysis.
- Make full use of patterns like the circuit breaker, CQRS, event sourcing and anti-corruption as deemed suitable.

Lastly, start small and experiment, instead of going with a mindset of breaking a monolith to smaller apps, identify the right candidate and migrate progressively, functions which do not qualify as per the above criteria for microservices and are performing well and do not change frequently, can be left alone within the monolith.