Building Service Assurance using efficient automation through component-level abstraction and integration

Composable IT & process fragments

Table of

Contents

Introduction

What is ‘composable IT’?

IT software architectures have heavily influenced the evolution of best practice in telecoms, with developers adopting methodologies, architecture and more to, for example, support the shift to cloud native environments and to reduce service delivery times. One focus area that stands out is automation.

An emerging trend that is likely to have a significant impact on transformation ambitions is “Composable IT”. What is this and how does it relate to telco automation? And how is it different from previous architectures and approaches that have promised to accelerate delivery, enhance automation and more?

In this paper, we’ll explore key concepts and show how you can accelerate automation — across all your processes — by adopting the principles of composability

Background

Simplify to accelerate creativity

For decades, developers have sought to simplify and accelerate service creation, design and delivery in telecoms networks. Most approaches have essentially depended on abstraction. In essence, this is what composable IT seeks to do – but through decomposed, reusable components, which can be assembled into different sequences to deliver desired functionality and to ensure seamless handover between tasks to achieve automated flows.

This abstracted approach has also been used in telecoms networks.

More than 40 years ago, engineers developed the concept of Service Independent Building Blocks, or SIBs that could be used to compose services.

The SIBs provided discrete functional steps and could therefore be assembled into a chain, which defined the service.

This methodology and the tools that support it is still behind some core network services - think emergency call handling, number translation, and so on. The idea was to decouple the desired functions from the underlying complexity through which they would be executed – with one expected outcome being that the group of people who could create and modify services could be extended beyond a small pool of experts.

Efforts to abstract network actions and events

Since then, there have been numerous similar initiatives, often driven from the perspective of services – because services make money, so reducing the cost of building and deploying them, and accelerating time to revenue were key strategic goals.

Composable IT rests on similar principles, but we can extend the applicability to components for everything – going beyond the service to encompass related elements, such as billing, user information, inventory records, provisioning, policies and much more.

As such, it's also applicable to automation.

The idea was to decouple the desired functions from the underlying complexity through which they would be executed – with one expected outcome being that the group of people who could create and modify services could be extended beyond a small pool of experts.

Composable IT

and process fragments

Every system or process can be broken down into discrete steps. We do this already with workflows that characterise any given process. Composable IT takes this concept and builds components that can then be assembled into a flow, bringing the required functions together, in the correct order, so that the process can be instantiated, without knowledge of the underlying technology.

So, if that sounds familiar, then what's really important here? Several things stand out:

1 - Reusability

- The components can be used in multiple systems or processes. Because certain steps or functions are common, the same component can be used in different applications.
- This also means that we can have a finite range of components that abstract pretty much anything we might wish to do, so we can use this library of components to build any kind of service or automations.
- We can also reuse chains of components that fulfil specific functions – grabbing a particular action or part of the flow from here to add to a new one, there.

2 - Interchangeability

- We should be able to remove, change or replace any individual component without disrupting the others that have been selected in the flow.
- This means that we can focus optimisation efforts at a granular level and can avoid the costs associated with the overall design. We can change the sixth in a chain of 10 without impacting the other nine.
- It also means that we can make changes, faster, because the related components will be unaffected by our actions.

3 - Interoperability

- The interfaces between components and from components to the underlying systems and solutions they control and interact with must be interoperable.
- So, between components, we may have APIs to enable parallel interaction, while below them, we may have APIs as well as specific protocols to other platforms.

So, with these principles in mind, what can we really accomplish by adopting composable IT practices – and how can we do it?

Every system or process can be broken down into discrete steps. We do this already with workflows that characterise any given process. Composable IT takes this concept and builds components that can then be assembled into a flow, bringing the required functions together, in the correct order, so that the process can be instantiated, without knowledge of the underlying technology.

In

practice

We need a (finite) library of components that represent stages in different processes. These can vary, but typical stages in, say, the purchase and delivery of a new service include:

- Enter / extract / update records in the CRM, ITSM, VMDR – ensuring the customer data is correctly logged. The IT Service Management (ITSM) systems support tickets that might be allocated to a customer or incident, while the Vulnerability Management, Detection and Response (VMDR) solution should include details of all connections, CPE, and so on, so that all issues can be located – the network inventory is part of that.
- Apply security policy – selecting the appropriate protocol to be applied, based on the service, connection, SLA tier, etc.
- Plan connection – determine the path to the customer, if applicable, or allocating the (e)SIM, essentially setting the parameters for access to the network.
- Activate connection – delivering the desired connectivity, via the selected access method.
- Control service quality – applying in-service actions to ensure consistent delivery of the relevant offer.

Of course, these have dependencies and are interlinked (e.g. control service quality might be required when service performance drops, but it might also be required when a quota has been used, so the service might be interrupted or downgraded according to policies in force).

A service or process is a discrete flow of actions that trigger new events

So, we need to understand a given service or operational flow, in terms of discrete steps that map to the components we have available, and we have to collect the requisite components and assemble (compose) then into the correct sequence of events, so that the actions can be completed and the next step triggered. Now, a component may in fact represent several steps – which distinguishes composable IT from the microservices-based approach that is also gaining traction. Microservices typically represent a single function, so in a way, a composable IT component may interact with multiple microservices in order to fulfil its function.

But, by activating the chain of components, we can secure automation of the service or process, and make changes during its lifecycle – without disrupting the overall flow. We can also do so without reference to the underlying microservices, or elements controlled by the components. The result? We have true independence of the flow from the devices or systems that it seeks to orchestrate. This means that, if the interface from the component to the objects with which it interacts in the network or operational domains changes, we need only update the component that requires this interface (for example, number six in the chain to which we referred earlier). The others remain in place, unchanged. This liberates us from the overall complexity and allows us to focus on individual steps.

Applying composable IT

to enable automation in telecoms

At We Are CORTEX, we have built a library of what we call “Process Fragments”. These are components that can be used, following composable IT principles, to assemble chains that implement any process. They are decoupled from the underlying processes and are also reusable, testable, and independently versioned process assets, so they can easily be inserted into operational journeys.

The Process Fragments also enable ‘plug-and-play’ integration with legacy systems and networks. And, users can experiment with them to assemble the correct flow, because they offer full version control, audit trails – and the opportunity to insert Humans in the Loop (HITL) for checks and where there are known gaps in a flow.

They include — by default — operational control components, as well as those that relate to business processes, governance, control and more. In other words, they offer the complete palette required to manage operations, deliver services and to comply with policies demanded of regulated service providers.

End-to-end service provisioning

Let's consider an example: service provisioning, end-to-end, from purchase to ongoing assurance and billing. At a simple level, this consists of known steps and workflows:

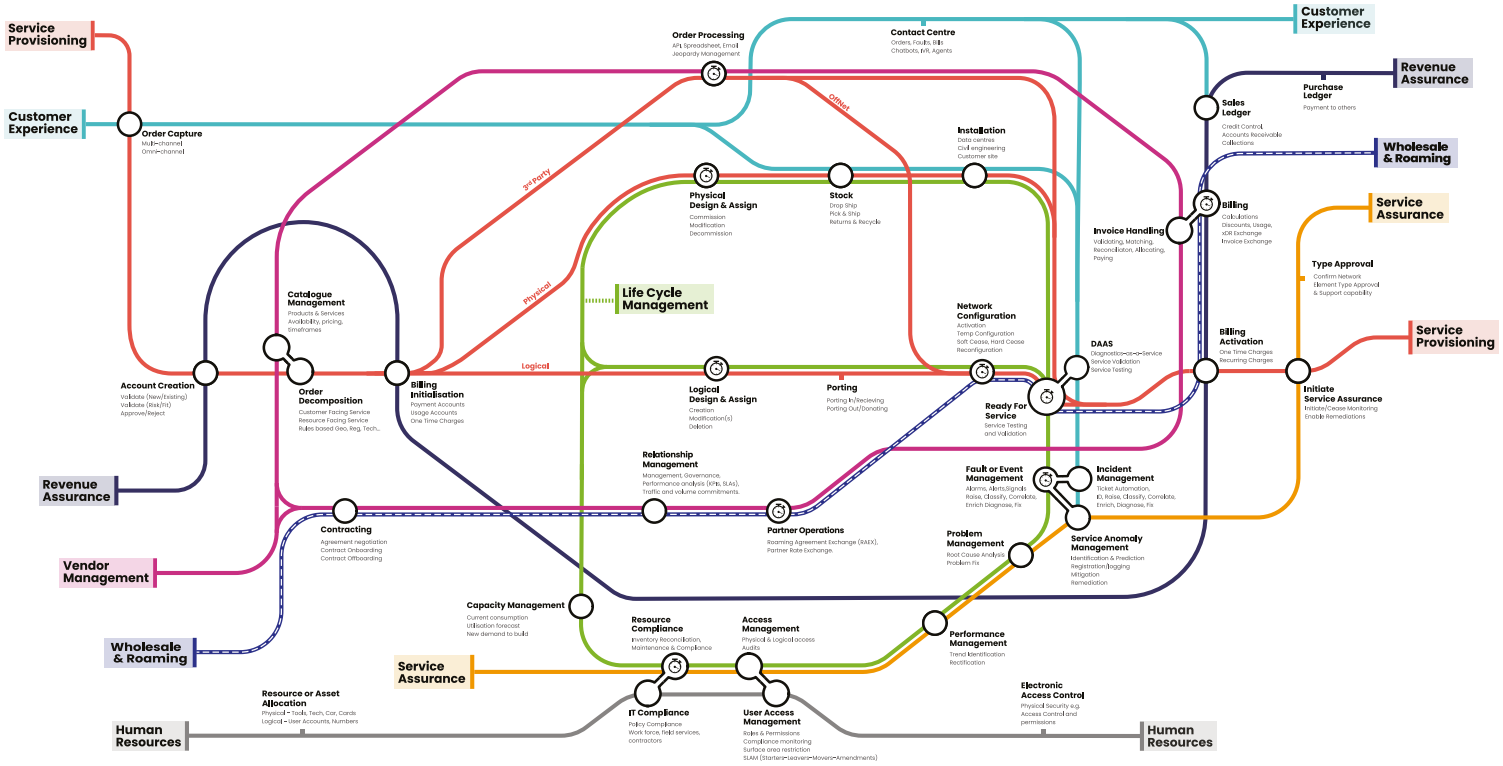
- Service discovery – purchasers review the available portfolio of services
- They choose the desired service and request it
- The availability of the service is checked, and provisioning / activation planned
- The service is activated, billing commences, and the live offer moves to the assured service portfolio

But, while these steps are common to almost any service, there are differences that need to be considered. For example:

- What kind of customer is it? A consumer or a business customer?
- Is the service in the consumer portfolio or business?
- Is it an existing or new account?
- Is the sale handled from a PoS point, like a till in a store – or is it through an existing corporate account, or a self-service platform? Or a partner?
- Which inventory platform holds the requisite details?
- Which SLA applies?
- How is this SLA enforced?
- Which support team is required in the event of any issues or tickets?

So, while the overall processes for consumer and business customers follow similar logic, the details of each step can be very different, calling for different systems and platforms – all to reach the same outcome: a billable service, delivered under the supervision of service assurance teams.

Our 'Tube Map' (Figure 1) illustrates the common processes followed by service providers and operators. It traces journeys for key functions, mapping out the stages in each.



To automate a particular process flow, you need to connect the systems responsible for each stop on the path in the correct sequence, so that when each action has been completed, the next can be initiated, and so on. Let's look at service provisioning from this perspective. As can be seen from the above, the service provisioning journey consists of clearly defined, logical steps. The chain is the same, regardless of the service concerned, or the customer, or how the service is acquired. We simply move from one set of actions to another:

- Order Capture
- Account Creation
- Order Decomposition
- Billing Initialisation
- Logical Design and Assign
- Porting
- Network Configuration
- Ready for Service
- Billing Activation
- Initial Service Assurance

All of these steps are required, even if no action is necessary. Take Porting, for example – there may be no porting to enable, so this is simply a checkpoint that could result in an action, but following a check, the flow can continue.

What matters here is that the logical steps can initiate specific actions. Thus, 'porting' for a fixed only service provider might relate to a fixed number, or a fibre connection to a property, while for a mobile provider, it would likely refer to a number that needs to be brought across from a different provider.

For service provisioning to be automated, end-to-end, you need process fragments, which support composability – and allow adaptation to the specific steps required for a given part of the process (porting, for example – or, to use another case, during 'order decomposition', we may wish to identify the correct inventory system and then process the appropriate interrogation to determine if the necessary logical, service and physical infrastructure is available.

Adapted for real-time operations

The WAC approach is inherently tuned to best practice for composable IT – but is also one that adapts to the demands of real-time telecoms networks, based on multiple generations of technology. It enables both generic tasks to be automated with the specific underlying requirements being adaptable to the needs of each operator (and internal stakeholder), and with respect to the interfaces presented from the operational systems, across the OSS, BSS, network and core.

Once a process or process fragment has been created, it can also be reused. For example, you can see from the Figure above that points on the Service Provisioning flow intersect with others in related flows. These junctions mean that automation for Service Provisioning can also be used to support necessary tasks for, for example, Vendor Management – because this also results in a catalogue.

Similarly, if the underlying specifics change (perhaps because a new method has been added, or an interface is replaced by another), then the flow remains valid, even if the solution triggered by an event changes – think opening a ticket on one platform, which is then replaced by another. A ticket is essentially a ticket with respect to the overall logic.

Our Process Fragments can interact with any element or entity in your network and operational footprint, so they enable the complete process lifecycle as well as accounting and user management.

What's more, they can be interlinked with process fragments used in other flows – and versioning enables all users to maintain the appropriate record for both the complete flow, as well as the elements for which they are responsible.

This enables not only operational agility but also shared responsibility – the Service Provisioning teams play a role in Vendor Management, just as they do in Revenue Assurance, as these flows intersect at critical junctions.

Finally, the Process Fragments that run in a flow can be packaged – but different versions of the packages can be run, so that each is modified to suit the specifics underneath (this part of the flow relates to a business service and uses this inventory, while this version relates to consumer services and uses that inventory) – but all can be run in parallel.

Conclusion

optimisation with We Are CORTEX

The problem, in the end with SIBs was that, despite the abstraction they offered, they were still too complex, and the user pool grew only slightly wider. Users still needed to understand the concepts that were required, even if the problems could be expressed in simple terms.

But with WAC, that vision has been realised, following the principles of Composable IT, adapted for real-time telecoms services. You don't need to understand the underlying systems and you may not even need to understand the process; just the bits for which you are responsible and the necessary decisions and actions at each point.

In this way, automation flows can easily be created and managed with oversight and control shared. They enable concurrent operation of any service or workflow in a dynamic, ever-evolving network environment for mission-critical services and operations.

...with WAC, that vision has been realised, following the principles of Composable IT, adapted for real-time telecoms services. You don't need to understand the underlying systems and you may not even need to understand the process; just the bits for which you are responsible and the necessary decisions and actions at each point.

WE ARE CORTEX

We Are CORTEX

Kings Park House
22 Kings Park Road
Southampton
SO15 2AT

Phone: +44 23 8254 8990
Email: hello@wearecortex.com
Visit: wearecortex.com