Slicing the Elephant: Using an Evolving Architecture Framework to Populate and Deliver the Backlog of Large Cross-Stream Initiatives

NORA SLEUMER, UBS Card Center
MICHAEL SEIZ, Accenture

Big institutions have difficulties delivering large programs that involve multiple streams. This experience report describes the application of an evolving architecture framework and how it can be used to populate and deliver the backlog in a structured and comprehensive yet agile way. The business architect and the IT program manager collaborated in creating a framework of an ever-evolving business architecture flow diagram coupled to applications and Jira issues at various levels of granularity. This enabled transparent and motivating discussions with the value streams and clarity over the data flows between the components in the whole value-delivering process as well as a structured delivery backlog. We called it "Slicing the Elephant".

1. INTRODUCTION

Our story begins in 2022, when the funding for an initiative to extend a customer communication process had come through, and it was time to start delivering. The program team was very small, and expected to supplement, but not replace, the agile practices being applied. Many value streams were involved in the whole communication workflow, with no overarching OKRs (Objectives and Key Results) coordinating the delivery of value. As we got involved we saw that a lot of meetings were held, many opinions given, but the road to actually deploying code into production was very misty. We needed a structure on which to hang the various changes that had to be made, an organizing process to show in what order it made sense to implement them by the various streams, and a larger cohesive vision to guide the team and reassure the management.

2. BACKGROUND

The international company we worked for had switched to an agile way of working in the previous years, with vertically-oriented value streams composed of multiple scrum teams. As long as changes impacted only one part of one business process, the locally-driven agile methods worked fine, prioritizing within the team and delivering according to available capacity. However, as soon as a new business process required multiple value streams to coordinate, the game of hot potato would begin – no one was able to put the issue in the backlog for delivery in the next increment because the work wasn't well-defined. Our program team had no line management or OKR authority over the various teams and thus no compelling incentives to convince teams to prioritize our needs and focus on defining what needed to be done.
The standardized 3-month planning increments were prepared in the previous increment and required some forward-looking cross-stream thinking. This usually consisted of backlog planning sessions using simple lists that were limited to the stream itself. We needed a framework with a long-term vision for moving the whole value generation forward without getting stuck in a huge overarching analysis phase. We developed a guiding principle for how to slice the elephant of a program into pieces that could be put into the various backlogs and onto a delivery timeline.

3. OUR JOURNEY: STARTING POINT, NEXT CHALLENGE AND SET UP

3.1 Starting Point

In an earlier program the focus on the minimal viable "proof of concept" evaluation had been a success, as was the insistence on keeping the sprint results and the backlog up to date, including having all items on a timeline at various levels of granularity. The clear structure and forward planning helped ensure a good handover of

Dr. Nora Sleumer, nora.sleumer@gmail.com
Michael Seiz, michael.seiz.84@gmail.com

activities during the summer after the pandemic, when everyone went on long and well-deserved vacations. The well-defined IT tasks even convinced the business lead of the complexity of the IT integration aspects. It was a big success, slicing the elephant and delivering the bare minimum needed to complete the evaluation.
We learned that taking the time to explain the architecture needs to business colleagues and management at various levels of granularity was worth it in order to complete the evaluation in an efficient manner and arrive at the right solution proposal.
We sometimes got comments such as “in agile you should only plan the next sprint” or “placing items on a timeline is waterfall”. This is not the right interpretation of agile, however. As shown in Jeff Sutherland’s Scrummaster course, which Nora took in 2012, it is a matter of granularity and timing. This is illustrated in the snapshot of the Scrummaster Training Handbook in Figure 1.

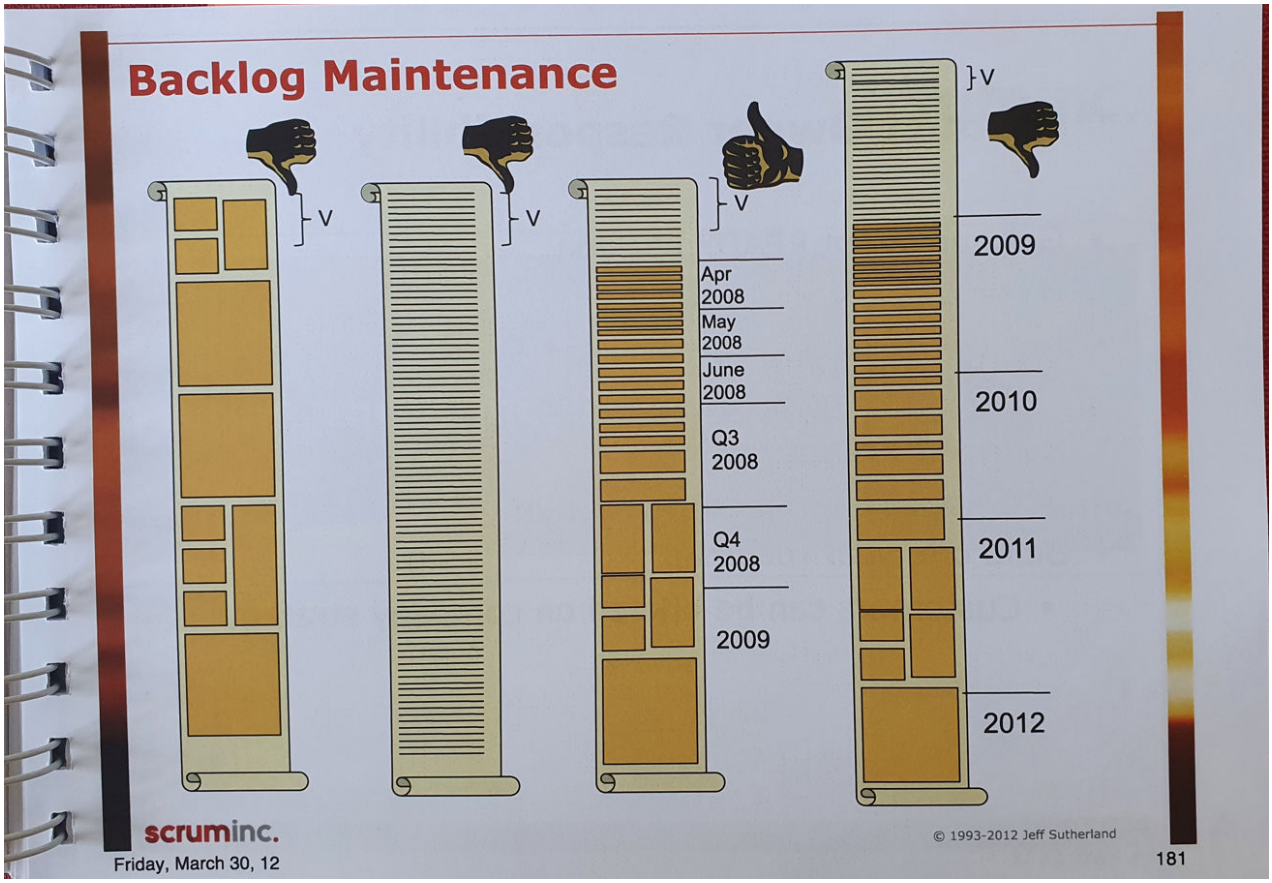

Figure 1. **Backlog Maintenance According to Scrum Inc.**

It is fine to define the upcoming work at a high level of granularity, and the longer-term work at ever coarser levels of granularity and to place them on a timeline.

3.2 The Next Challenge

Our next large program was messier, with the goal of reestablishing customer communication with those not logging in to their accounts, starting a multi-year process as mandated by the regulators. The changes spanned many value streams that were all at capacity and needed to be convinced that their parts were essential to making progress. Business had written some initial use cases in a table that quickly became unmanageable as the non-happy flows and exceptions were brought up. On the one hand we needed to capture all of these elements, while on the other hand we didn’t want to wait with starting the implementation until every single step had been written in pseudocode. We set up a Jira structure with Advanced Roadmap to absorb all the new ideas and present a high-level overview as well as be able to drill down into the details. The structure needed guidelines in order to be logical and practical. It needed an organizing principle to create a larger cohesive vision.

3.3 Set Up

As the Program Architect, Michael oversaw the end-to-end processes, mapping the business flows to the architectural processes and aligning the abstract coarser granularity program-level views with the more detailed views of the teams and their applications. This last part was sometimes quite tricky, as abstraction by definition leaves out details that are sometimes crucial. His challenge was: how to connect overarching views with detailed flow diagrams and develop a valid solution.
Nora was the IT Program Manager, guiding the many streams for which she had no line management responsibility but needing to ensure that the work they had to do for the program was prioritized in their backlogs. OKRs are meant to guide the prioritization for individual value streams but when they are defined at a cross-value stream level they are too broad to have much influence on the increment-by-increment planning. Her challenge was to get the many teams, which are not incentivized by the same objectives, to work together.

3.4 The Problems we had to Solve

The dimensions of the problem:

- Involvement of several “value streams”, teams, organizational units, etc.
- Interfaces between multiple applications: no single End-to-End Ownership
- Changing assumptions & plans
- Dependency on enterprise architecture decisions
- (Unknown) dependencies between value streams
- External requirements/ findings that needed to be closed by a certain date
- Different risk levels of value streams

The main questions:

- How to get to manageable chunks of work?
- How to define the levels of granularity allowing sprint level and program level overviews?
- How to define an MVP and bring it into production so that it provides the first benefits for the users and focuses the team?
- How to best use limited capacities?
- How to manage and control the project deliverables over all the value streams?
- How to do this sustainably?

We solved the problems by defining a slicing approach in two phases:

1. An Incrementally Evolving Target Architecture with a well-defined series of MVPs
2. Slicing the architecture flows into tasks of various granularity and mapping this to a hierarchically-structured backlog

4. AN INCREMENTALLY EVOLVING TARGET ARCHITECTURE

In our agile world, the term "Target Architecture" might evoke notions of a rigid waterfall approach. However, our experience defies this perception. Rather than adhering to a fixed architectural blueprint, we embraced a dynamic process of continuous adjustment. Months of refinement have been fueled by new insights and evolving requirements, acknowledging that there's no such thing as a "Final Architecture" at the outset. This mindset shift is integral to ensuring sustainability, as it allowed us to adapt and respond to changing needs effectively.

4.1 Business Process Flow Architecture

At the top level, our approach centers around the Business Process Flow Architecture, which serves as the foundation for further decomposition into application architecture and data flows. This hierarchical breakdown enables us to gain clarity, facilitating a deeper understanding of system interactions and dependencies at various levels of granularity. Complementing this structural framework are sequence diagrams, providing a visual representation of the flow of interactions within our system.

In Figure 2 we illustrate the business process flow diagram for the extended communication for inactive customers program. Various communications need to be sent out at different times and people, such as the relationship manager, informed. This became surprisingly complex with the addition of monitoring functionality and the resolution of unhappy flows.

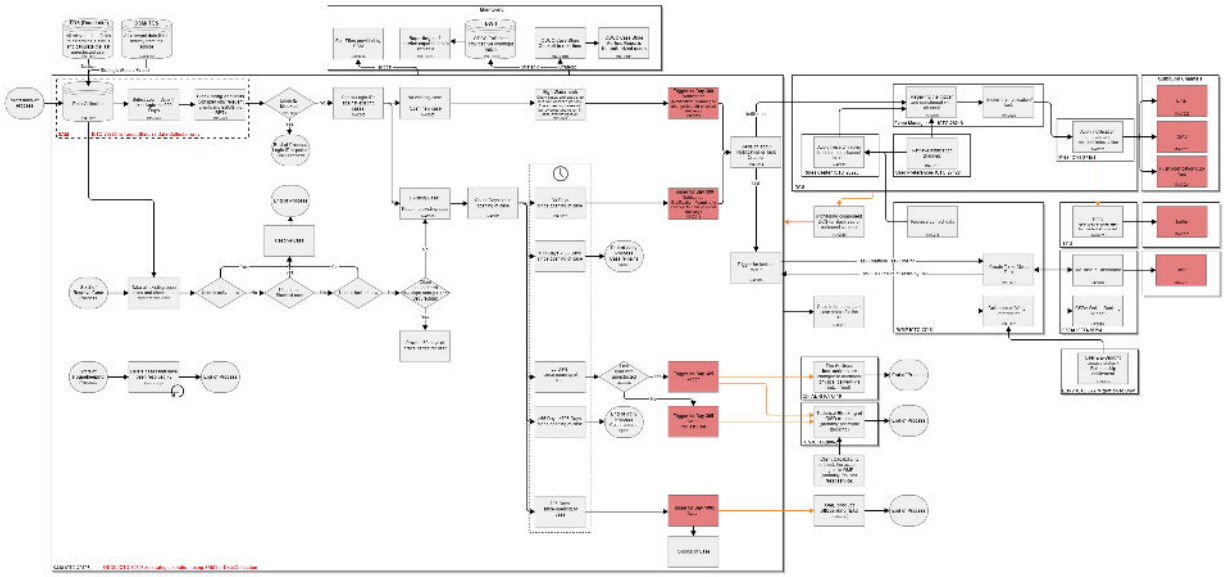

Figure 2. **Business Process Flow Diagram for Inactive Customers Reactivation**

4.2 Explicit Versioning

When a somewhat stable version is reached, it is versioned and baselined and used as a basis for discussion with the stakeholders – management, architects and the various value streams that are involved. We made it clear to everyone that the business process flows are evolving continuously so that the threshold for engagement is lower and people are less hesitant to work on it. No longer needing to deliver a "finished and final" product immediately makes people, especially perfectionist architects, more willing to engage. In Figure 3 we show how the evolution of the Architecture Target State is explicitly identified in the backlog and put on a timeline.

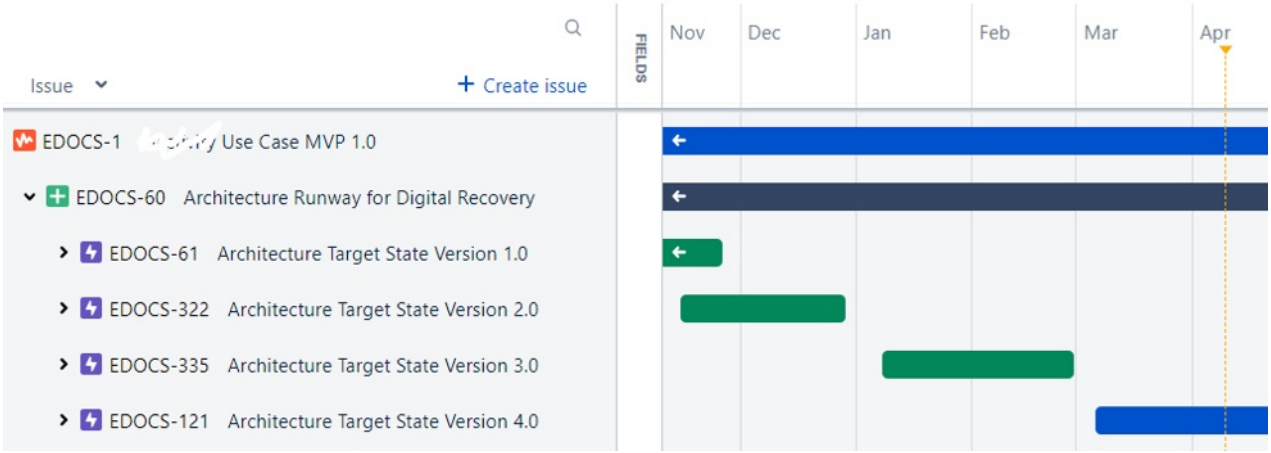

Figure 3. **Explicitly Versioning the Architecture Target State in the Backlog**

By embracing change as a fundamental aspect of our architectural process, we empowered the team to navigate complexity and drive meaningful outcomes in an ever-evolving landscape.

4.3 Identifying a series of MVPs

Recognizing that the grand vision encapsulated in the "Final Target State Architecture" could not be confidently implemented in a single stride, we advocated for a phased approach. Instead of attempting to tackle everything at once, we focused on starting small and learning what really worked. By identifying the minimum end-to-end flow that brought value to the customer, we mitigated the risk of overcommitting

resources and optimized our efforts for maximum impact. This approach aligned with the agile principle of slicing the elephant into manageable Minimum Viable Products (MVPs), each representing a crucial step towards realizing our larger vision.

In Figure 4 the first MVP encapsulating the initial attempts at communicating with an inactive customer was identified as delivering the first slice of business value.

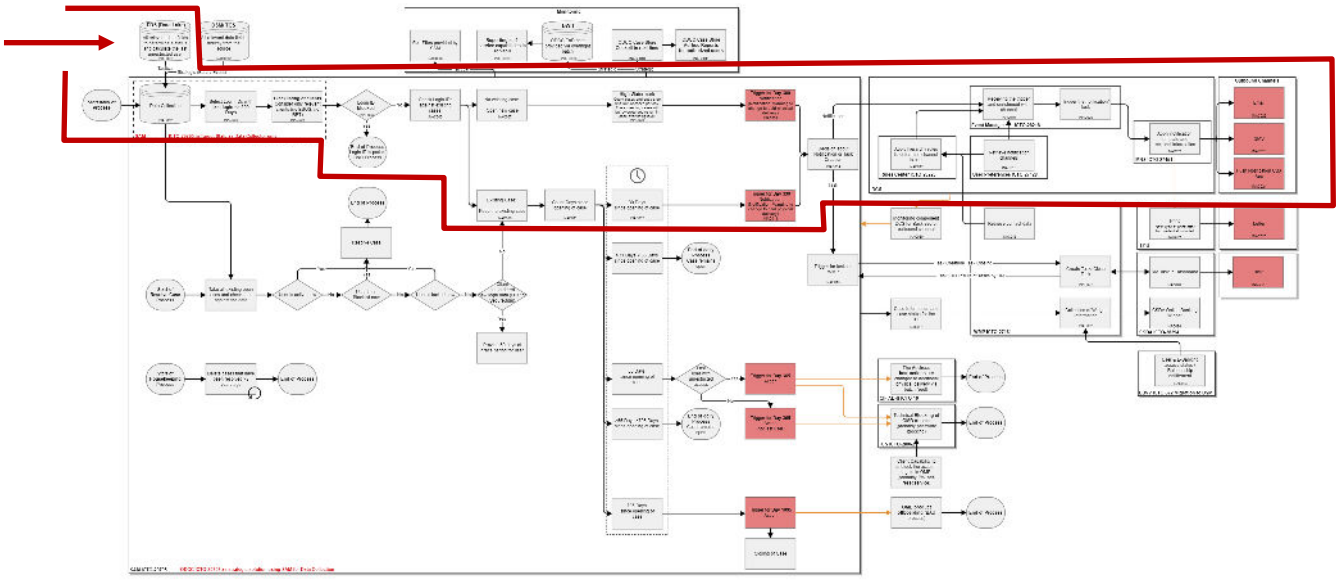

Figure 4. **Identifying the first MVP**

As the program progresses, we identified further slices, building on experiences with the first one. By focusing on each MVP in turn, we ensured that our delivery approach remained lean, iterative, and customer-centric, driving meaningful outcomes while fostering adaptability and resilience in the face of uncertainty.

5. SLICING AT VARIOUS LEVELS OF GRANULARITY: HIERARCHICAL DELIVERY PLAN

In our pursuit of agile excellence, we adhered to the principle of "slicing the big slices," recognizing that breaking down complex tasks into smaller, more manageable components is essential for success. This process extends beyond merely dividing large tasks into smaller ones; it involves a continuous cycle of refinement and decomposition that is a reflection of the architecture business flows. We emphasized the importance of slicing tasks in an organized and sustainable manner, ensuring that each iteration contributes meaningfully to the overall objective. This is not always easy as people tend to get lost in details and to mix concepts of finer and coarser granularity.

5.1 The Structured Hierarchy of the Backlog

Our approach is guided by a structured hierarchy, where

- themes represent the overarching end-to-end user flow,
- features span across value streams, identifying one part of the use case,
- epics are tailored to individual value streams,
- and stories belong to specific scrum teams.

This is illustrated in Figure 5, using Advanced Roadmap. The Theme “EDOCS-1 … Use Case MVP 1.0” is decomposed into the features EDOCS-60, EDOCS-18, EDOCS-143 and EDOCS-431. The implementation feature EDOCS-431 is further decomposed into the epics EDOCS-221, EDOCS-432, DST-10102 and SAMNEW-2437, the latter two belonging to the actual value streams DST and SAM. (Not all features and epics are for implementation purposes – that is a whole discussion in itself and beyond the scope of this experience report.)

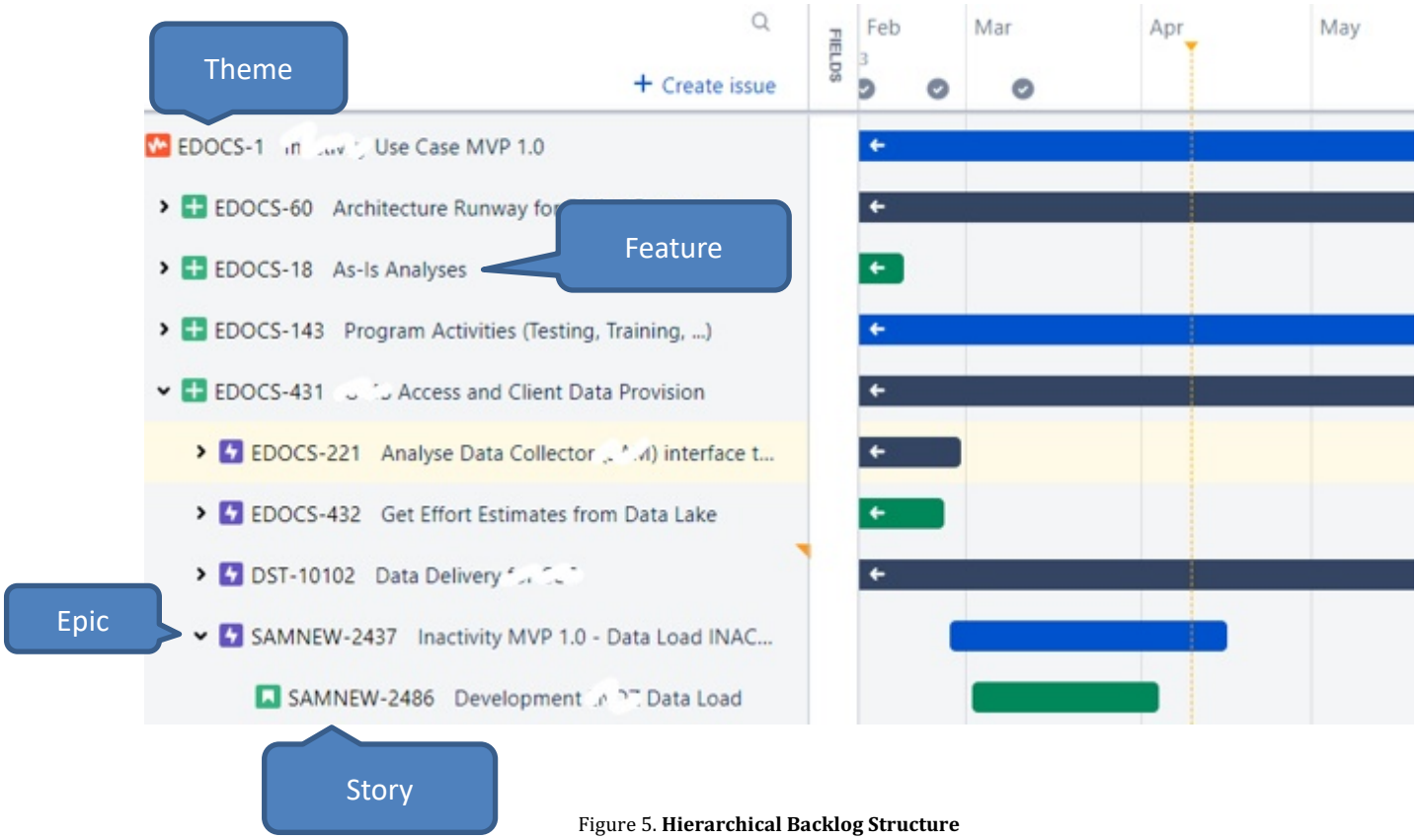

Figure 5. **Hierarchical Backlog Structure**

This hierarchical slicing approach allows us to navigate complexity from the top down – giving a high-level overview at the theme level, and being able to drill down into the details as needed.

5.2 Domain-based Logical process/ development packages

For this generic example case management process for reaching inactive clients, the Value Streams, together with the Program, determine intermediately-sized slices (sub-processes) that can easily be explained to management and team members alike in a sustainable way – allowing high level view drilling down to details.

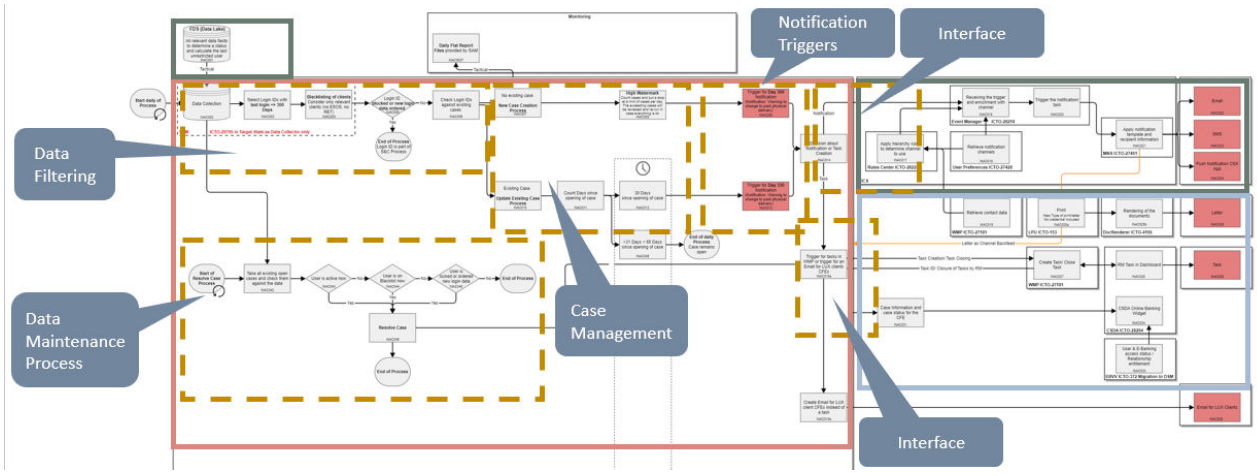

Figure 6. **Domain-based sub-Processes**

Figure 6 illustrates the decomposition of MVP 1 into the various domains that are owned by the value streams and scrum teams that in turn can be further split into development packages.

5.3 Serving the Elephant: Linking the Target Architecture with the Delivery Plan

In our architectural approach, we emphasized the importance of assigning unique identifiers to each box within the architecture diagram. These identifiers serve as crucial markers that facilitate planning and execution of testing processes at both the component and process step levels. By having a unique ID associated with each box, we ensure traceability and accountability throughout the development lifecycle.

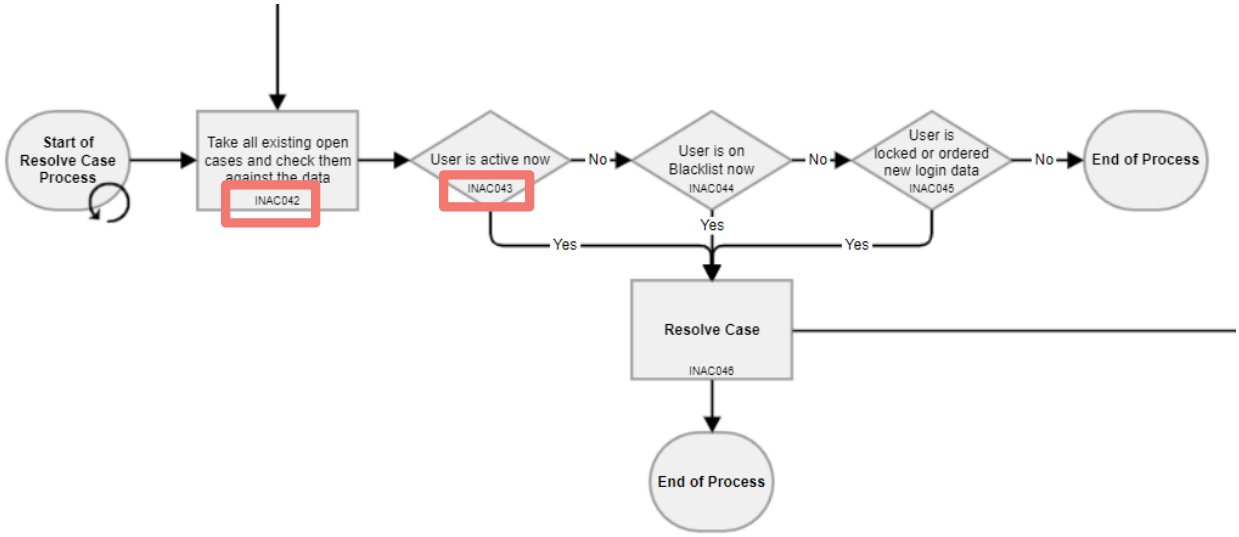

Figure 7. **Identifiers in the Architectural Diagram**

In Figure 7 the steps are labeled with a unique identifier in the architectural diagram. These labels are then integrated in the issues that are captured in the backlog, as shown in Figure 8.

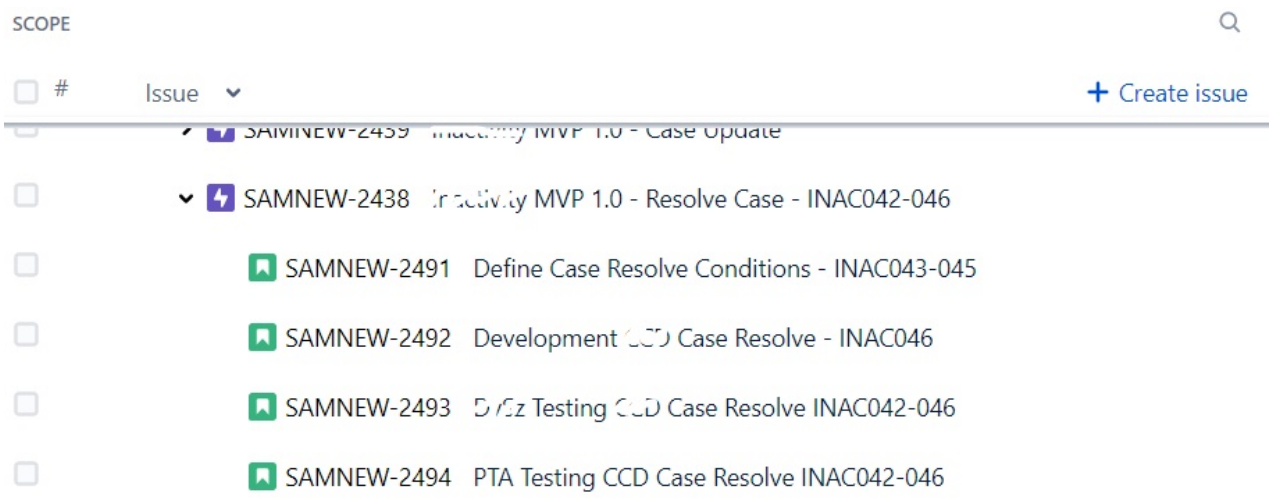

Figure 8. **Corresponding Identifiers in the Backlog**

As complexity inevitably grows, especially in dynamic environments, it becomes paramount to manage it sustainably. We address this challenge by establishing a clear linkage between the architecture and the delivery plan. This integration allows us to navigate increasing complexity effectively while maintaining alignment with our overarching objectives.

5.4 Program Oversight of Critical Parts

Value Streams are accountable within their streams, but interface problems between two value streams require extra attention from the program. It's not enough for each team to just do their part; it all has to work together.

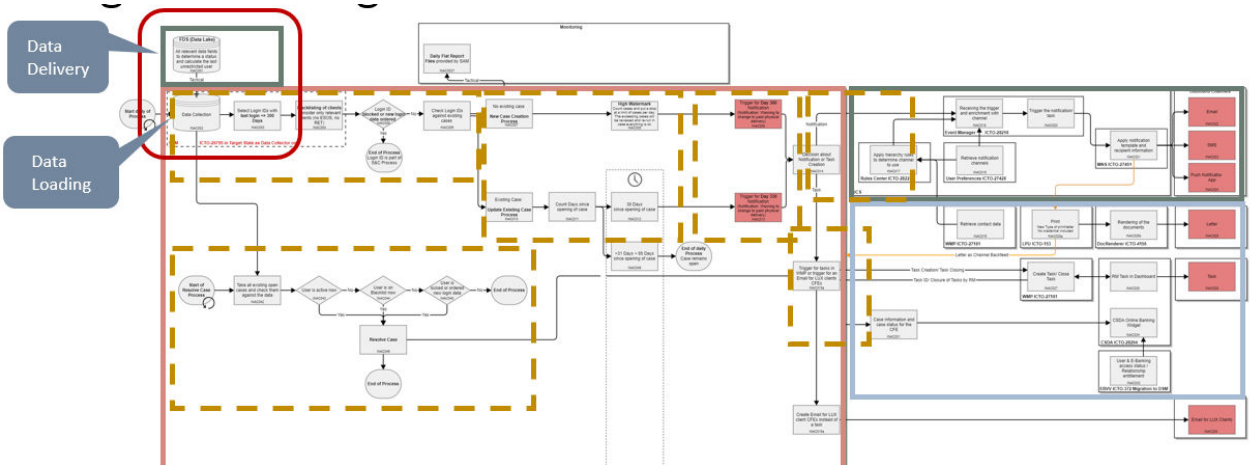

Figure 9. **Identifying Interfaces between two Domains**

In Figure 9 the data delivery team DST has to have an understanding with the data loading team "SAM". This is reflected in the backlog by putting the two epics "DST-10102" and "SAMNEW-2437" into the same feature "EDOCS-431".

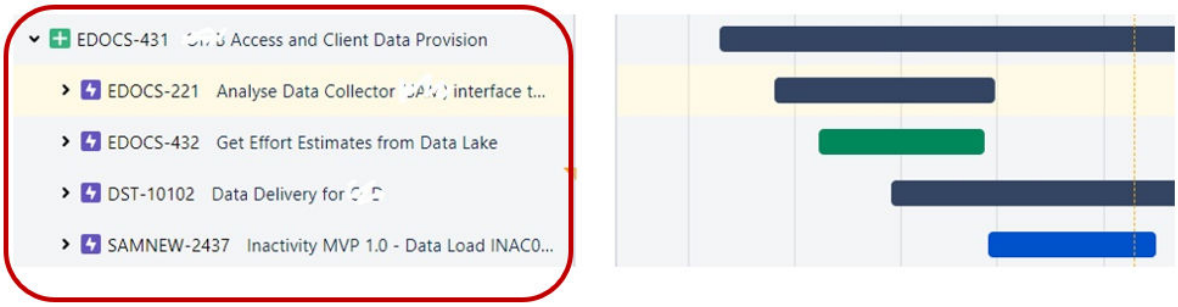

Figure 10. **Placing Tasks on each side of the Interface into the same Feature**

By identifying the interfaces and ensuring that the epics representing each side of the interface are placed in the same feature, the interdependencies of the epics is transparent to both the program and the two teams. In the example above, once the handshake initiates collaboration, the "DST" and "SAM" value streams autonomously assume responsibility for implementing the "Data Delivery" and "Data Loading" epics while being aware of the state of progress of the other team. Documentation is also linked to from the epic, so that each team can easily find the information about the interface that they need.

6. DEALING WITH RESISTANCE AND GAINING MANAGEMENT SUPPORT

We encountered challenges in convincing resource managers to prioritize our backlog items amidst competing demands. With teams already stretched thin, it was imperative to demonstrate the value of our approach. Introducing them to our hierarchical structure and highlighting their crucial role in the end-to-end process proved instrumental in garnering their support. Witnessing the enhanced visibility and productivity afforded by the structured backlog, they not only embraced it for our project but also applied it to others, recognizing its benefits across the board. However, resistance persisted among some individuals, wary of the increased transparency inherent in our approach. Delving into the granular details of each team's backlog raised uncomfortable questions about time allocation and task efficiency, challenging established norms and

autonomy. Yet, this transparency ultimately drove greater efficiency in implementation, fostering a culture of accountability and continuous improvement within our agile ecosystem.

Securing management backing was pivotal to our success. Central to this support was the presence of an engaged and enthusiastic product owner from the business side who grasped the significance of our approach right from the outset. From the earliest stages of development, this product owner advocated for the review and grooming of the multi-dimensional backlog, leveraging a hierarchical structure rather than the conventional context-free linear list. This approach necessitated a collective mental shift within the team, requiring everyone to think in terms of a tree structure and to determine the appropriate level of granularity for describing functionality. Additionally, we faced the challenge of handling components that didn't neatly fit into a single branch. Despite these adjustments, the commitment and vision demonstrated by our product owner galvanized the team, fostering alignment and ensuring that our agile processes were firmly grounded in the realities of business needs and priorities.
In parallel the overarching program lead represented the whole program in such a way that it leveraged the drill down possibility that the hierarchical structure enabled.

7. CONCLUSIONS

We propose an innovative approach to tackling large-scale programs by slicing them into manageable components with various levels of granularity and delivering them incrementally, fostering confidence in our ability to execute. We can highly recommend this approach as it is sustainable, balanced and transparent for all stakeholders.

Central to our strategy is the design of an explicitly evolving target architecture, which serves as a guiding framework for prioritizing Minimum Viable Products (MVPs). Utilizing unique process step IDs ensures clarity and consistency in communication, minimizing the risk of misunderstandings. The additional effort for keeping the hierarchical backlog and the architecture up to date is more than worth it as frustration is reduced, efficiency is increased, and everyone can enjoy the progress that is made.

Hierarchical slicing allows us to create work chunks of varying granularity, facilitating efficient execution and oversight. We explicitly acknowledge the dimensions of evolutionary change and development increments over time, encouraging an agile mindset and readiness to adapt to evolving requirements. Ownership is assigned according to the hierarchy level, promoting accountability and alignment with organizational goals.

By implementing these principles, we established a foundation for efficiently and sustainably delivering large-scale programs, navigating complexity with agility and confidence.

8. ACKNOWLEDGEMENTS

The authors would like to thank the program leads Hannah Bischof and Marcin Piwonski for their patience while we evolved our approach, and their enthusiastic adoption when they realized how it all worked together. We would also like to thank our Shepherd, Ademar Aguiar, for asking the pertinent questions right from the start, which guided us throughout the writing process.

REFERENCES

Sutherland, Jeff. “Scrummaster Traning” Scrum Inc., 2012