Prodapt powering global telecom

Building Digital Enablement Layer – Part 1

Provide robust digital capabilities and improve NPS by **3X**

Credits | **Derrek Schutman** | **Sumit Thakur**

Most DSPs Aim to Provide **Digital Capabilities** to Customers but Struggle to Transform

According to McKinsey research - 70% of digital transformation projects don't reach their stated goals

Vertical Slicing

Transform per business segment or product, covering everything from top to bottom of BSS/OSS

When | Preferred if back-ends are not suitable to provide needed capabilities

Complexity | Full IT transformation tends to fail. Too complex to take impact all over.

- **"Big bang" approach based on an operator driven architecture** (with homegrown tooling) is very costly, adds more complexity, and impacts the running business.
- **Utilizing COTS tooling** to create a truly digital set of capabilities often requires a multi-vendor strategy (best-of-breed). This is an extremely fragmented and scattered approach.

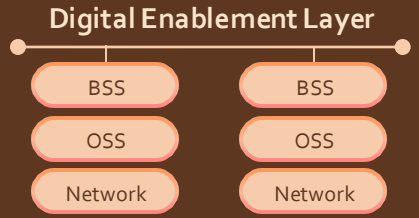
Typical **approaches** DSPs take to provide **digital capabilities**

Telcos with robust digital capabilities boast a profit margin of **43%**, compared to their counterparts whose margins hover around **21%**.

To capitalize on this opportunity, a **horizontal Digital Enablement Layer is an optimal approach.** This can be achieved with reasonable **budget & timeline (max 10% of total IT CAPEX for transformation, and 1-year lead time,** if done right).

Horizontal Slicing

Build a horizontal digital enablement layer on top of the traditional IT systems



When | Preferred if back-ends have reasonably adequate functionality and can be transformed over a period

Advantage | Hides back-end complexity. Paves the way for a smooth transition/IT transformation without affecting the business' digital needs significantly

- **Provides** a **bridge between the digital and traditional world,** allowing both to co-exist with seamless switching between them.
- **Ensures business operability and continuity** while allowing the reuse of needed legacy functions underneath.
- **Enables gradual modernization** of legacy O/BSS to a digital platform.

Digital Enablement Layer Empowers DSPs with Robust Digital Capabilities

To ensure a successful transformation, DSPs must take a holistic approach

3 Pillars of Successful Transformation

Technology

Implementation Strategy

Organizational Culture and Mindset

Digital Enablement Layer

- Decoupling & creating layered microservices architecture
- Creating centralized mono repository

Use of Industry Standards to Guide Implementation

- Setting up clear transition architecture
- Defining clear demarcations & layering in the implementation

People and Process Transformation

- Applying SAFE/Agile in Telecom
- Developing talent with digital skills; digital product management

In this **PART-1** of the insight, we explore the **Technology** pillar highlighting key focus areas to consider for building an effective ***Digital Enablement Layer***.

PART-2 of the insight will cover the **other two pillars** in more detail.

Key Focus Areas to Consider for Building Digital Enablement Layer

Provide robust digital capabilities to improve NPS by 3X and ensure operational excellence

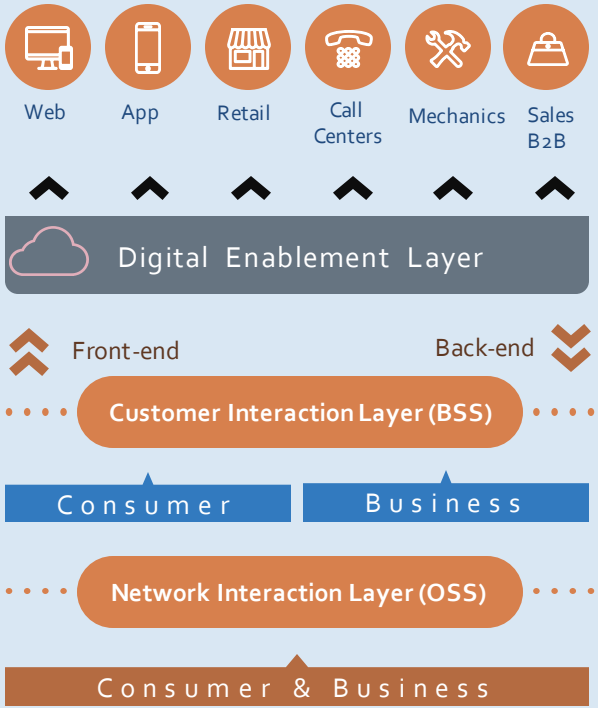


Fig: Implementing "Digital Enablement Layer"

These **4 focus areas** with the given recommendations can enable DSPs to avoid pitfalls and achieve digital capabilities successfully.

1. **Decoupling customer-centric engagement**
2. **Bridging the gap between domains**
3. **Ensuring visual consistency in omnichannel experience**
4. **Creating a centralized mono repository of micro applications**

Implementing a **mono repository** in the right approach significantly **increases re-use potential** and **innovation time.**

Provides **50%** re-use potential for the development of the 2nd portal and onwards.

With multiple brands/segments/ user groups, re-use potential can increase up to **70%.**

Significantly reduces **innovation time**, with high levels of automation in the CI/CD, focusing on the needs of digital layers (e.g., security & usability testing)

Decouple Customer-centric Engagement Layer from the Current O/BSS

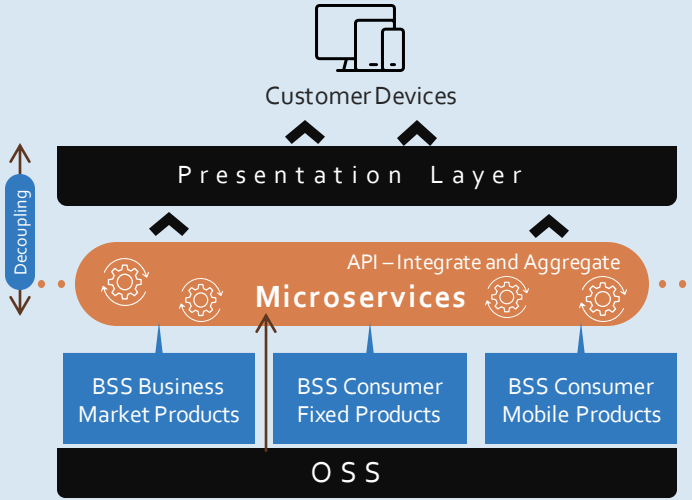
1 Decoupling customer-centric engagement

Customer-centric back-end features need to be decoupled from the existing O/BSS to create a layered microservices architecture.

A decoupled and layered architecture creates a separation of concerns.

It sets clear boundaries in responsibilities and makes it easier to isolate impact areas.

A required change in UI/UX should not impact the back-end functionality and vice versa.



Decoupling the front-end from the back-end will make changes purely on the UI faster & cheaper. This can result in a **25% reduction in effort and lead time**

Build the core API functionality

- **Standardize** access to existing data and services (e.g., customer details in mobile, ZIP code check for fixed, unbilled usage, etc.)
- **Normalize data** between separate domains (e.g., customer data delivered by the API is always in the same format regardless of the source)
- **Re-use** API logic for different channels

Separation of concerns by creating clear boundaries

- Ensure **one functionality resides in only one microservice** and not multiple.
- Set **clear responsibilities** for each layer and define the edges of the layer very clearly. E.g., caching is done in each layer with a different scope. Thus, it's functionality at each layer needs to be defined clearly to create clear boundaries.

Create layering with complexity spread evenly

- Creating something top or bottom-heavy will result in complexity. E.g., business logic can be handled in the microservices layer and not in the presentation layer.
- In cases where required and possible, implement a **headless architecture**, with clear demarcation of UI and logic. **API-based connection** (e.g., RESTful API or Storefront API) to back-end enables DSPs to have the front-end technology of their choice because the content isn't bound by a predetermined user interface.

Have optimal number of layers

- Creating a **multi-layer** architecture is required but certain layers can be avoided if they don't add value. E.g., Directly expose OSS services if there's no BSS involvement/translation required while utilizing COTS/out-of-the-box capabilities of back-end tools.

Bridge the Gap Between Domains with an Aggregation Layer to Create and Orchestrate Cross-domain Events

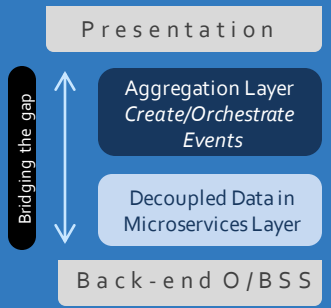
2

Bridging the gap between domains

The key reason for introducing APIs is to increase speed and flexibility by decoupling channels from the back-end. However, in order to support true digital capabilities, more than just decoupling is needed.

An additional dimension is required to combine data and services across various domains into new unified services.

This implies that the decoupling layer not only needs to decouple back-end functionalities but also needs to create new data and orchestrate events to support unified customer experiences across channels and product segments.



Use domain driven design

- Techniques like **event storming** creates appropriate domain models that form the basis of the architecture for microservices and the boundaries between them.
- **Change data capture techniques** are the key to make this work with existing/legacy systems and enable incremental transformation.

Use back-end capabilities

- Back-end applications are good in a function (e.g., billing, rating), but they are not created to be brilliantin presenting this.
- Use technology to **combine cross-domain** back-end information, add business logic/rules, and present logically in the right format as needed by the UI.

Ensure smart combinations

- If the back-end landscape is scattered – e.g., stovepipes per product, combine smartly to hide the complexity.
- Use **smart combinations** based on datasets. E.g., **combine product data and usage data** to advise on bundle up/downgrades or renewal of the contract with better conditions and price.

Orchestrate front-end events to back-end systems

- Manage complete status and orders while **orchestrating corresponding events** in the back-ends. E.g., an incoming event on Web care can have a corresponding event in the fixed BSS, mobile BSS, or both.

Ensure Visual Consistency in Omnichannel Experience

Ensure that independent channel interactions coordinate to create one cohesive & consistent customer experience

Ensuring visual consistency in omnichannel experience 3

Customers engage with DSPs across various channels, including the web, mobile, kiosks, online chat, and by visiting storefronts. Visual inconsistencies across channels might signal different functionality, flows, or offerings.

Design for a seamless handoff with great visual consistency

- **Standardize the UI and UX building blocks** for multiple front-end channels using technologies such as **"FrontX."**
- Incorporate **work baskets** in the process for better handoffs.
- **Integrate digital and physical** aspects of the user experience as a single process, rather than separate processes. Integrate back-end systems and let them synchronize data on the fly.

Use multi-brand digital style guide

- Use a **similar style guide** on all different channels. This **ensures familiarity**, where users can take advantage of any knowledge acquired in previous interactions.
- Style guide made in HTML, SASS, and JS can consist of all the UI elements, color schemes, and screen layouts.

Optimize loading times

- On a mobile phone, customers expect similar performance as on a desktop.
- Make use of **smart browser caching** to optimize loading times.
- Limit sizes of media used; call web services only when absolutely needed.

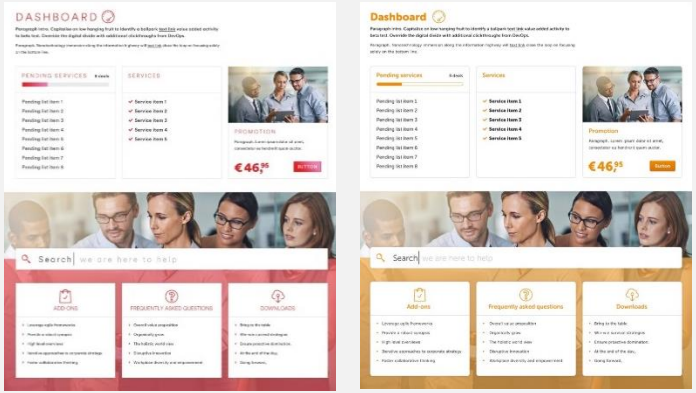


Fig: Sample image of using multi-brand digital style guide

Standardized UI and UX building blocks save front-end development time up to **25%** and budget up to **70%** while ensuring a consistent experience for multiple channels.

Make use of smart digital helpers

Leverage concepts like **smart recommendations** by intelligent chatbots, **next best action prediction** based on Big Data analysis and Artificial Intelligence (e.g., recognize current customer emotion based on chatbot input to increase customer engagement)

Create Centralized **Mono Repository** to Drive Reusability of Micro Elements

Build once and use multiple times to scale operations

A B

Creating centralized mono repository of micro applications 4

Existing O/BSS architecture prohibits reusability

The provided APIs lack front-end focus and business logic.

Business logic is something that can be centralized so that many front-ends can use this.

The back-ends should remain focused on their core business and process.

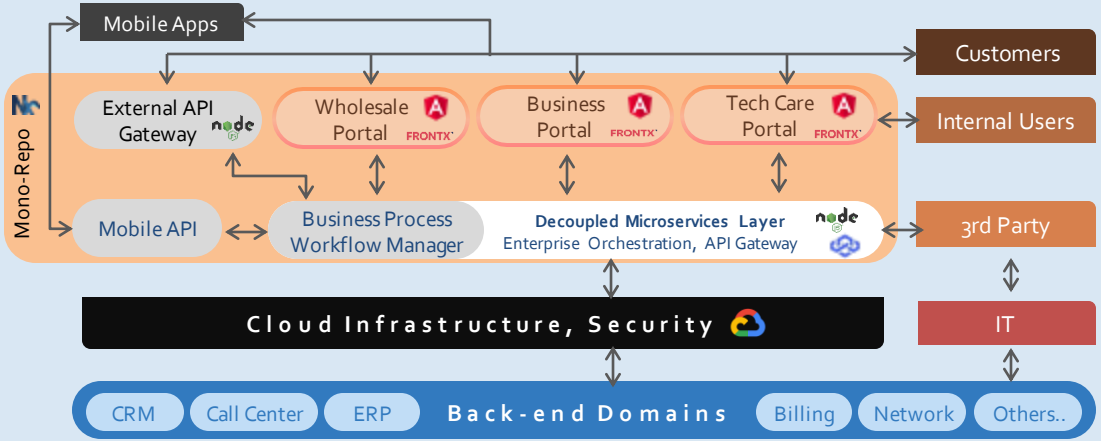


Fig: Sample representation of Mono Repository Architecture

Centralized mono repository drives **re-use potential** for the Nth portal to **40-60%** of the functionality. This also reduces **regression testing time** by **30%.**

Store micro applications as a reusable snippet of code in a single repository

A single repository will clearly indicate dependencies for regression testing and focus on those aspects that are really affected. This **reduces cycle times** significantly.

Ensure the reusability of back-end modules

- Create a reusable back-end module that can be used in various user journeys.
- This drives reusability not only in the UI part but also in E2E functionality.
- Reusability of the APIs is directly proportional to the ease of finding the right API at the right time.
- Also, it is equally crucial to address the challenges brought by frequent API evolution.

Create Centralized **Mono Repository** to Drive Reusability of Micro Elements

Build once and use multiple times to scale operations

A B

Make APIs more findable and standardize to increase re-use potential

- Findability and reusability are driven by the way **APIs and functionalities are documented**. Maintain detailed documentation of APIs, its functionalities and implementation choices - written in an easily understandable format.
- Create additional **discovery-specific services** to improve the findability of API products.
  - Use **design-time discovery** to make it easier for API users to learn about API's existence, its functionality, and the use cases that it can solve.
  - Use **run-time discovery** to help software clients find the network location of API, based on a set of filters or parameters.
- **Define coding standards** with clear guidelines in place. A uniform appearance improves the readability and maintainability of the code.

Defining Coding Standard

Consistency	Safety	Maintainable	Scalable
Readable	Modern	Ownership	CI/CD ready

Fig: High-level coding standards (which require detailed definitions underneath)

Handle API creation/evolution in a much smoother way with automation in CI/CD and testing

- Keeping consistent API versioning, automated code quality checks, validating coding standards, automated UI & browser testing, and automated security checks are essential to smoothen the deployments.
- E.g., Making **"test & security"** checks as part of the CI/CD pipeline improves quality and reduces testing effort, thereby decreasing time-to-market.
- For the **Lifecycle management** of the APIs a clear versioning and support mechanism should be agreed upon and carried out. This ensures all-consuming parties move to the latest version of the API. For this automated deployment & testing (CI/CD) are also critical. Clear documentation & release notes (like with off-the-shelf applications) help in determining the changes/impact.

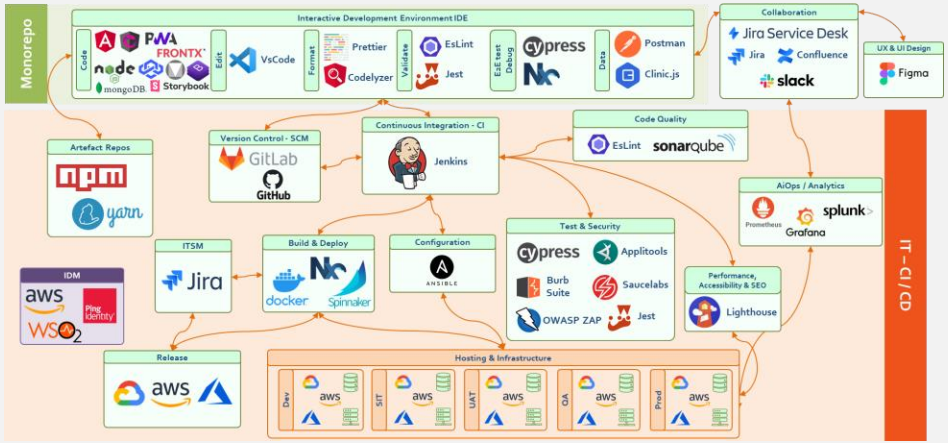


Fig: Sample CI/CD pipeline with tools focusing on quality and automation

Benefits Achieved by a Leading DSP In Europe After Implementing Digital Enablement Layer

Improved Customer Engagement

Enhanced User Experience

NPS Score

3x

Mobile App Rating

3 to 4.7

The DSP faced major challenges in providing digital capabilities. This led to poor CX (Customer Experience) and missed opportunities.

Implementing the recommended approach as discussed in this insight resulted in the following benefits.

Availability

99.999%

70%

Cost

Reduction in OPEX through cloudification

Increased Revenue

Boost annual channel portal revenue by

50%

Reduce time to market by

35%

Sales conversion rates for eCom channel via the portal

20-30%

Orders to visitors % increase

33%

higher than the annual average!

Get in touch

USA

Prodapt North America, Inc.
Oregon: 10260 SW Greenburg Road, Portland
Phone: +1 503 636 3737

Dallas: 1333, Corporate Dr., Suite 101, Irving
Phone: +1 972 201 9009

New York: 1 Bridge Street, Irvington
Phone: +1 646 403 8161

CANADA

Prodapt Canada, Inc.
Vancouver: 777, Hornby Street,
Suite 600, BC V6Z 1S4
Phone: +1 503 210 0107

PANAMA

Prodapt Panama, Inc.
Panama Pacifico: Suite No 206, Building 3815
Phone: +1 503 636 3737

UK

Prodapt (UK) Limited
Reading: Suite 277, 200 Brook Drive,
Green Park, RG2 6UB
Phone: +44 (0) 11 8900 1068

IRELAND

Prodapt Ireland Limited
Dublin: Suite 3, One earlsfort centre,
lower hatch street
Phone: +44 (0) 11 8900 1068

EUROPE

Prodapt Solutions Europe &
Prodapt Consulting B.V.
Rijswijk: De Bruyn Kopsstraat 14
Phone: +31 (0) 70 4140722

Prodapt Germany GmbH
Münich: Brienner Straße 12, 80333
Phone: +31 (0) 70 4140722

Prodapt Digital Solution LLC
Zagreb: Grand Centar,
Hektorovićeva ulica 2, 10 000

Prodapt Switzerland GmbH
Zurich: Muhlebachstrasse 54,
8008 Zürich

Prodapt Austria GmbH
Vienna: Karlsplatz 3/19 1010
Phone: +31 (0) 70 4140722

SOUTH AFRICA

Prodapt SA (Pty) Ltd.
Johannesburg: No. 3, 3rd Avenue,
Rivonia
Phone: +27 (0) 11 259 4000

INDIA

Prodapt Solutions Pvt. Ltd.
Chennai: Prince Infocity II, OMR
Phone: +91 44 4903 3000

"Chennai One" SEZ, Thoraipakkam
Phone: +91 44 4230 2300

IIT Madras Research Park II,
3rd floor, Kanagam Road, Taramani
Phone: +91 44 4903 3020

Bangalore: "CareerNet Campus"
2nd floor, No. 53, Devarabisana Halli,
Phone: +91 80 4655 7008

THANK YOU!

Prodapt powering global telecom