How Domain-Driven Design Can Boost Legacy Systems Modernization

Legacy modernization initiatives struggle to maintain business alignment when business and IT leaders treat it as merely a technology refresh exercise – even as COVID-19 accelerates such modernization demands. By transforming legacy systems into a set of services and applications based on domain-driven design principles, business can be a fully participating partner throughout the modernization journey.

Executive Summary

Heraclitus, the Greek philosopher, was indeed prophetic when he said that *change* is the only *constant* in life. This principle applies particularly well to IT: When almost everything is in flux, how long can enterprise backbone applications such as airline systems, core banking platforms, trading platforms or billing applications remain rooted in legacy architectures? Not long, the COVID-19 crisis notwithstanding.

Industry estimates indicate that the global market size for modernization services will reach $24.8 billion by 2024, from $11.4 billion in 2019.[1] Over 100 billion lines of code

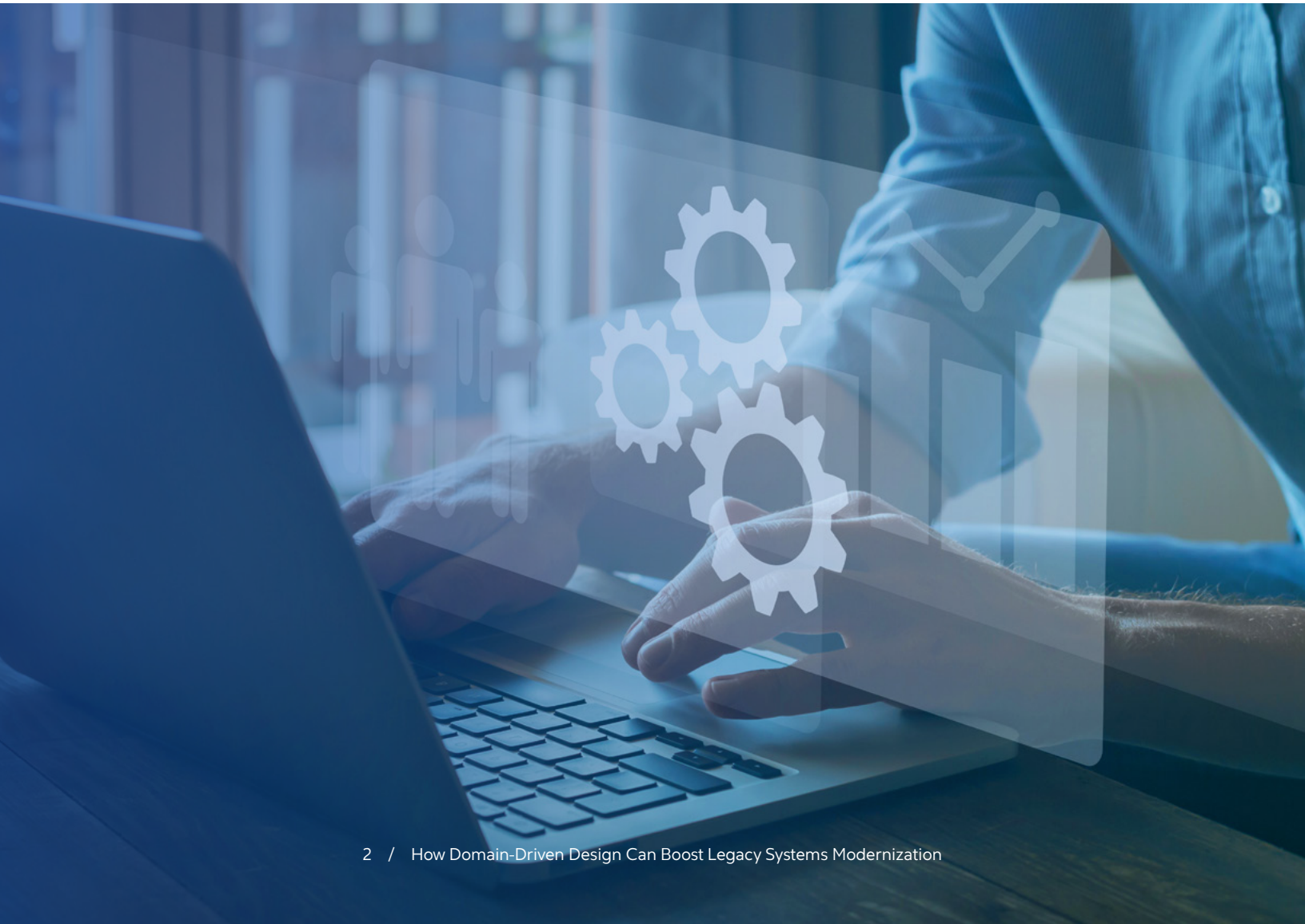
Cognizant

still exist in legacy technologies such as COBOL,[2] indicating the mammoth volume of work ahead.

Legacy applications, in general, are monolithic by nature, support critical business processes and are hardened over a period of time – often with little updated documentation. Thus, modernizing these applications is quite challenging.[3] Another reason it is such a hurdle: Modernization is not just a technology exercise; immersive user experience and redefining business processes to leverage modern digital capabilities are critically important concerns.

The technology industry has developed various design patterns to solve technological problems, such as better organizing code. However, today's modernization needs are more demanding, which makes business cooperation of paramount importance. Domain-driven design (DDD) offers a conducive bridge between the technology and business sides of the organization via the creation of a ubiquitous language that is understandable by both domain experts and technologists.

This white paper is based on our work in modernizing a core legacy application for Kvaerner, a leading Norway-based provider of engineering, procurement and construction (EPC) services. (See "Modernizing Legacy Applications at Kvaerner," next page.) We will show how DDD principles[4] can be used for gradual (progressive) modernization of a monolithic system into sets of domain-centric services and persona-driven apps.



Quick Take

Modernizing Legacy Applications at Kvaerner

Kvaerner is a multinational heavy engineering company engaged in the fabrication of construction materials. The Norway-based company has an in-house application, built over many years, that relies on legacy technologies to handle the core business processes of materials management, construction management and completion verification. With rapid externalization and the need for collaboration, Kvaerner wanted to modernize progressively (part by part) without affecting existing projects which are managed via the legacy system.

Working with Kvaerner, we looked at the legacy monolithic system, its business processes and user personas involved, and arranged them into a set of bounded contexts. This resulted in the collection of related domain services (a microservice exposing a particular domain feature). We then created persona-based apps via a microservices architecture – while maintaining a strong connection with the business side.

To illustrate this, we highlight the work conducted to transform the company's material order and warehouse environment. This legacy system provides the capability to order material from a warehouse for the construction of heavy objects as part of planned work orders.

As legacy systems are typically monolithic, underlying repositories hold business logic. They reference other domain objects directly, which results in a tightly coupled architecture. This white paper demonstrates how to modernize the material order function within the warehouse domain in a legacy system that is tightly coupled with associated domains and also how to form a set of cohesive domain-centric services independent of other capabilities. Transformed modules in the resulting new ecosystem, however, must still coexist with old ones, as described below. This paper also examines how this newly formed set of domain services can be aggregated to deliver end-to-end business capabilities to specific user personas.

Envisioning progressive modernization

Progressive modernization with domain-centricity provides a framework to transform a legacy system part by part, by strangling the monolith[5] (see Figure 1). This will ensure incremental business value realization and allow for any course corrections during the modernization journey.

Strangling the monolith

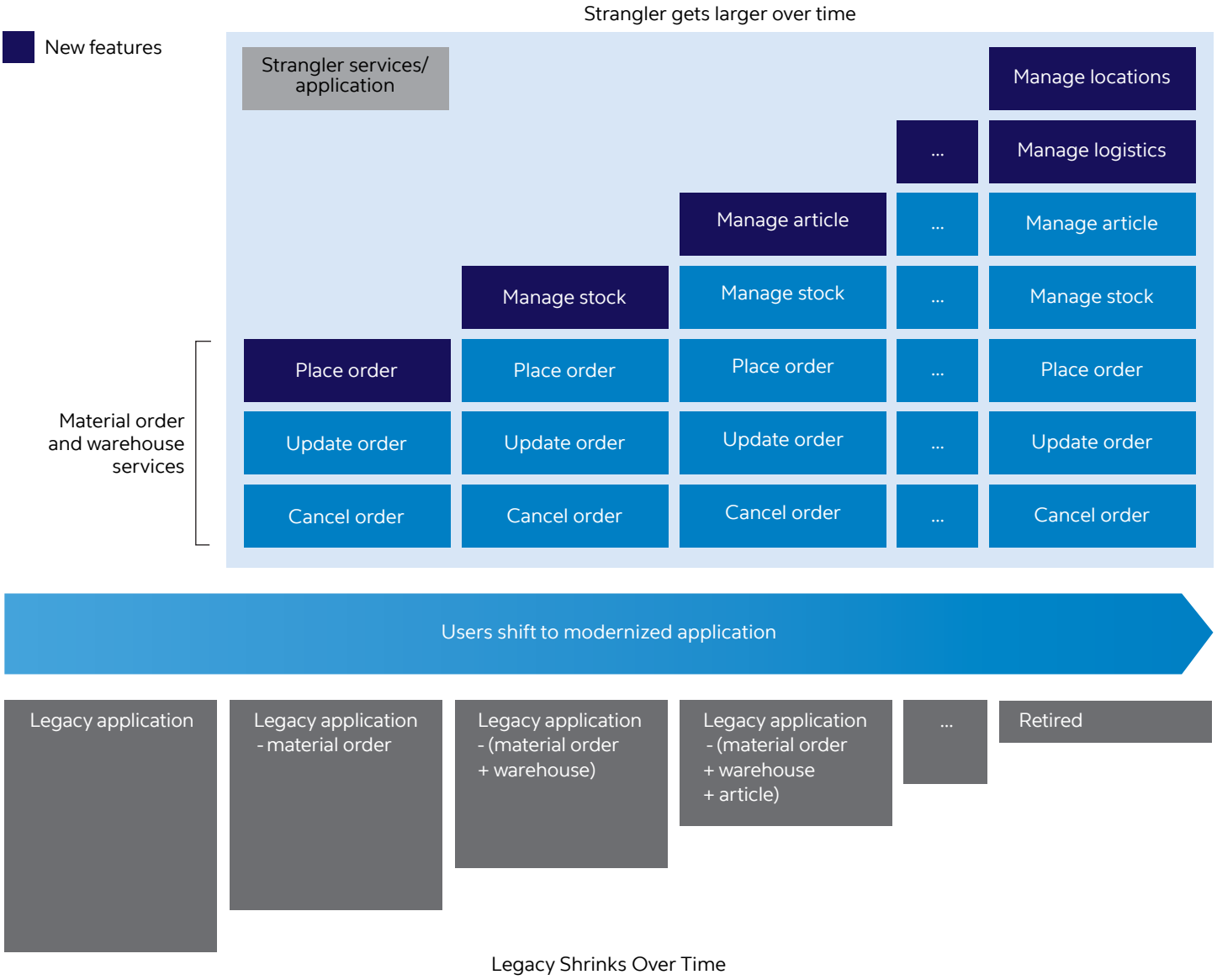


Figure 1

Progressive modernization: Implementing the concept

We propose a modernization framework based on user personas and DDD concepts (see Figure 2). In the world of DDD, the domains are contextualized further to "bounded context" to provide an unambiguous, unique functional understanding. This is particularly necessary when the domains are

A framework for modernization

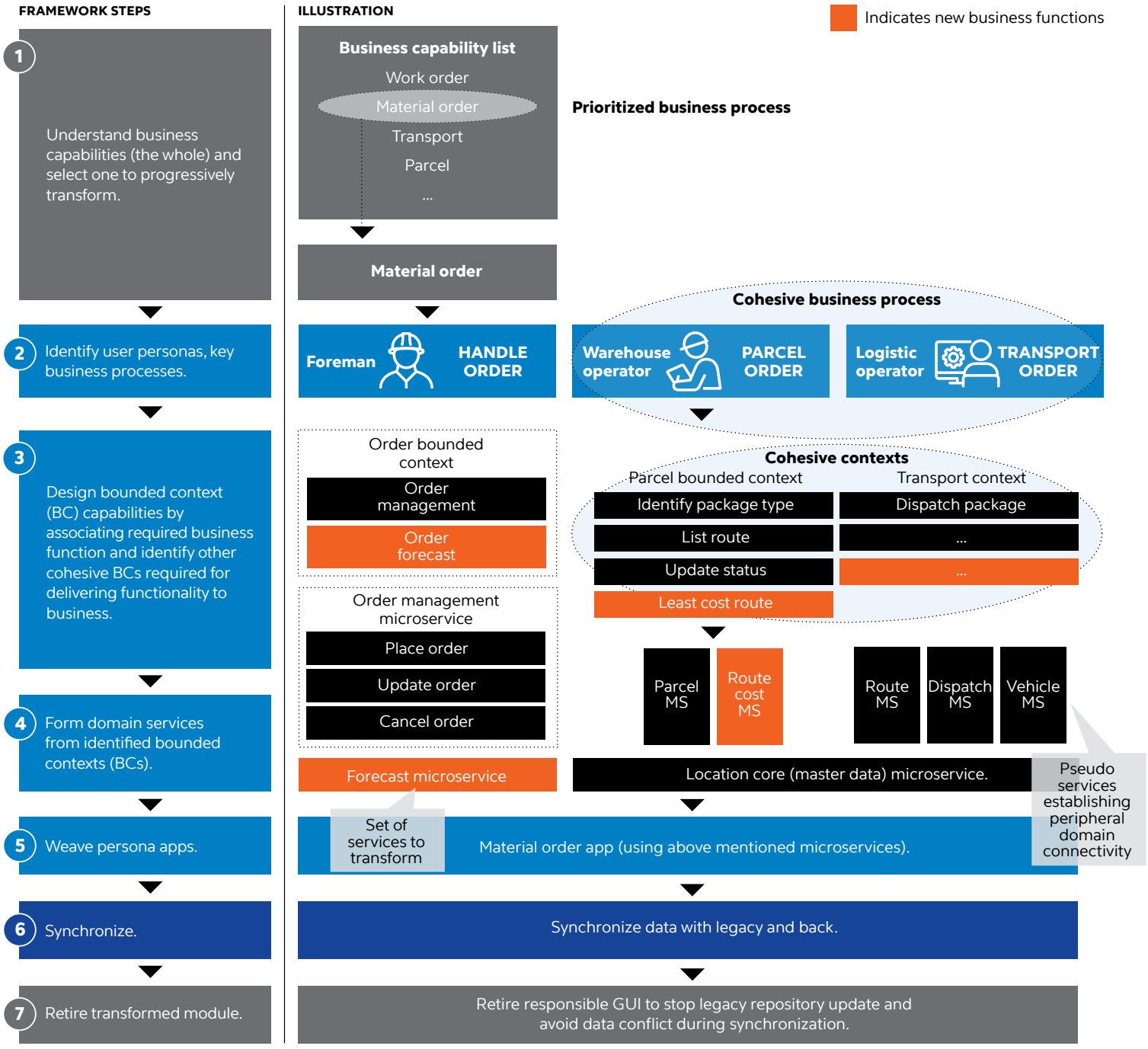


Figure 2

large and complex. Decomposition to manageable parts via bounded context also helps to simplify and yet retain the business context. The framework steps are as follows:

1. **Decompose domain.**
2. **Identify user personas.**
3. **Design bounded contexts.**
4. **Form domain services.**
5. **Weave persona apps.**
6. **Synchronize with rest of legacy.**
7. **Retire transformed domain from legacy.**

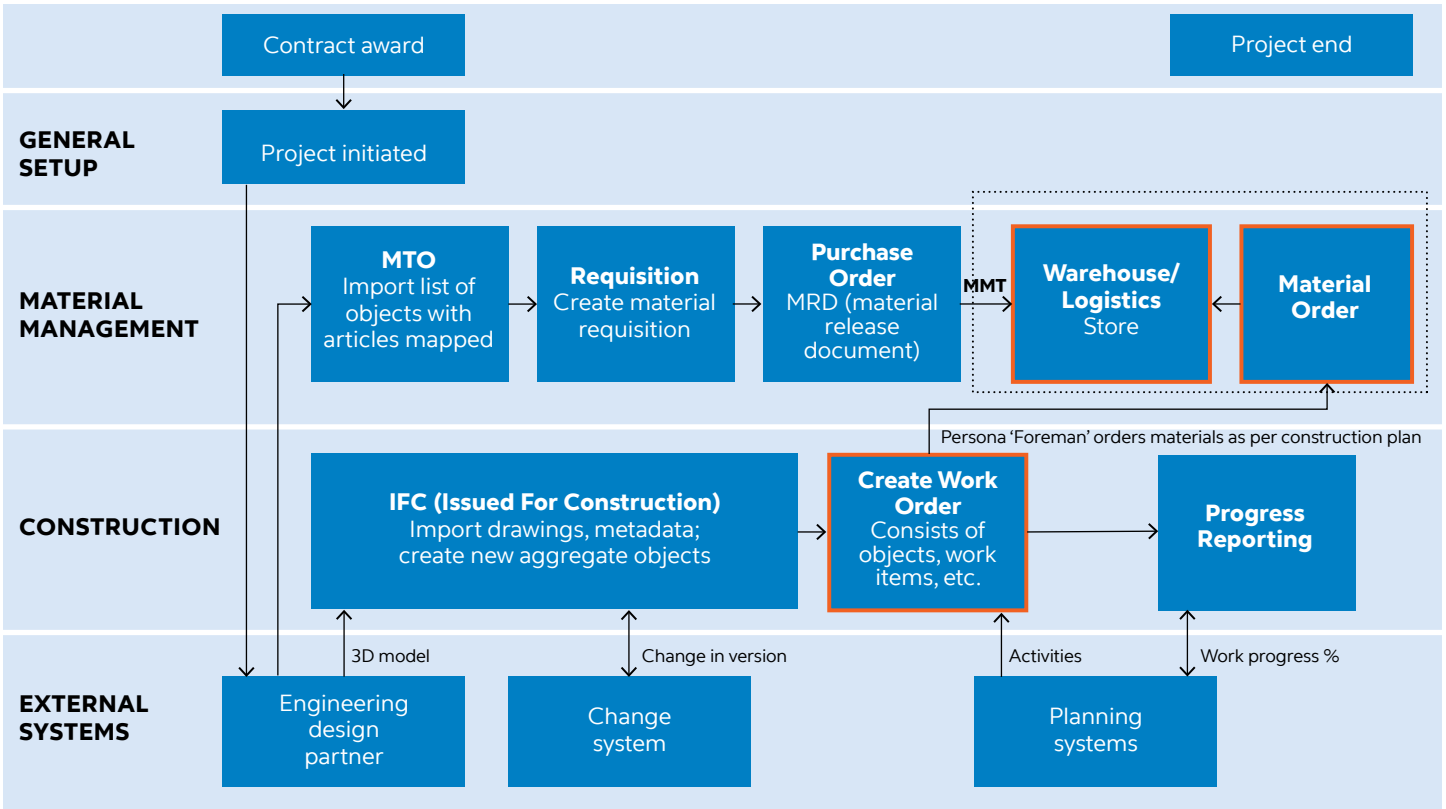
STEP 1: Decompose domain

Grouping business capabilities by domains helps to visualize legacy apps as a collection of discrete capabilities. Sourcing, financial planning and warehousing could be different domains for a manufacturing plant. The domains can be analyzed as:

- **Core:** Provides key differentiating capability for running the business.
- **Supporting:** Ancillary capabilities to support core business capabilities.
- **Foundational:** Capabilities that are foundational (e.g., enterprise integration, monitoring, sending notifications, etc.).

Gaining a holistic view of current capabilities split by domains is the first step in modernization. In Kvaerner, we considered material order and warehousing as the first set of domains to transform. Figure 3 depicts a simplified process view, demonstrating dependencies with peripheral domains.

Domains at a glance



'Articles' are objects purchased from vendors.
'Objects' are construction components made in house.
'MMT' Materials Movement Ticket.
'MTO' Materials Take Off; refers to list of materials with quantities and type.

Figure 3

The business features work order and warehouse are linked with material order. The following sections articulate how to form a persona app for material order, while maintaining the links between the transformed app and dependent modules in a legacy application.

STEP 2: Identify user personas

User personas – or representational user groups – infuse the domain model with real-world context. Applications address multiple user personas. Having a clear view of how each of the personas interact with application and for what purpose will help developers understand the domain in the dynamic usage context.

Chief personas involved in using the application are foreman, warehouse operator and logistics operator:

- **Foreman:** Deals with construction. Understands upcoming material requirements from work order and places order to warehouse. The foreman's role will be the users of a material order application. Place order will issue an entry into a new system repository and trigger changes in the legacy system so the *warehouse operator* is notified about it. (New and legacy interactions to be covered in the synchronization section in detail.)
- **Warehouse operator:** Is responsible for packaging ordered materials for construction and changing material order status to packaged. Also, this persona receives materials from vendors and maintains inventory.
- **Logistics operator:** Receives the package from the warehouse and delivers it to the foreman. Once the package is received, the foreman updates material order status as completed.

STEP 3: Design bounded contexts

A bounded context is part of the tactical phase of DDD. Our investigation revealed that a material order in Kvaerner is a combination of two bounded contexts – order management and order forecasting. We aggregated the associated domain entities and services based on identified bounded contexts. For example, order management has the following entities:

- Stock.
- Order and order item.

Forecasting order consists of:

- Order forecast.

A bounded context is the boundary within domains applying the set of domain entities. Domain services act upon entities as mentioned above. It also has dependencies on entities from peripheral domains, e.g., article, parcel, location, etc.

STEP 4: Form domain services

DDD has two phases – strategic and tactical. Strategic defines the holistic structure of a system focused on business capability, whereas tactical provides the set of design patterns useful to creating domain models. Design patterns such as entities, aggregates and domain services help design loosely coupled and cohesive microservices.

Once the bounded contexts are fully determined, it is easy to derive domain services that deliver the discrete functionality (or services) in an independent fashion that constitutes a microservices architecture. These are loosely coupled and independently deployable, providing the flexibility to evolve in an agile fashion.

Implementation of the domain service can be based on the microservice design paradigm.

Microservices in this context: Order management and order forecasting. In each bounded context, the following services are required:

- Order management:
  - Place order.
  - Update order.
  - Cancel order.
- Order forecast:
  - Forecast order.

These services are derived based on the capabilities needed, master data and integration needs.

STEP 5: Weave persona apps

In this step, we look at the user personas from Step 2 and microservices in Step 4. We then weave the microservices so the team can deliver the business capability that the user persona needs.

An attendant challenge with microservices is the proliferation of a number of such services that are developed over a period of time. In addition, from a user experience perspective, several microservices need to be tied together to deliver some meaningful information to end users or personas. To overcome these challenges, we propose the use of the back end for front end (BFF) pattern (see Figure 4).

From a user experience perspective, the persona-based application will need information across the transforming and supporting domains. Our scenario's material order and warehouse examples includes transforming domains; article and locations are supporting domains. Supporting domains

Back end for front end

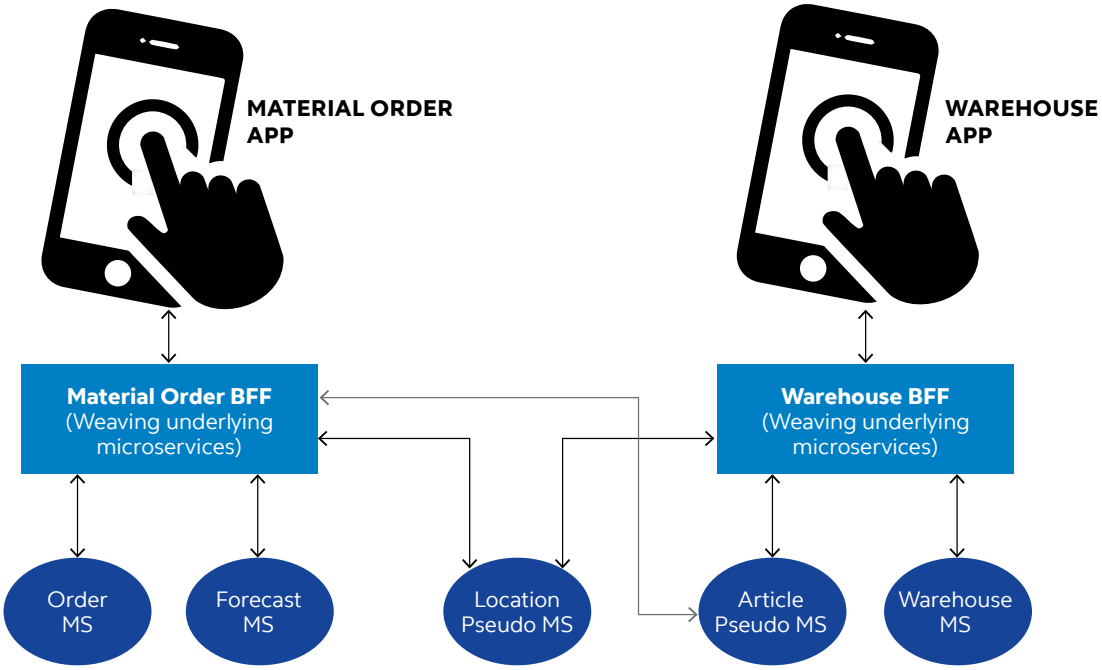


Figure 4

remain in the legacy system. To elegantly handle this, we used a pseudo-service which acts as a quick wrapper over legacy domain capabilities. These pseudo-services will eventually be converted into real microservices according to Kvaerner's business transformation plan.

STEP 6: Synchronize with rest of legacy

A transformed domain is a strangled part of the overall system. Other business capabilities already have established dependencies on transformed capabilities, and vice versa. In principle, we should not change legacy implementation to avoid disturbing other working modules and existing integrations. The following considerations are important from the synchronization aspect:

- Keep the legacy data model as is to avoid impacting other non-transforming domain modules.
- Establish synchronization between the legacy and new data model.
- Use technologies to synchronize immediately to avoid dealing with stale data and conflicts in both legacy and new systems.

STEP 7: Retire

As the modernization program proceeds and data is owned by the new application, the old data model and application features will be redundant to some degree. The corresponding legacy modules can be progressively retired. Optimally, we can retire legacy modules by abandoning or disabling their usage without trying to remove or decommission corresponding artifacts from the legacy system. This zero-touch strategy for the legacy system will reduce the risk of impacting its usage for other modules as well as reduce development costs.

Careful attention must be paid to undocumented system interfaces. The applicable regulatory requirements must be considered, as data may be needed for many years. Use of appropriate storage technology will help reduce the cost of maintaining legacy data.

Looking ahead: Progressive modernization

Enterprises should employ progressive modernization techniques to realize continuous business value. By applying proven modernization frameworks, risks can be mitigated. Aligning with business stakeholders in a continual fashion is important to ensure optimized outcomes from business and IT toward unified goals as well as establishing robust relationships across customers and partners.

Goals must be set beyond modernization, toward digital ends. Enterprises should offer new services using their increased ability for monitoring, exposing new services and data monetization. This opportunity should be leveraged to further the focus on transformation of their internal operational and developmental culture to Agile, AI and tool-driven methods.

Avoiding a predominantly server/back-end focus, enterprises should look to recast their user experience as part of all modernization initiatives. Particular attention must be paid to cyber-security[6] as microservices and decomposition often expand the number of software artifacts, thereby increasing the risk of exposure.

Endnotes

1 "Application Modernization Service Market," www.marketsandmarkets.com/Market-Reports/application-modernization-services-market-149625724.html

2 https://en.wikipedia.org/wiki/COBOL

3 Westerman, T. H. D. G. (2018, April 10). Why So Many High-Profile Digital Transformations Fail. Retrieved from https://hbr.org/2018/03/why-so-many-high-profile-digital-transformations-fail.

4 Evans, E. (2013). *Domain-driven design: Tackling complexity in the heart of software.* Upper Saddle River: Addison-Wesley.

5 "Monolith to Microservices using the Strangler Pattern", Samir Behara, September 2019. Retrieved from https://dzone.com/articles/monolith-to-microservices-using-the-strangler-patt

6 Torkura, Kennedy & Sukmana, Muhammad Ihsan Haikal & Meinel, Christoph. (2017). Integrating Continuous Security Assessments in Microservices and Cloud Native Applications. 10.1145/3147213.3147229.

Acknowledgments

The authors would like to thank Senthil Ramaswamy Sankarasubramanian, Chief Technology Officer in Cognizant's Manufacturing & Logistics and Energy & Utilities (MLEU) business unit, Narayanan Jayaratchagan, Chief Architect MLEU, and Sivakumar Paramjothi, Senior Architect MLEU, for their technical insights. A special thanks to Bjarne Notland, legacy application SME, for sharing his domain understanding and Prakash Deore Senior, Manager MLEU Consulting, for articulation of the material order business process flow used in this white paper as an illustration. We thank Kvaerner for all their encouragement, support and guidance.

About the authors

Chiranjib Chowdhury

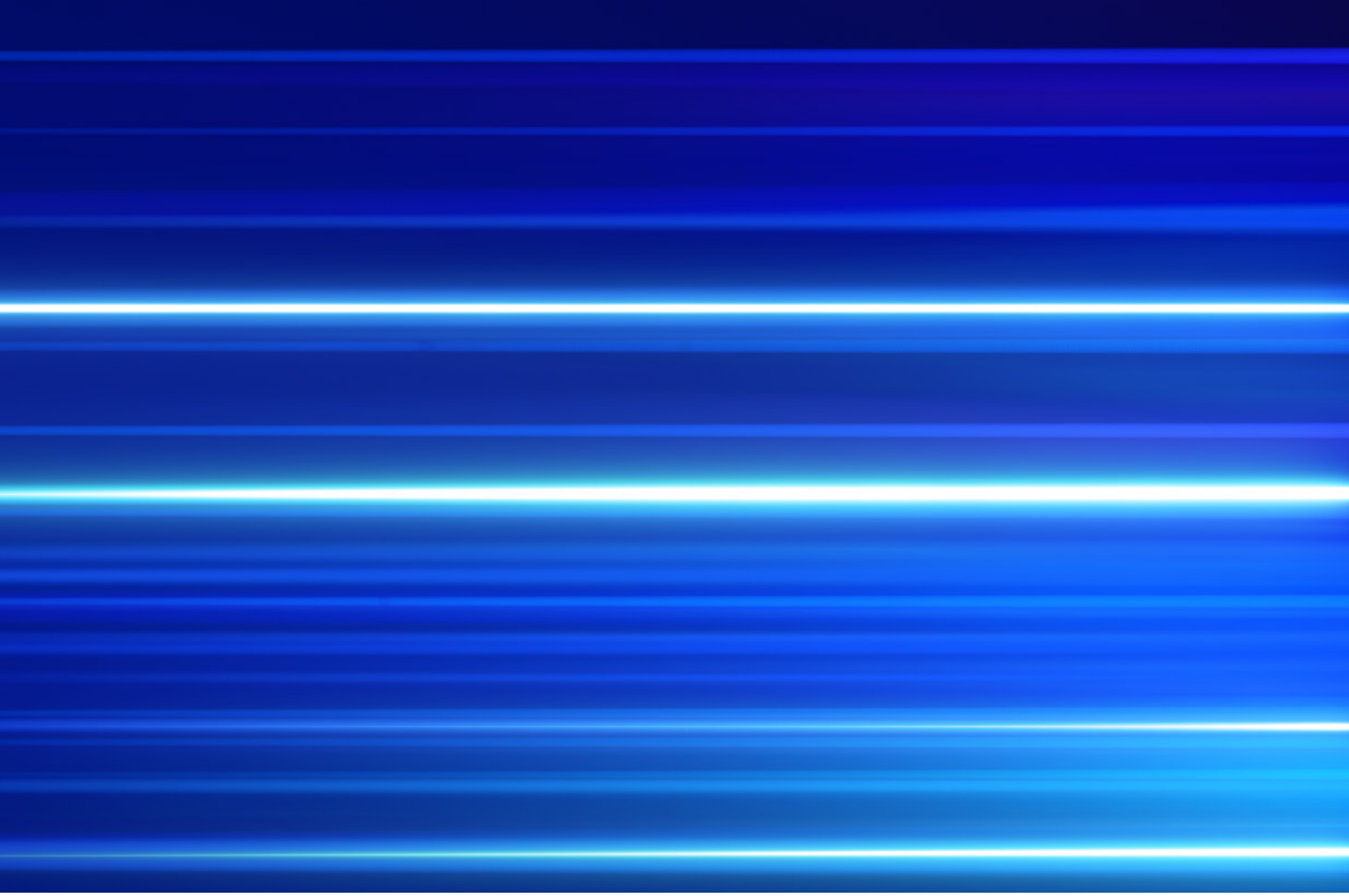
Senior Architect, Cognizant GGM CTO Organization

Chiranjib Chowdhury is a Senior Architect working as a part of Cognizant's Global Growth Market (GGM) CTO organization. He has over 18 years of experience in information technology as a solution architect, framework designer and consultant for many cutting-edge technologies across multiple customers and platforms. Chiranjib is a technology and integration expert across Windows and Linux platforms and has an active interest in enterprise security, microservices, DDD, blockchain, machine learning and IoT. He has a degree in physics from Calcutta University and a master's degree in computer science. Chiranjib can be reached at Chiranjib.Chowdhury@cognizant.com | www.linkedin.com/in/chiranjib-chowdhury.

Raghuraman Krishnamurthy

Senior Director, Cognizant Products & Resources

Raghuraman Krishnamurthy is a Senior Director within Cognizant's Products and Resources (P&R) business unit. He focuses on digital solutions and emerging technologies for clients in the consumer goods and retail business. Raghu's areas of interest include enterprise architecture, microservices, data, machine learning and IoT. He is a senior member of ACM and is the secretary of the ACM Chennai (India) Chapter. Raghu holds a master's degree from the Indian Institute of Technology, Bombay, and obtained MOOC certificates from Harvard, Wharton, Stanford and MIT. He can be reached at Raghuraman.Krishnamurthy2@cognizant.com | www.linkedin.com/in/raghuraman-krishnamurthy-ba01a94/.



About Cognizant

Cognizant (Nasdaq-100: CTSH) is one of the world's leading professional services companies, transforming clients' business, operating and technology models for the digital era. Our unique industry-based, consultative approach helps clients envision, build and run more innovative and efficient businesses. Headquartered in the U.S., Cognizant is ranked 194 on the Fortune 500 and is consistently listed among the most admired companies in the world. Learn how Cognizant helps clients lead with digital at www.cognizant.com or follow us @Cognizant.

Cognizant

World Headquarters
500 Frank W. Burr Blvd.
Teaneck, NJ 07666 USA
Phone: +1 201 801 0233
Fax: +1 201 801 0243
Toll Free: +1 888 937 3277

European Headquarters
1 Kingdom Street
Paddington Central
London W2 6BD England
Phone: +44 (0) 20 7297 7600
Fax: +44 (0) 20 7121 0102

India Operations Headquarters
#5/535 Old Mahabalipuram Road
Okkiyam Pettai, Thoraipakkam
Chennai, 600 096 India
Phone: +91 (0) 44 4209 6000
Fax: +91 (0) 44 4209 6060

APAC Headquarters
1 Changi Business Park Crescent,
Plaza 8@CBP # 07-04/05/06,
Tower A, Singapore 486025
Phone: + 65 6812 4051
Fax: + 65 6324 4051

Codex 5644