Business Orientation in Solution Architecture and Functional Solution Design

White Paper

Feb 5, 2019

Tim.Hayes@JDA.com

Contents

Revision History

Version	Date	Author	Description
0.0	Jan 29, 2018	Tim Hayes	Initial draft
1.0	Feb 4, 2018	Tim Hayes	Revision, corrected typos and improved concept flow
2.0	Feb 5, 2018	Tim Hayes	Enhanced detail in Business Metrics section, added End Notes, and major enhancements to Technical and Governance sections
2.1	Feb 5, 2018	Tim Hayes	Enhanced detail in End Note 3.c
2.2	Feb 6, 2018	Tim Hayes	Minor word-smithing

Abstract

In designing a functional software solution architecture for a business, it is important for the Solution Architect to understand the business context and to design the software solution to maximize the value realized by the business from the implementation. In most environments, it is not enough to get the application installed, configured, and working; the Solution Architect must optimally configure the solution to deliver maximum business value, and ensure that the system continues to do so over time. This requires understanding the business goals, knowing how best to configure the software to achieve business objectives, and guiding the business in establishing workflows and processes which ensure continued success.

Introduction

Functional Solution Design can be described as the activity of a Solution Architect (SA) in positioning and configuring a software application to meet a business need or solve a business problem. Solving a business problem well requires a thorough knowledge of the problem itself, the ability to translate the business requirements into appropriate workflows, functional and technical requirements, and integrating the software application into the business environment so as to generate maximal business value. This is easier said than done, and requires much more than a technical knowledge of the software application.

Simply asking the customer what they want the software to do, how they want it to work, and then trying to make the software satisfy customer expectations as much as possible, may seem like a reasonable approach but it is often problematic. The customer does not generally understand the software application in detail, or the best practice methodology for deploying and using the software in their particular business context, or even their own business requirements very well. The customer needs guidance in this process, and this is one of the key responsibilities of the SA.

The following are general principles for Solution Architects to follow in the context of functional software solution design and implementation, to help them guide their customers through the implementation process and ensure success.

Business Metrics

Firstly, to be effective in any software implementation, the SA needs to fully understand the relevant business goals and how the software solution is intended to help the business better achieve these goals. These goals are represented by *business metrics*: quantifiable measures or key performance indicators (KPI) used to track and assess the behavior of specific areas or activities of the business. Examples of common supply chain business metrics are: *Forecast Accuracy*, *Perfect Order Fulfillment*, and *Inventory Turns*.

In the context of tight timelines and limited resources, it is tempting to skip over this basic step of identifying and defining the key business metrics to be impacted by a software implementation, assuming they are clearly understood and will be addressed -- since they were (hopefully) a primary focus of the sales cycle. Yet this lack of explicit focus on business value during the software implementation can result in inefficient solution design, and ultimately in failure to fully realize the benefit expected from the software.

Ideally, relevant business metrics should be defined in detail early in the implementation process, including: verbal description and business context(s), mathematical formulas, trigger points[1], corrective workflows[2], metric interdependencies[3], refresh frequencies, reporting requirements, and data sources. The SA should be prepared to engage with, encourage and guide the customer in extending and enhancing related business processes and infrastructure capabilities as needed. This ensures that the business can effectively measure how business processes are being executed in order to drive the desired business value using the software solution.[4]

Starting an engagement by identifying key business metrics that are expected to be impacted by the implementation, defining these KPIs clearly and positioning each solution feature to enable the business to most effectively optimize them, and then baselining these metrics just prior to enabling the software solution, is critical for measuring and validating the value generated through the software deployment.

Business Requirements

Once the SA understands the *business metrics* which are expected to be impacted by the software implementation, it is essential that the SA identify specific *business requirements* which must be satisfied to optimize these metrics and achieve the goals of the enterprise. Mutual understanding of this particular term, *business requirement*, between the SA and the customer is key to project success.

A business comprises people working together to achieve business goals. Their activities impact business metrics, so each person must act in a particular way to optimize these metrics. These *required* activities, which people need to perform to achieve the goals of the *business*, are the *business requirements*. For example: *Generate Demand Forecast*, *Resolve Late Orders*, and *Set Inventory Targets*.

Once this is understood – what people need to do to optimize business metrics in a given business context – the SA can design user workflows using their expert understanding of the software solution and industry best practices to enable people to accomplish these business objectives using the enabling solution's capabilities in the most optimal manner.

Functional Requirements

Once the SA understands the *business requirements* and required workflows in their proper business context, the SA uses their expert knowledge of the software application to determine the *functional requirements*: the particular features and capabilities required in the solution design to align both human and software capabilities to achieve the business goals. Defining functional requirements in the context of optimized business process workflows enables the SA to close all solution gaps while maintaining focus on value delivery.

This is where the true business value of employee skill sets and the application's functional capabilities are understood: these capabilities enable people to complete business processes efficiently and effectively to satisfy their business requirements. For example, a demand planning system must enable the generation of a demand forecast appropriate for the given business context, including market analysis, stakeholder consensus, and the effects of promotions and product life cycle phases. A master planning solution may be required to provide an interactive *Demand-Supply Workbench* to enable a planner to efficiently execute the steps of a *Resolve Late Orders* business process. To enable an inventory planner to efficiently set inventory targets, an automated solver must consider various types of holding costs, expected forecast error and stockout penalties by supply chain stage and segment type.

Defining functional requirements in the context of optimized business processes and business requirements clearly tied to business metrics helps ensure that enabling technologies are configured and implemented to provide maximum value to the business, and facilitates change management by providing a value-based context to promote solution adoption by the business organization. This is how the software solution is designed to provide maximal value for the business.

Technical Requirements

Once the SA clearly defines the *functional requirements* in a meaningful business context, it is natural then to identify the *technical requirements*, the enabling system infrastructures, integration points and capabilities, and the data sources required to enable business and functional requirements. Defining technical requirements in the context of optimized functional requirements and business processes enables the SA to implement the software in a manner that delivers maximum business value.

This is where report configurations, interface designs, workflow frequencies, on-demand workflows, staging and orphan data requirements, system performance and data maintenance capabilities are clearly understood to be of value to a business, and designed and documented with this value in mind. For example, focusing on the purpose of a *Late Orders Report*, and how it would best enable a master planner to take effective corrective actions in a timely manner as he/she executes the steps of the *Resolve Late Orders* business process, drives design decisions about report content and layout, filtering and messaging capability, graphical inventory projections, data sourcing, system performance, refresh timing and frequency to maximize its value to the business.

A significant concept in any system's technical infrastructure relates to enabling corrections and enhancements to the deployed software solution without destabilizing the live, working environment. This generally requires separate, compatible environments for development, and for testing to ensure the quality and stability of all fixes and enhancements.

Defining technical requirements in the context of optimized functional requirements is precisely how the software solution is implemented to generate optimal value for the business.

Complexity

In the context of any software implementation, it is common for the customer to become enamored with certain nice-to-have options available in the software. In such contexts, the old acronym KISS (*Keep It Simple Stupid!*) is as appropriate as it is in many other areas of life: just because a feature is available in the software does not mean that it should be deployed.

As an analogy, think of building a racecar: there are a wide variety of fascinating, clever little gadgets which can make a racecar more fun to drive and show off to friends. But each feature implies added cost, an additional integration and failure point, as well as extra weight to carry around the racetrack, consuming fuel and impacting the balance and integrity of the racing machine. If the objective is to safely, fairly and consistently cross the finish line in the fastest possible time, everything about the car should be carefully and intentionally designed to support this objective. Every component should fulfill a key purpose in the overall design of the car, being both necessary and sufficient; if any aspect of the design can be omitted or simplified without impacting the final outcome then the design is sub-optimal.

So it is with software solution design: just because a feature is available, and provides graphical appeal or otherwise drew smiles during the sales cycle, does not mean that it belongs in the solution. If the business value of a particular non-default configuration or option is not clearly understood and documented, then it is the responsibility of the SA to push back on the requirement and ensure that the customer understands the cost/benefit analysis of including it, and is able to demonstrate that the expected benefit offsets the expected cost in terms of development time and expense, potential risk due to possible system dynamics or untested business scenarios, and overall system performance and maintenance. The SA must repeatedly work through this design analysis process with the customer, analyzing every aspect of the solution, to ensure optimal solution design.

In particular, the SA must remain cognizant of the fact that any custom development work needs to be supported and maintained long-term. The typical software solution provider will not support custom development unless it can be profitably productized, so support responsibility for custom work often falls on the customer, which likely does not have the necessary expertise and/or resources to develop and maintain such solutions, which is why they are asking the available solution team to develop it in the first place. Yet consenting to deploy custom solutions without properly equipping the customer to fully support them is a recipe for failure, eventually resulting in a dissatisfied customer.

Data Matters

The importance of data integrity, completeness and cleanliness can hardly be overstated in the context of computer-based software solutions. The *Garbage In Garbage Out* (GIGO) principle applies here with a vengeance. The business benefit derived from a software implementation generally depends entirely upon the integrity of the data being fed into the software application(s), and this is not always self-evident to the business users or managers.

To ensure that data quality is consistently maintained, the customer must be advised and guided in the development of business metrics and related business processes designed to detect problems in system data and to correct these issues promptly. This includes transactional data, master data, and all system parameters; such values must be adjusted as needed to reflect reality consistently and accurately. The SA is instrumental in aligning corporate culture with this data validation requirement, ensuring alignment in both the management and user communities, in order to ensure long-term success.

Governance

Each software solution requires a governance process which ensures that the solution remains viable and optimally configured over time. Software systems which model real-world scenarios tend to become obsolete as system requirements change, so these systems need to be maintained, repaired and enhanced to continue to function according to their purpose. Therefore, the software solution should be inspected at regular intervals to ensure that it is properly aligned with the current business environment and that it is performing optimally.

Ensuring optimal performance implies the need to develop and implement *operational metrics* which monitor process execution and the quality of business process deliverables. These metrics are as essential for delivering value long-term as the higher-level business metrics impacted by the solution.

Further, protocols must also be designed and implemented for reporting, escalating, and resolving all issues which are encountered during business workflow execution related to using the software.

Continuous Improvement

To be complete, any solution implementation must be designed to enable *kaizen*: continuous improvement. As business requirements change, or are better understood, or more efficient ways of working are discovered, the business should adapt, reconfigure and enhance the software solution, reporting capabilities, supporting technical systems and infrastructure as needed. Continually improving business workflows and automated processes ensures that optimal value is always being derived from the software solution. The SA may need to advise the customer of this requirement, and guide in the design and implementation of relevant business processes and methodologies in order to be fully successful. Such processes often comprise simple, systematic feedback loops built into solution governance business processes which ensure that proper business workflows are being followed and encourage innovation in improving efficiency and reducing waste.

Summary and Conclusion

Effective solution design requires much more than a deep technical knowledge of the software application being implemented. It begins with understanding business goals and business requirements, and is accomplished by designing business processes and configuring solution workflows which work together to optimally leverage enabling technological capabilities so as to maximize value for a business. Guiding the customer in designing business processes which focus on maintaining optimal value delivery through effective enablement and continuous improvement of coordinated business activities is inherent in good solution design, and is the key which ultimately drives business value.

End Notes

1. A ***Trigger Point*** is a threshold value specifying a boundary condition for a metric. Then the metric value passes this threshold in a sub-optimal trend (i.e. the system *violates* the trigger point), then corrective workflows are executed to reverse the trend and restore optimal system behavior.
2. A ***Corrective Workflow*** is a business process which the business role responsible for correcting system performance follows when a business metric trigger point is violated.
3. Metric interdependencies can be classified as follows:
    a. ***Parent*** metrics are higher level business metrics which are directly impacted by a given metric. For example: COGS is typically a parent metric of Inventory Turns.
    b. ***Child*** metrics are lower level business metrics which directly impact a given metric. For example: COGS is typically a child metric of ROI.
    c. ***Complimentary*** metrics are positively correlated business metrics which tend to improve when a given metric improves, and also to deteriorate when it deteriorates. For example: Forecast Accuracy and Safety Stock Carrying Cost are typically complimentary metrics.
    d. ***Conflicting*** metrics are negatively correlated business metrics which tend to deteriorate when a given metric improves, and also to improve when it deteriorates. For example: Expedited Freight and Inventory Turns are typically conflicting metrics.
4. For more information on KPI methodology see the white paper KPI Methodology (in SharePoint under Global Strategic Services : Customer Engagement : Implementation)