Aligning Organizational Goals to Guidewire Products Out-of-the-Box (OOTB)

Aligning Organizational Goals to Guidewire Products Out-of-the-Box (OOTB)

Guidewire software solutions offer users flexibility that enables them to extend base product features or design completely new features strategic to the way they do business. However, users must balance flexibility and the need to stay within their original timeline and budget. They should focus on the strength of the core system – which no doubt was a significant factor in their original product selection decision.

So, how much time and money should Guidewire users invest in changing the product? What new features should they add and how much configuration should they take on in the first versus subsequent releases? Many users have told Guidewire that they do not want to over configure the application and want a way to track the amount of configuration that occurs in the development cycle. As a result, Guidewire created Product Alignment to help users measure how far they are diverging from the base product as they make project scope and design decisions. Setting a Product Alignment goal along with other program goals will help keep a project on track and provide a way to communicate this to project sponsors.

Product Alignment is part of another methodology concept Guidewire calls Value Alignment. The main goal of these concepts is to help users make the right decisions about what features they should configure and how much configuration they should perform. In the end, users should invest in configurations that are:

1. Foundational in the OOTB application and must be done in order for it to perform properly for the user;
2. Providing business value to the end user and meet the objectives set in the user's business case; and,
3. Required to meet a regulatory or statutory requirement.

Leveraging base product functionality

It is not uncommon for users to embark on the implementation project with the expectation that the new Guidewire application will provide an enhanced user experience but will function essentially the same way as the legacy system that is being replaced. However, replicating how the old system works in the new system will not result in the best return on investment. In order to realize the full benefit of the investment, the user should view the new application as a tool that will support a workflow with improved efficiency and effectiveness. However, the user will not realize the benefits until the system is in production. Accordingly, it will need to take a measured approach to balance the value of additional configuration against the delay in realizing the rewards. This is the goal of Guidewire's Value Alignment philosophy: an initial release with less configuration combined with a strategy of multiple releases that will support quicker initial implementation, foster a continuous cycle of improvement, and simplify future product upgrades.

A carrier that uses a notes-heavy system with a diary process for messaging purposes is a prime example of a user that should adopt a new way of working. The development team will need to convince the stakeholders to embrace a system that relies on discrete data elements, rules, and more exception-based processing, instead of building the limitations of the legacy processes into the new system. It is important for the development team to demonstrate to the business the depth and flexibility of the application OOTB, the implementation methodology, and how adapting to the built in functionality will add value and reduce cost in the long run – and all without sacrificing the user experience.

Guidewire Product Alignment

Product Alignment allows users to measure how far their implementations diverge from the base product. It follows a numeric scale from 0 to 4; in which 0 indicates no alignment between a new requirement and the base product and 4 indicates full alignment.

The inception phase of the project is where scope and a business case are defined in order to create a deliverables plan for implementation during the project development phase. One of the project goals the carrier should set during the inception phase is Product Alignment, which should be above 3 if the carrier wants to remain as close as possible to the base product configuration. The carrier may set a lower number if it needs to accommodate unique business cases that are outside the industry norm. A set of related functional requirements is a "story," and each story should be evaluated and receive an appropriate Product Alignment value. If a story does not meet the alignment goal, then it should be a low priority or removed from the project scope altogether.[1] It will be important to reassess Product Alignment during the development phase in order to compare the actual alignment to the inception phase baseline.

Even if a carrier decides to implement a Guidewire product as close to OOTB as possible, some configuration will have to take place to configure the product to the carrier's specific needs. Base product features which Guidewire considers foundational may take significant time and effort to configure but would be considered fully aligned and receive a Product Alignment value of 4. For example, during a PolicyCenter implementation, configuration of the product model, underwriting rules, and policy holds would equal a full alignment, as would delinquency invoicing in BillingCenter. Defining and configuring the LOB structure in ClaimCenter and configuration of users and groups, and generating sample data would be considered fully aligned regardless of the Guidewire application. Moreover, even though accelerators are not part of base product functionality, their implementation would merit a Product Alignment value of 4 as long as the customer has not significantly changed them. Any configuration beyond that should closely relate to a specific benefit in the business case.

Low Product Alignment should not necessarily be viewed in a negative light. The lower the Product Alignment value, the more impactful the business benefit must be in order to support it. It is certainly not uncommon to find a Product Alignment value approaching zero (near custom work) that has a business value significant enough to adequately support it. (However, the selection of a packaged solution would suggest that a relatively small percentage of stories should possess a low alignment value.)

Finally, setting the Product Alignment goal during the inception phase and tracking the actual measure against the set target or business case will help keep the project on track and should foster more realistic scoping discussions and decisions throughout the entire project.

Prioritizing requirements

It is inevitable that new requirements will be introduced during the project development phase, especially with Agile methodology (when the carrier is learning the product throughout development). In order to accomplish the project's planned objectives, it will be important to put on hold any new requirements that do not correlate to the planned business case and therefore are out of scope. A clearly defined business case will help the team analyze the new requirements and determine if the application already has built-in functionality that is in line with the business case. It is up to the development team to make sure that the carrier, in order to realize the most value from the project, utilizes the application's core functionality. The development team should help distinguish between must haves and nice to haves. They can revisit the latter at

[1] A Guidewire consultant will help the customer analyse each story and assign an appropriate numeric value.

the end of the development phase (if there are sufficient resources available to work on them without jeopardizing the project deadlines or the ability to upgrade in the future). Otherwise, they can schedule future releases.

Sometimes, because of time and resource constraints, the must haves also require prioritization. If there is a work-around available that enables completion of the claim, billing, or policy lifecycle, then it can at least temporarily replace these requirements, which should have the lowest priority and be developed at the end of the project (if time and resources permit) or perhaps in a subsequent release.

Throughout a project, developers should be willing to challenge the business on requirements that may pose risks and inefficiencies. Likewise, the stakeholders should always be prepared to adequately defend the business case. As product stewards, the developers are responsible for steering the project towards effective requirement prioritization.

Communication and project management

In order to keep the project on track and everyone on the same page, effective communication both within the implementation team and with the business is especially critical early on in the development phase and should be an area of emphasis both during the inception and throughout the project. For example, configuration developers must be aware of in-process development work across the workstream. If not, the same functionality requirements may overlap and end up on two different functional requirement documents that address different parts of the application. As a result, multiple developers could end up spending considerable time and effort duplicating the functionality that already exists, which could easily lead to product inconsistencies.

A project also can suffer delays when there is a lack of alignment and clear communication on the deployment schedule between the configuration and integration teams. Most integration requirements have interdependencies with configuration. For instance, if the integration team is currently working on ISO integration point and the configuration team is scheduled to work on ISO two months later, then the integration team will not be able to complete its work. The testing team also will experience delays because it will not be able to finish testing until both the integration and configuration pieces are complete. If both teams communicate during the planning phase of the project, then they can avoid these pitfalls.

Untimely, requirement gathering also may throw off alignment between teams. It is imperative for subject matter specialists and product owners to have a clear understanding of the progression of a claim within the new system. Project leadership must prioritize requirement gathering to match the configuration and integration teams' schedule of deliverables.

Guidewire coding best practice standards

Guidewire provides documentation for best practice coding standards. This includes naming conventions, coding standards, documentation, and performance. Carefully guard against the temptation to stray from these standards. Following these standards during the development phase of the project will definitely increase the value of the final product in the long run. Adherence will reduce the amount of time needed for application maintenance and future upgrades.

Best practices include:

- Configuration developers always should add a suffix (_Ext) to the names of anything they add to the data-model. This practice will avoid conflicts with base entities that Guidewire may add or change in the future. Developers should the suffix to the names of new entities, fields, and typelists, and display names within display properties.
- Developers should add a suffix to new PCF files in order to avoid conflicts during the upgrade process. This practice also will help make the upgrade process more efficient because there fewer files will need evaluation.
- Developers should adopt a naming convention for rules in order to make debugging easier. The name of a rule should contain an identifier of the parent rule set, a numeric value, and a description. It is important to keep the rule names short yet meaningful, since the file path and the rule name together must not exceed 255 characters.
- It is important to apply defensive programming in methods and functions. Developers can avoid most null pointer and out of bound exceptions by using exception handling, such as try/catch blocks, and checking for the boundary conditions in the for loops. Using spaces, empty lines and indentations properly will make the Gosu code easier to understand and maintain.
- Developers should use comments inside programs for better code readability and understanding. Block comments are used at the beginning of the classes and before each method. A block comment should contain the description of the class or method, list of input parameters, output being returned (if applicable) and the author. Single-line, trailing and end-of-line comments should describe specific functionality within a method.
- For better performance, developers should avoid excessive use of postOnChange inside PCF files. Every postOnChange on the page requires a server call, so a complex page with multiple postOnChanges will significantly reduce performance. If possible, developers should use reflection.

Managing change-order requests

Change-order requests should be prioritized to deliver the greatest benefit to the carrier. The development team should engage the project sponsors to facilitate request prioritization. Just like the requirements that surface during the development phase, change-order requests should be prioritized as must-haves or nice to haves, and scheduled for development accordingly. Some of the change-order requests may not be necessary because later stages of development will address any related, perceived gaps. Developers will need to analyze these requests to avoid future duplication of effort.

In conclusion

By conforming to the OOTB application, following the Guidewire best practice standards, maintaining effective communication within the development team, and prioritizing new requirements, the development team can keep the project on time and within budget, help to deliver a sustainable and extendable product, and ensure efficient realization of business value.

To have a deeper discussion about aligning organizational goals with Guidewire products OOTB, please contact:

Russ Riggen
Principal, Advisory Services
+1 630 561 3626
russell.riggen@us.pwc.com

Ricky Raisinghani
Director, Advisory Services
+1 312 298 2695
ricky.raisinghani@us.pwc.com

Yury Purim
Manager, Advisory Services
+1 312 298 3092
yury.purim@us.pwc.com

For more information about PwC's Guidewire capabilities, please contact the authors or the following:

Marc Gallo
Principal, Advisory Services
+1 510 219 6609
marc.gallo@us.pwc.com

Dirk George
Principal, Advisory Services
+1 610 316 3290
dirk.d.george@us.pwc.com

Imran Ilyas
Principal, Advisory Services
+1 630 699 0657
imran.ilyas@us.pwc.com

Paul McDonnell
Insurance Advisory Co-leader
+1 203 470 1772
paul.h.mcdonnell@us.pwc.com

Rajcan Surface
Director, Advisory Services
+1 312 298 3143
rajcan.p.surface@us.pwc.com