Boomi Integration Services
KitePipe

Phone: 844.232.2227
E-mail: boomihelp@kitepipe.com
Web: Kitepipe.com

Successful Boomi Integration Projects

How to Structure and Execute a Boomi Integration Project, with Framework, Deliverables, Risks, and Estimating Tips

kitepipe™

Premise and Audience

Audience – Beginning or experienced managers who are executing Boomi integration projects, and would benefit from a framework with which to structure the project, and specific tips and techniques.

Services team managers that need a better way to structure and estimate an integration services project, especially one that uses the Boomi platform.
Project Managers who want to understand the scope of work involved in a Boomi integration project, and where the risk and scope control points.

Anyone who wants a good feel for what a Boomi project is like before setting out on one.

Premise – the reader has development experience, some understanding of data structures, and some experience with delivery of software development projects. With this background, you can immediately apply the concepts and lessons in this whitepaper.

Also – This model works best for high-value business transactions within a single organization. Many other integration scenarios – Pub/sub, EDI, etc, are partial matches to the framework. Fortunately, high-value business transactions is what we mostly do at Kitepipe, and where the integration team can deliver significant value to the organization.

CONTENTS

Introduction

Why talk about project framework and deliverables? It's the highest value thing I can talk about to the Boomi Integration community. Project-level mistakes are the big mistakes. This document gives you a checklist of things to look for in structuring your project.

At Kitepipe, we do A LOT of Boomi projects. In fact, that's all we do – for companies large and small, new tech and old-line, Biotech, Software, and Logistics. Last year (2016) we logged hours against more than 60 individual Boomi integration projects, and I've had the opportunity to pay close attention to what is common across many projects, and where things go off the rails.

In this updated version of the "Sprints" I've captured learnings from the last three eventful years at Kitepipe. By far the biggest source of project problems were scope issues – basically disagreement, unawareness or changes of the scope of the effort that eventually was covered by the project.

Often the seeds of big project problems were baked into the assumptions of the initial estimate. I've paid many times for my own estimating mistakes, and developed a keen sense of scope and risk in the process. No process will protect you from "changes" in mid-stream, but I've found that applying a detailed framework early in the project can surface those "changes" earlier, in time to do something about them.
By describing a framework with which to understand the scope of a Boomi Integration project, I seek to give you a set of tools with which to understand the scope of your project, develop better estimates, and manage risk along the way.

Larry Cone
Chief Architect, Kitepipe
July, 2017

The Six Critical Sprints

So, what are the Critical Points in a Boomi integration project? What is a good framework with which to understand the scope of a cloud integration project?

We use a services model for Boomi projects with six sprints, and each may be larger or smaller depending on the project.

The Sprints are:

1. Basic Connectivity: In the first sprint, you develop basic Boomi processes that demonstrate connectivity to the integration endpoints – this helps manage your project risk.

2. Data Transformation: In the second sprint, you work on the data transformations and lookups– the meat and potatoes of the project.

3. Business Process Support: In the third sprint, the focus is on supporting the larger business process – Alerts, notifications, post-backs, and migration issues.

4. Transaction Quality: In the fourth sprint, having achieved data transformation, focus on validation and transaction quality checks within the context of the overall business process.

5. UAT and Go Live: The fifth sprint is often part of a larger project, and supports overall testing and go live goals.

6. Clean-up and Tech Transfer: The last sprint is where you clean up, document, and hand-off.

I'll describe each one of these in some detail, and discuss reasons, goals, deliverables, potential risks, and estimating guidelines.
But, if you read no farther, you already have a valuable framework for managing scope and risk in a project.

Basic Connectivity:

In the first sprint, you develop basic Boomi processes that demonstrate connectivity to the integration endpoints.

This can take a few hours for simple endpoints with existing connectors, or a month where custom development is needed to drive a complex API.

Goals:
To demonstrate basic connectivity to the endpoints.

Deliverables:
-Develop a simple process that shows basic read write activity to the endpoints, or a listener that accepts calls from a source system or web hook.
-Confirm that connectors, sandbox environments, account logins, and connectivity is in place to pass records from one endpoint to another
-Prototype and demonstrate the technique that the process will use to select records.

Risks:
-Are connectors available? For a RESTful API without a pre-built connector, that is implemented with security signatures that require custom coding, it can take several weeks to push your first record from one endpoint to another.
-Local data base connections can require local atom ionstallation, networking, account creation, and permissions updates.
-Some EDI connections such as AS2 connectors require opening of ports on firewalls. It is good to identify these issues early as there can be lead times involved.
-Is the design of the record selection or handshaking appropriate? Simple schemes rely on last updated fields, so make sure these are available on the records that you plan to use. More complex handshakes use custom fields that flag records for interface, and are then cleared by the process. Validate the handshake or selection design, and get any custom fields in place.
-This is a good time to verify the number of Objects that the integration will touch on either end, as this is a key measure of complexity.

Estimation Guidelines:
-Key factor here is the availability of connectors for the endpoints, and local atoms, if not already installed.
-Other factors include the complexity of the handshake scheme, Batch vs. real-time transactions, the number of objects handled by the process, and any connectivity components that are out of your direct control, like firewalls, Atom servers, or third party EDI infrastructure

Data Transformation:

In the second sprint, you work on data transformations and lookups. The rest of the project should be far enough along to have significant amounts of test data available, and in a series of iterations you push over test data, identify quality issues (missing or invalid data) and add logic, lookups, and transformations.

This is also where you handle differences in instantiation – that is, for example, when an order comes in as one document, but must inserted into the target as a series of line items. Or, when shipping instructions come in as multiple line fields (looking at you, SAP), but must be consolidated into a single field in the target.

This is the bulk of the Boomi development effort, and is the place where you design and develop the basic structure of the Boomi processes. Most of this work is done in the Boomi Build environment with small data volumes. The deliverable from this sprint is a set of basic Boomi Processes that are unit tested, and ready for additional development.

At this point you can say that you are "Moving data end-to-end", which is comforting to everybody.

Goals:
To build out the source queries, listeners, transformations, and lookups required to accept data from the source, transform to the format of the target, handle differences in instantiation, and post transactions to the target (or publish to multiple targets).

Deliverables:
-A set of Processes that move transactions between the source system and the target system.
-The Maps, scripts, profiles, and connector objects required to drive the processes
-A set of test data sufficient to exercise all of the branches and conditions in the processes
-An executed set of unit tests, with formal or informal documentation, that demonstrates that the process moves data "end to end" and applies selection and transformation logic.

Risks:
-Is there good documentation that covers required fields, date formats, record dependencies, etc?
-Are the Objects accessible as planned? Some RESTful APIs may require three calls to change a relationship (GET the old one, DELETE it, PUT the new one)
-Are there complex transformations of EDI or XML documents? Pulling order or shipment items out of a complex EDI or XML document can be tricky
-Have you accounted for all status combinations? Things like Adds, Updates, Cancellations, and Deletes may require separate handling and separate processes

Estimation Guidelines:
-Number of Processes, number of connectors in the processes, number of Maps, number of Profiles.
-Complex EDI or XML profiles can require fussy trial and error to transform from one format to another

Business Process Support:

In the third sprint, you work on fitting the integration process (a piece of implementation) into the overall business process. This is an area where we have had some notable misses and surprises, and where we deliver significant business value.

Business process support means this: how does the Boomi integration process fit into the overall business process, and what features and functions are required to make the fit seamless, and to drive business value?

Here are some examples of integration features, and how they support the business process:

Completion notifications – when users want to know that the transaction was successfully processed

Error/exception handing – someone needs to know when to fix data problems

Status notification – when a multi-step process in in progress, and users need to know

Approvals – when an approval is needed to finish the transaction

Handshakes – when a complex multi-state integration must be managed from the end-points, and the integration be kept stateless

Recovery – when transactions must be recovered due to catastrophic integration failure

Awareness – prevent the integration from being a “black box” in the users’ mind by providing notifications and logging

Migration – one-time process features needed to process initial loads of data

Certainly, there are many integration scenarios that don't require these types of features: Extract, Transform and Load into data marts is about volume of completed transactions, and status and quality are not issues. Most Pub/Sub scenarios don't care about the completion status, or recovery/retry.

But, increasingly, integration processes are called upon to handle live data not yet complete from a business process standpoint. In these cases, the Architect or Designer must understand the business process context, and build in features that support the overall business process.

In an ideal, waterfall world, you would figure out the details of how the integration fits onto the overall business process up front, in a requirements document, and build any implementation components in during the build.

One of our significant discoveries in building Boomi integrations is that this discussion can be left to later in the project. This may sound counter-intuitive, but it works.

Here are some reasons to defer:

-You can quickly get started on the more technical parts of the system (connectivity, mapping) while you gain an understanding of the business context

-Looking at the data and the endpoints will inform you about what questions to ask. Key fields are "status" type fields.

-Often, no-one on the business side has a full view of the overall process – teams tend to concentrate on their platform and process. The Sales guys in Salesforce live in a country very distant from the Finance team in Netsuite. So the more context you have, the better the questions you can ask to elicit issues and needs.

-Often, no one has thought about "migration issues" – what happens when we first turn this on. You, the architect, know that integrations are designed to work in a steady state, with endpoints synched, and only deltas being passed. You, the architect know that often one-time configurations are needed to handle the initialization. You can ask better questions later in the project.

So, you can get started with the configuration tasks, learn something about the data, and can ask good questions and design a complete solution. In my opinion, this phase is the difference between technicians building an IT solution, and consultants helping to solve business process problems. This is the phase where you build in features that are visible to users and management, and create a strong perception of value in the management team for the solution.

Being able to handle business process support in this way is one of the reasons that Dell-Boomi Atomsphere is the best integration platform available.

At Kitepipe, we have built our teams and tuned our processes to maximize the business value of our integration projects. Our developers are highly trained, experienced, and productive at building Boomi processes. They are also experienced IT Consultants who can ask the right questions about the overall business process. With this combination, we quickly deliver Boomi integrations that solve business problems.

Why Dell-Boomi Atomsphere is the best platform for Agile Integration:

It is a hyper-productive development platform, so that you have project hours/budget to do more than basic mapping and transformations.
It is an agile product, where you can build the basic process, then revise and add BP support features
It is a hyper-productive product that frees up developers to be business process analysts
It is a full-featured product that supports all types of business process features, across the full life cycle.

Goals:
To understand the business context of the integration, and to build in features that minimize transaction friction and maximize velocity, transparency, and business value.

Deliverables:
-A high level flow chart that puts the integration in the context of the business process, and communicates the structure of the integration process.
-Notifications, alerts and logs that communicate status and errors.
-Boomi process features and additions that support the business process
-A migration plan that describes how the newly linked systems will get from the current state to the target steady state.
-An inventory of views and reports in the endpoints that will show transaction status, summary counts, and errors

Risks:
-Do you have access to knowledgeable business process owners that can explain the context?
-Are you knowledgeable enough about business processes to identify potential problems and solutions?
-Do you have a library of Boomi patterns from with to draw solutions from?
-Are there business process owners that can make decisions about statuses, notifications and recovery?
-Are parts of the overall business process missing from the discussion? These can include Returns, Reissues, special products or offers, additional status fields, out of stock, customer-sub-partner situations, etc. This risk here is that you find out about these very late in the project.

Estimation Guidelines:
-Number of Processes, number of endpoints, number of departments.
-Watch out for missing business processes, and ask questions like “Are these orders ever cancelled? How does that work?”

Transaction Quality:

In the fourth sprint, you work on quality and completeness checks for the transactions. Some integrations don't have these requirements – in Pub/Sub models, it is the target system's responsibility to do validation, for instance.

Here are some situations where validation in the integration is needed:

-**High value transactions** – B-to-B sales transactions, for instance, where many people care about the status of the transaction moving from Salesforce to Netsuite

-**When the target is the source of master data** – when "subsidiary" systems push data into a master system, you should validate the transaction against the target before attempting to create/Update

-**When logic is complex** – complex validation logic is best handled in the integration layer, insulated from changes on either end

-**When endpoints are out of your control** – either because of organizational issues, or they are part of another organization

-**When the data is time sensitive** – often transaction, HR or user data is time or date sensitive, and should be validated before it is passed on to other systems

-**When the target systems don't provide much/any validation upon input**. Some systems will happily accept an invalid transaction and not error, or will provide cryptic error messages. You can do better with validation in Boomi

If several of the above situations are present, consider building validations into your Boomi processes. In our experience, 95% of the errors in an integration are data errors – invalid, missing, old, incomplete, mal-formatted data. So validation should focus on data quality. Sometimes, the hardest part is to understand the common or uncommon errors in the data.

How to do Validation/quality checks with Boomi:

-Use a Try-Catch to handle an error in the Boomi Process. Try-catch is great, but should be reserved for un-anticipated problems, like connector or end-point failures. Better and more effective tests can be done with other techniques.

-Business Rules Shape: the best tool for applying complex tests or business logic. Not so easy to use – check out our 5 part video series about using Business Rules Here http://www.kitepipe.com/instructional-videos

-Cleanse shape: A quick way to apply required value logic across many fields. If you have tried this shape and are mystified, understand that it uses metadata in a profile to provide the rules.

-Returning connector errors: Set your outbound connector calls to return errors (instead of failing) and test for and process the error

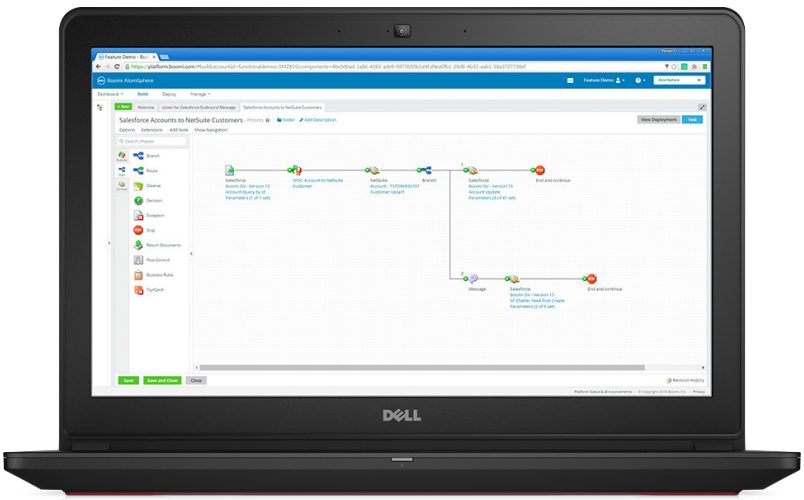
-Lookups against data sources (internal to the process, or external systems)

Goals:
To understand and detect transaction data quality problems, and to handle them in a way that speeds the resolution of the error.

Deliverables:
-A list of fields that require quality checks, and the requirements for those checks.
-The Maps, business rules, lookups and flows required to do the checks
-A set of test data sufficient to exercise all of the branches and conditions in the processes
-Some design around error handling – we often try to get all errors and messages into a single cache, and dump it at the end of the process to a notification scheme (email, post-back, etc)

Risks:
-Is there good documentation that covers required fields, logic, data transforms, lookups, etc?
-Can your team or others access the source system and build error data?
-Has anyone thought about who is in the best role to fix transaction data problems?
-Who should be notified for platform / endpoint problems, and how?
-Resistance to the concept of putting data quality logic in the integration layer. Pub/Sub and ESB practitioners typically trust the subscriber to do its own validation. So check the situations list above.

Estimation Guidelines:
-Number of Processes, number of Maps, number of mapped fields.
-Watch out for objects with very different validation requirements in different status states – these multiply the build and test effort



UAT and Go-Live:

Integration can be part of a larger project, and the system testing is often done in that larger functional context. This insures that the Business Process goals are met, not just the technical integration specs. In this context, the Integration Consultant supports UAT, User Acceptance Testing, and makes fixes and updates.

Kitepipe Deployment Documentation Components:

-Deployment Checklist: this is a set of tasks that are done by the consultant as part of the deployment, and include setting tracking IDs, reviewing Extension initial values, refreshing profiles, updating start shape queries, and several other pre-production tasks. These are dated and initialed to provide an audit trail.

-Extension Sheet: All extended values in the process are logged in a sheet, and the Dev, Test and Prod values are documented. Plus any persisted parameters are noted.

-Support Sheet: Here we note problems that may occur in Production, and where to go to diagnose and fix them. This is a "cheat sheet" for a developer who may come along later, and need to quickly repair a problem.

In some cases, right on the heels of UAT is Conversion, when a target system is loaded from the source system, often using the Integration Plumbing in Boomi. Where the endpoints are established systems, with an existing integration (which may be manual) conversion is not usually required. Where the target endpoint is a new setup, the integration processes can be used in conversion, to load the new system.

The Boomi consultant plays a key role here, as one-off fixes are often required to solve data conversion problems.

With or without conversion, the tested integration processes are prepared for deployment, and then deployed. This may involve the use of runtime atom production environments. It certainly includes the creation of tracking fields, and their assignment to document properties in each process. Plus, environment extensions that will control test vs. production behavior must be identified, assigned, and tested in each environment.

At Kitepipe, we use a set of deployment documentation that changes a bit for each project. The components are consistent, and include a set of sheets that document the steps and settings needed to successfully deploy onto production. The Checklist helps avoid omission of steps in the process, and the properties sheets document settings that a support developer will need to refer to.

This is a time to do performance tuning if needed, make messages and errors consistent, and to ensure that tracking fields and environment properties are consistent and appropriate. We typically don't design for performance – there are many quick config changes and process updates that can fix performance problems in a Boomi runtime.

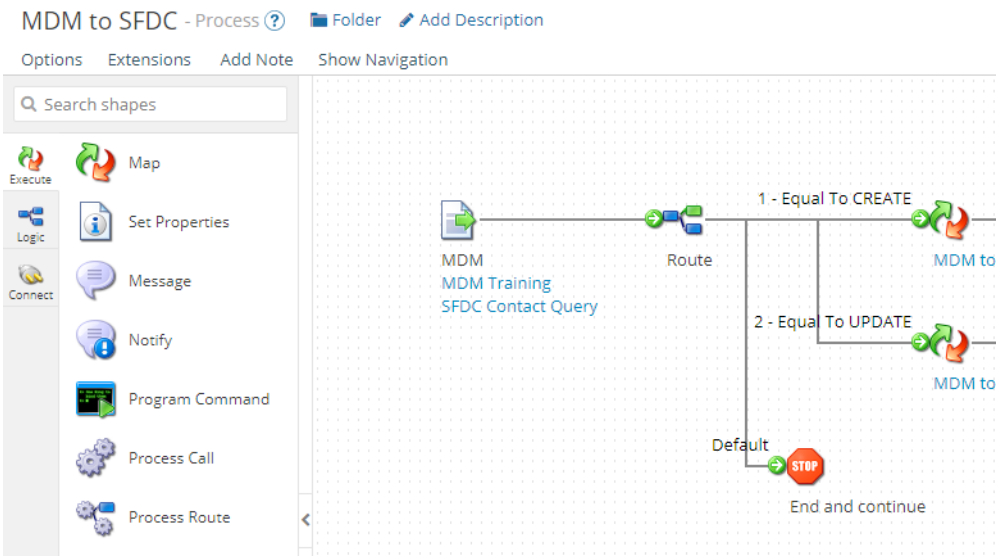
This work happens mostly in the deployed Test and Prod environments, and monitoring and debugging skills are key.

Goals:
To deliver a set of integration processes running live in production. This includes system testing and deployment, and may include conversion tasks if the integration processes are to be used for data conversion, as well.

Deliverables:
-Comprehensive system tests that are designed and executed from a business process point of view, not just from an integration technical point of view.
-Updates or fixes to the Processes driven by a UAT punchlist.
-Deployment documentation, as noted above
-A set of deployed processes, with appropriate tracking and environment extension properties, running in production.
-If in scope, data conversion processes and load cycles that load an empty target system from the source system as part of the project. This is often a separate deployment and UAT cycle.

Risks:
-Are there well-populated sandbox environments, or test accounts, that can support extensive system and UAT testing?
-Is there a business process-level system test plan which will exercise all aspects of the integration? Testing to interface specs can miss important questions like: Who will update this field, and how and when will they do it?
-Is the integration going to be used to do data conversion as well? That is, accomplish the initial load of the target system? This is a separate project all in itself, even starting with a tested interface, and can add significantly to the scope of the overall project.
-Are the runtime UAT and Prod atom environments in place?

Estimation Guidelines:
-Number of Processes, number of connectors in the processes, number of Extensions.
-Scope of system test plan
-Requirement to do data conversion with the newly developed integration processes – volume and variation of that data.



Clean-up and Tech Transfer:

Once the dust settles and the integration is live, there is an opportunity to clean up the Build environment, document the Boomi integration processes, and train the maintenance resources.

In the heat of battle when bringing a big project live, you can end up with multiple versions of Profiles, logic, and processes. Boomi has great versioning and where-used tools, but it is very easy to end up with three identical record profiles in use in five different processes. You want to delete and consolidate the "dead objects" and not leave it for the next guy to try to figure out what is the difference, if any, between these two five hundred element XML profiles.

We use specialist tech writers to capture mapping and logic detail from the Boomi environment.

Although Boomi is somewhat "Self Documenting" – you need some experience to go into the Build environment and answer detailed questions about how this field is getting into that one, or why that record was excluded.

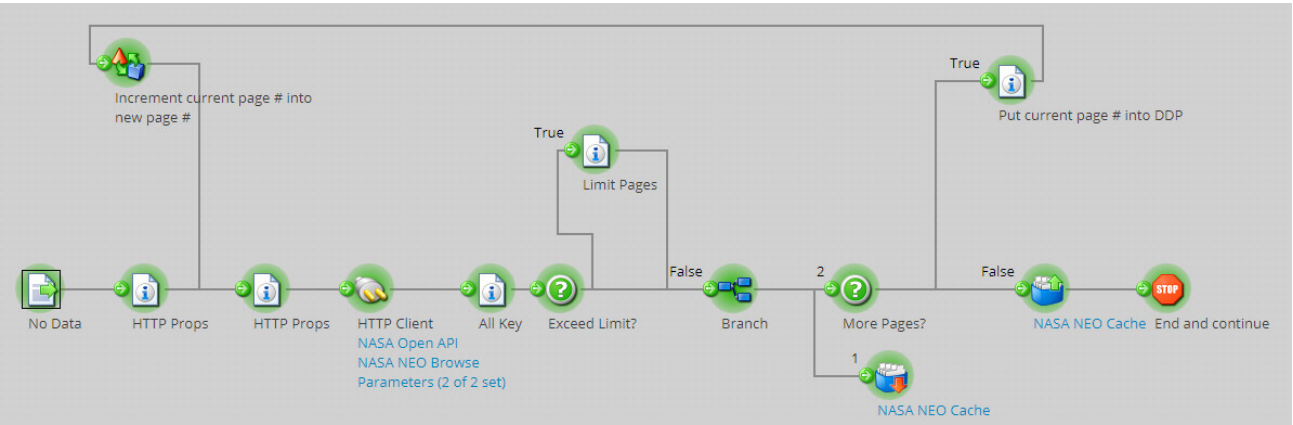
Plus, you will inevitably be doing some fine tuning of the Interface as un-planned operational and data situations occur.

Goals:
To make the integration processes “Maintenance-Ready”, and hand off the processes to the maintenance team.

Deliverables:
-A set of “cleaned-up” boomi processes, where redundant components have been removed, and naming and process structure is consistent from Process to Process
-Documentation that describes the overall context, and specifics about what is different and unique about each process. Documentation need not describe each process in detail, but should identify process and document properties, environment extensions, any extensive scripting, and error handling, at a minimum
-Exported main maps, in excel form, identifying the version number.
-Go live documentation, as described above.
-Handoff to the maintenance team – often done as a walk-through of the processes in the build environment, and review of recent Process Reporting logs

Risks:
-Have multiple developers worked on the project without clear guidelines and standards? If so, there may be a good bit of cleanup to do
-Are there trained Analysts to take over the maintenance? You don’t need to be at a developer level to keep an eye on a set of Boomi processes in production, but it helps to have a playbook containing common error situations, and what to do.
-Are the environments stable – source, target, and atom environments?

Estimation Guidelines:
-Number of Processes, number of connectors in the processes, overall hours estimate for the project.



Integration Analysis – the "Zero Sprint":

An Integration process in production can be thought of as a technical means for spreading a single business process over multiple technical environments.

It often appears that through the integration, one process is handing off transactions to another, or sharing data with another that is unrelated. It is common to think of the two ends of the integration in isolation, and just to focus on the data or transactions that are passing between them.

You can miss important aspects of the project when you treat the endpoints as independent. A classic situation is order fulfillment, where order information is passed from one system to another, but product master file information that is needed in fulfillment (like addition of new products) is not passed.

A way to get a better view of the dependencies that might otherwise be missed in the integration is to do and Integration Analysis up front, and ask some key questions:

-Is there a higher level business process that includes both ends of the integration environment? For instance, you might identify Product Management and Customer Service as higher level business processes that need to be supported by the integration.

-What are the needs of these higher-level processes? Product Management may want to insure consistent product master information. Customer Service may want to have order fulfillment status and history information available.

-What business events impact both sides of the process? These are often the gotchas that only become clear once things are in production. For instance, cancelled orders must often be rolled back on both sides.

-What exception conditions impact both sides of the Integration?

Techniques to use in the Analysis Phase:

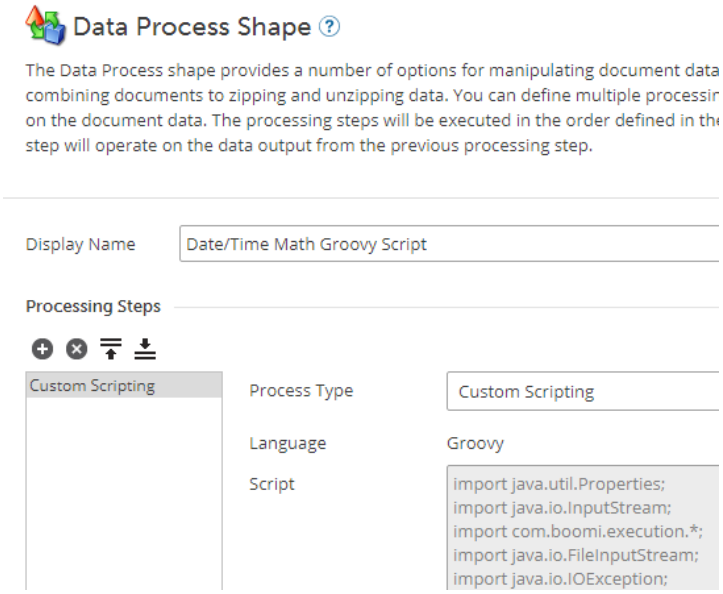
-White-board sessions with supervisor and manager level users to identify business process exceptions

-Use-case workups of high-level business processes that capture normal and exceptional situations

-Change analysis of the Use Cases, where you ask "is there a situation where the xxx master-file changes?" The goal here is to challenge assumptions about data availability and stability, and identify exception situations

This is a good opportunity to use Use Case analysis to describe and walk through the high level processes, and especially to identify exception conditions. As developers, we all handle the "happy path" – it is the exception situations that bite us. Exception situations in handling a single field are part of the development process. But, exception conditions at the business process level can have a significant impact on the project scope and schedule.

It is the integration processes that you don't know about that have the greatest impact on the project. This sprint encourages a thoughtful search for situations and exceptions that impact scope.



In Conclusion:

It is my hope that this White Paper has given you a framework with which to think about, structure, scope, and estimate integration projects. I've distilled my many learnings over an extensive IT services career in hopes that you might benefit, and execute integration projects more successfully.

And thank you in advance for comments, suggestions, anecdotes, or improvements that can help the Cloud Integration Community be more successful.

Larry Cone

About Larry Cone

Larry Cone, Founder and Chief Solutions Architect at KitePipe, has been active in the IT Services sector for most of his career, first at Accenture, where he learned the basics of professional IT services. Later as founder of Cone Software, he built a custom software development practice focused on field and shop-floor computing in the Pharma industry. Along the way he led the software development team for the Emmy and Academy-award-winning Skycam 3D volumetric camera motion system.

Currently Leader at Kitepipe, he is applying business analysis, cloud architecture, and quality principles to deliver outstanding results in Cloud Integration Projects.

Larry manages the Boomi Services Team at Kitepipe, and is involved in all aspects of Boomi integration delivery. He consults with Dell-Boomi on Architect-Level training materials. He writes a popular cloud applications blog at ittoolbox.com/coneblog. He spoke on Boomi integration architecture at Gartner Data and Analytics 2017 and Boomi World 2017."

Phone: 844.232.2227
E-mail: boomihelp@kitepipe.com
Web: Kitepipe.com