GETTING TO THE HANDSHAKE: A ROAD MAP FOR I.T. / BUSINESS COLLABORATION

INRULE

INRULE

Contents

Getting to the Handshake

This paper is a high-level walkthrough of the process we have been successfully using with clients to get them to what we at InRule call “the handshake.” The handshake has both literal and symbolic meaning. Literally, it is an actual handshake that signifies agreement to the structure of how decision logic will be communicated between subject matter experts and developers. Symbolically, however, it represents much more than that.

The handshake represents the acceptance of an accountability model which makes “what the decision logic is” a business focus and “how that logic is executed” a technology focus. Throughout this process we bridge the gaps that exist between IT coding decision logic and users driving dynamic applications that assist with or automate their decision making.

Example Business Problem

For purposes of this paper, we’ll use an example business problem to highlight some of the aspects of the process. Our example is a mortgage company that is losing money because they are approving too many risky loans. They have realized that the problem stems from the fact that the software can’t be changed as fast as market conditions are changing around them and loans are being incorrectly approved in that gap. The inability to roall out decision logic changes in a timely manner is causing the business to lose money and they would like to address the problem using a decision platform.

The business requirements for this example are twofold:

Requirement Number 1 - The right people (those who understand mortgage eligibility) should be responsible for changing the rules behind mortgage decisions.

Requirement Number 2 - Changes need to be rolled out quickly. It is not acceptable to wait out a typical software development and testing cycle to make a simple change to a mortgage eligibility rule.

A Separation of Concerns

As contrary as it might seem, bridging the gap between IT and the business is based on a separation of concerns. Separating concerns is about taking the existing process of how software is changed and designing a communication and accountability model to streamline that process. This process takes participants from business intent to the technology implementation, and does it with strict and mutually understood boundaries around what each stakeholder has to worry about.

First, it is important to have clear ownership boundaries so that everyone involved is clear about their own responsibilities. Typically, subject matter experts are responsible for defining the decision logic, but after that, the accountability and responsibility lines become blurred.

The developers hold the subject matter experts accountable for the quality and consistency of the logic’s definition and the subject matter experts hold the developers accountable for the accuracy in translating from the description to the program code that executes it.

Second, we need to reduce the number of translations associated with traditional development. Most development starts with requirements documentation by subject matter experts, followed by developers interpreting the requirements and coding the logic so that it will perform as the subject matter experts intended. The elapsed time for each cycle time is very lengthy and along the way someone is bound to be overly presumptuous in their interpretation or ambiguous in their communication. Either of these scenarios causes another lap around the track. Although these errors from translation are not intentional, they still lead to finger pointing and animosity across the gap. We want to eliminate those meetings where the sole goal is figuring out who

to blame for why something does not work the way it should.

In addition, we want to capture the intent directly from the person who understands it best - the subject matter expert. In this case, our subject matter expert is the person who knows whether or not a mortgage should be approved. The subject matter expert, or SME for short, needs a tool to articulate their intent in a way that they connect with. Additionally, SMEs also need to ensure that what they have created is correct. This is referred to as verification and it is absolutely essential to separating concerns. Without a way to replay and test their decision logic, they will not accept accountability. We want the SME to be confident that the rules and calculations they have authored are correct and that the expected results will occur.

This separation of concerns produces a sense of ownership and accountability. People become invested because the logic is something they created and it gives them control over the rules and calculations at the heart of the decisions they are on the hook for every day. In our example, the decision to either approve or deny mortgage will be exclusively in the hands of the business instead of a shared ownership with IT.

Starting the Journey

To get to the handshake we want to **discover, value, document, design**, and **implement** the decision or business logic in a model of separated concerns. For this journey, for some fun, let's take a Google Maps approach.

Our journey starts at typical corporate software development, where some are the common sights are costly maintenance, slow change turnaround, and lots of finger-pointing.

Where we want to get to is **thinking in rules**, where subject matter experts maintain decision logic, turnaround for changes is quick and accurate, communication is efficient, and teamwork transcends the technology divide.

Get Directions from A to B

From: A ***Corporate Software***
Most Companies, World

Sights to see:
- Change turn-around is slow
- Maintenance costs too much
- Business logic is frequently wrong
- Lots of finger-pointing between business analysts, and development staff

To: B ***Thinking in Rules***
Best Companies, The Future

Attractions you can't miss:
- Business domain experts maintains business logic
- Turn-around is quick
- Accuracy is amazing
- Maintenance costs dropped by 90%
- The team is truly working as a team

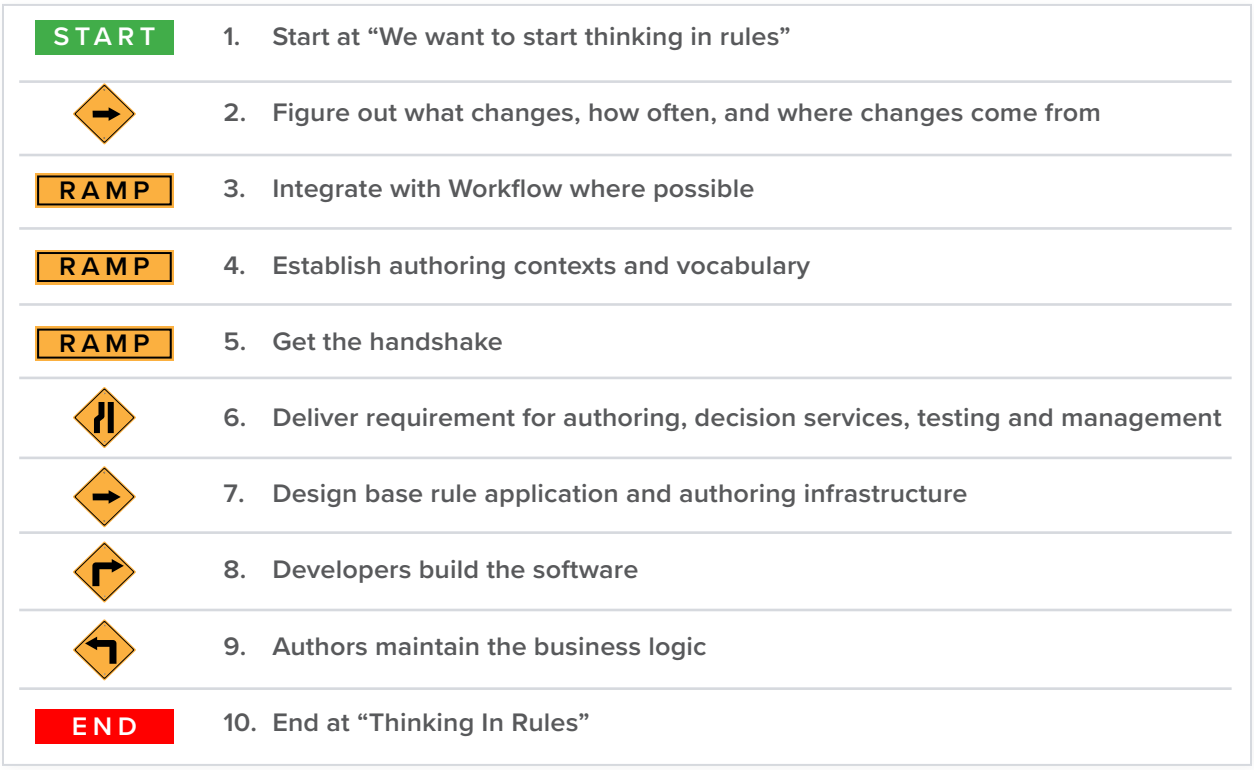
Routing Options

- [x] Optimize for long term
- [ ] Use Embedded Authoring
- [ ] Integrate with Workflow where possible

Get Directions

Overview of the Process

The figure below show an overview of the process that will be explained in this paper.



Early in the process, all team members work together to identify the things that change over time and make decisions about how to author them, how to describe them and how to consume them.

Late in the process, developers and subject matter experts will be working in parallel on different layers of the system, they both will then arrive together at the destination … **thinking in rules**.

Figure Out What Changes, How Often, Where Changes Come From: Variability Analysis

The first thing to evaluate is what changes. This is referred to as variability. In the current example, the mortgage eligibility rules are the thing that changes, and as such they are referred to as a variability point.

During variability analysis we collect the variability points and drive out what has changed historically, what needs to change now, and what we might expect for changes going forward.

In our example, the variability analysis might discern that mortgage eligibility rule variation occurs approximately once a week and that changes come from risk analysts.

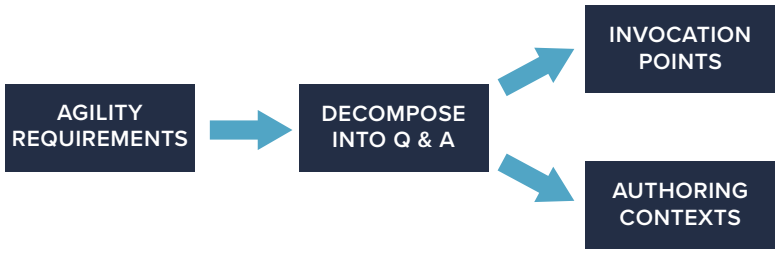
The output of the variability analysis is the **agility requirements document**. This document helps everyone involved understand what changes, who changes it, how frequently it changes and the historical costs associated with these changes.

At this point we’re not centering in on the technology solution, we are capturing the agility requirements and associated cost / benefit at the business level. This document is intended to span the gap and present the value proposition of agility to the business side while illustrating the relevance of agility to the technology side.

Separating Logic Definition and Logic Consumption: Question and Answer Modeling

Apps and Power Automate. Within Power Apps, InRule can be leveraged across Model-Driven Apps, Canvas After analyzing what changes, the next step is the question and answer modeling process. Q&A modeling is a break from the old method in which a business person writes requirements that the developer may or may not find useful or in some cases even make sense of. Instead, Q&A modeling results in a structure for communication between the business and IT.

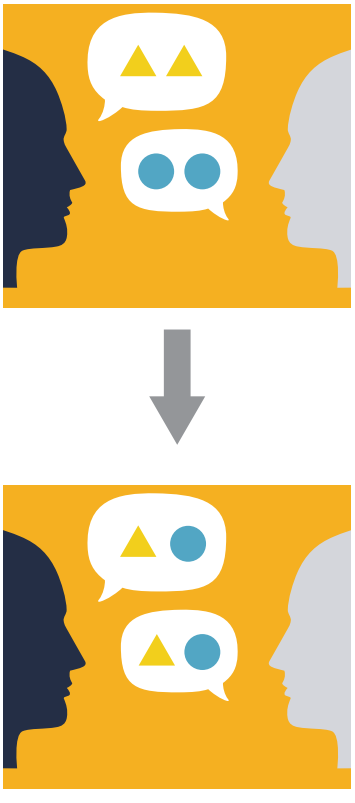
With question and answer modeling, each variability point is captured as if it were a real-time dialog between the subject matter expert and the developer. In essence, the developer asks "What should I do now" and the subject matter expert replies "Do this." This helps the developer get in the mindset of asking the subject experts to apply their expertise. At the same time it helps the subject matter experts get in the mindset of responding to a request from the developer, rather than articulating their logic and rules in a vacuum.



In the case of our business example, the question and answer model would result in something like this:

Q: The developer asks "Is this mortgage eligible?"

A: The subject matter expert answers with "yes it is" or "no it is not."



The key design discovery from this process is what the questions are and how the questions are posed. Each question and answer pair will ultimately become a delivered logic service. The question becomes the invocation of the logic by the developer, and the answer is defined by the subject matter expert in the authoring tool.

Q&A modeling provides structure and helps reduce the complexity of finding a common language. Developers will ultimately be concerned with asking questions and subject matter experts will ultimately be concerned with answering them. Q&A modeling also drives out gaps or one-sided scenarios where we have, for example, an answer the subject matter wants to give, but no question being asked that supports it. Jointly working through these issues brings a shared awareness of what each side of the dialog is actually doing.

The result of this modeling is a set of invocation points and authoring contexts. Authoring contexts are the subject matter expert's view of the logic that is needed, for example ***"Edit mortgage eligibility rules."*** The invocation points are the developer's point of view of how the logic will be called in code, for example:

MortgageDecisionService.IsEligibleForMortage(application).

Q&A modeling enables IT/business collaboration by aligning how people communicate, what the logic actually does, and how the underlying technology will be implemented. This helps drive out the design and builds a bond between IT and the business. It is building a way for them to communicate to each other in a vocabulary that resonates with both of them.

Establishing Authoring Contexts and Vocabulary

Information and actions analysis is a process that establishes the vocabulary that an author understands and uses to express logic. Each question and answer pair has an associated authoring context to express the answer. An authoring context has two basic ingredients:

» *What is known*

» *What actions can be taken*

It works something like this: The subject matter expert and the developer pretend that they are on opposite sides of a wall and can only pass paper notes underneath the door to each other. The developer is going to pass a question along with some associated information and the subject matter expert is going to pass the answer in return. This process highlights two things: the vocabulary for the information the author receives, and the vocabulary used by the author to express a response.

In the example, the first step is to ask the subject matter expert what information is needed to answer the question ***"Is this mortgage eligible?"*** The SME tells us he needs things like the age of the house, the credit score of the applicant, and so on. The items are captured on a white board and these define the author's vocabulary. The response vocabulary is composed of the possible actions, for example, ***"the mortgage is not eligible."***

Now comes the fun part, the vocabulary negotiation. Here is a really simple example: the subject matter expert says:

"I need the birth date and age."

The developer says:

"If I give you the birth date, you can figure out the age."

Through this negotiation they ultimately come to final agreement about a finite set of authoring vocabulary that will meet the needs of the subject matter expert.

Get the Handshake

The handshake comes at the end of the vocabulary negotiation. The developer and analyst stand up and the developer asks, "Do you subject matter expert agree that if you are provided with the price of the house, the age of the house, and so on, that you can determine eligibility for a mortgage?" The subject matter expert says yes, and asks, "Do you the developer agree that you can provide these pieces of information and carry out the directives of eligible or not eligible?" and the developer says yes.

Then the developer and subject matter expert literally shake hands.

This is more than a symbolic moment. It fosters a shift in the way IT and the business communicate. From that moment forward, their communication about new requirements is not a business person saying "I need you to change the logic in this application." Instead the new communication is something like "Can you put this new piece of information in the request?" The developer doesn't even need to know why that information is needed, because their concerns are now separated from the actual logic.

Much like question and answer modeling, this analysis serves to strengthen team communication at the same time that it drives out design details.

As shown in the figure above, two design documents – Authoring Requirements and Verification Requirements – are created as a result of the information and actions analysis and vocabulary negation. Authoring requirements define what the authoring experience is composed of. Verification requirements define what tools the subject matter experts will use to verify that the rules they have authored are in fact what they intended.

Deliver Design Documents for Authoring, Decision Services, Testing and Management

Once requirements are captured we must balance three things to start finalizing our design:

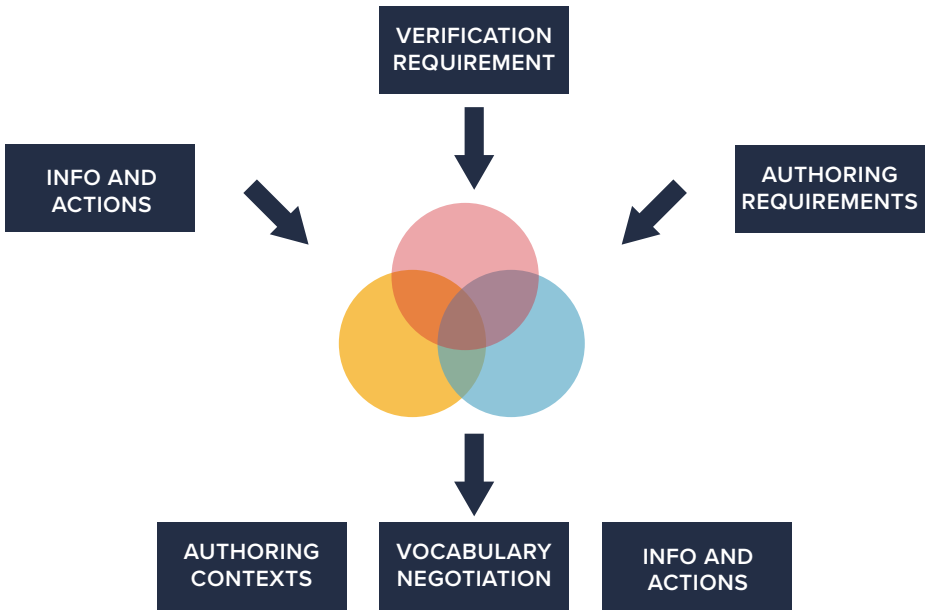
- *The agility needs of the business*
- *The needs of the author*
- *The needs of the developer*

The intersection of these defines where the team will start putting stakes in the ground for project design.

During the design process we want to capture:

- *The authoring design, which covers the editing and verification of the logic*
- *The management design, which covers the storage, versioning and distribution of the logic*
- *The logic service design, which covers how the logic is called and how the response should be interpreted*

By the end of the design phase, the team has the designs for the authoring, rule management, and logic services. All three of these can then be built and tested in parallel.



Design Base Rule Application and Authoring Infrastructure

Now that we understand our authoring contexts and required vocabulary, we need to code the schema for each authoring context and customize the vocabulary as required. This is done by creating a new rule application and respective entities for each authoring context.

Developers Build the Software / Authors Maintain the Decision Logic

At this point in the process we have established: what changes, how it is described, how to author it, and how it will be consumed.

We also have helpful design and requirements documents used by different people involved in the project.

Developers and subject matter experts can now work in parallel. The developers will build the software and subject matter experts will maintain the decision logic.

Developers Build the Software / Authors Maintain the Decision Logic

At this point in the process we have established: what changes, how it is described, how to author it, and how it will be consumed.

We also have helpful design and requirements documents used by different people involved in the project.

Developers and subject matter experts can now work in parallel. The developers will build the software and subject matter experts will maintain the decision logic.

Using InRule® with the Process

Developers use irAuthor® to set up a schema that reflects the vocabulary agreed upon during the vocabulary negotiations. The rule authors, typically the SMEs, then use irAuthor and irVerify® (part of irAuthor) to create and test rules.

Using irCatalog®, developers set up permissions and a rule promotion strategy.

At runtime, irSDK® integrates rules with the software application by taking data from the calling program, getting the rules from irCatalog, sending the combination of rules and data to the irServer® rule engine for execution, and getting the results – the intent of the author – as a response.

Summary

Let's take a look at a change request using our original mortgage business example. There is urgent need for a rule change to prevent mortgages where the building is in a flood risk area and the foundation is cinder block.

- ***The subject matter expert writes up the change and explains it to the developer***
- ***The developer interprets it, codes it and tests it***
- ***The developer goes back to the subject matter expert for approval and (in the best case scenario) the change is approved. If not, we go back to the previous or even first step.***

With the Thinking In Rules model, the process looks more like this:

- ***The subject matter expert asks the developer to provide the building foundation type***
- ***The subject matter expert modifies associated rules and verifies that results are correct.***

This process helps make decision logic a business focus, and logic execution a technology focus. That's what we mean by Thinking In Rules.

About InRule

InRule Technology is an intelligence automation company providing integrated decisioning, machine learning and process automation software to the enterprise. By enabling IT and business leaders to make better decisions faster, operationalize machine learning and improve complex processes, the InRule Intelligence Automation Platform increases productivity, drives revenue, and provides exceptional business outcomes. More than 500 organizations worldwide rely on InRule for mission critical applications. InRule Technology has been delivering measurable business and IT results since 2002. Learn how to make automation accessible at www.inrule.com.

Inrule Technology

651 W Washington Blvd #500
Chicago, IL 60661
www.inrule.com