Are Financial Markets facing a Requirements Crisis? Does the answer lie in Behaviour-Driven Development

In this article, ***Mike O'Hara****, publisher of The Trading Mesh – is in conversation with* ***Konstantin Kudryashov, Liz Keogh, David Evans, Alan Parkinson, Chris Matts*** *and* ***Dan North****.*

HINDSIGHT INVIQA TheTradingMesh
Unravelling the complex, interconnected systems of global financial trading

Introduction

The Banking and Financial Markets industry is a dynamic and ever evolving landscape, which in recent years has become increasingly reliant upon technology. And as more and more processes within banking and finance become automated, that dependency is becoming ever greater.

This is particularly the case in trading and exchange operations where - in order to keep up with changing market conditions, a rapidly evolving regulatory landscape and increased competition - practitioner firms are constantly required to either deploy new systems or enhance their existing ones, all at substantial expense.

Given this rapid pace of change, the task of identifying project requirements and translating those requirements into clear system specifications poses a significant challenge to all concerned, leading to what some in the industry are now calling a "requirements crisis".

Project failures

It is an unfortunate fact of life that many IT projects end up costing a great deal more money and delivering much less value than planned. In fact, in 2013, only 39% of IT projects were delivered on time, on budget, and with the required features and functions[1].

So it is clear that many projects fail, but why?

> *" Many software project failures, regardless of the industry, are caused by people focusing too much on scope and functionality rather than the added value the project needs to deliver."*
>
> Konstantin Kudryashov, BDD Practice Manager at Inviqa

According to Konstantin Kudryashov, BDD Practice Manager at leading open source development firm Inviqa, one of the reasons is that people tend to focus on the wrong things when putting requirements together.

"Many software project failures, regardless of the industry, are caused by people focusing too much on scope and functionality rather than the added value the project needs to deliver".

He refers to a recent survey on business technology by McKinsey[2], which reports a growing dissatisfaction with IT's ability to facilitate and meet business objectives. "IT has become less effective at enabling business goals because IT people spend time talking about features and functionality, which in some cases don't deliver any value at all".

"In any business, the first thing an investor looks at is how the business will provide a return on investment. But we're in this weird state in IT where people spend all their time talking about features and functionality, which in some cases don't deliver any value at all".

[1] *http://www.projectsmart.co.uk/docs/chaos-report.pdf*
[2] *http://www.mckinsey.com/insights/business_technology/it_under_pressure_mckinsey_global_survey_results*

Another common reason for failure is miscommunication, particularly in more complex projects where requirements are often communicated between business people and developers through a series of 'Chinese Whispers', resulting in much of the context being lost in translation. Research shows that 54% of defects in delivered software are due to incorrect, misunderstood or badly specified requirements as opposed to implementation errors and program bugs[3].

So what can be done to address these challenges?

Behaviour-Driven Development

An increasing number of firms in the finance sector are now turning to Behaviour-Driven Development (BDD) as a way to ensure that not only are the correct requirements identified, specified and communicated, but that projects do indeed deliver real value to sponsors through a truly collaborative process.

BDD, a concept created by Dan North in 2003, is a software development process that grew out of TDD (Test-Driven Development) as a way of enabling business analysts and software developers to better communicate and collaborate via shared tools and shared processes.

Like many of the best ideas, BDD is very simple in concept but very powerful in terms of what it can achieve. At its core is the 'Given-When-Then' format, which allows a requirement to be specified based upon examples of its desired business behaviour. This simple approach enables business analysts and those with domain knowledge to specify, in a framework immediately understood by developers and testers, what should happen when a particular event occurs under a given set of circumstances.

"BDD really improves communication", says Kudryashov.

"It helps every single person in the delivery team to understand why they are doing what they are doing and why they are delivering a specific piece of functionality", he adds.

Clarifying uncertainty

Independent consultant Liz Keogh, a BDD expert who has been working with the framework since 2004 and who regularly blogs on the topic, explains how BDD can help remove uncertainty from a project.

"BDD uses examples in conversation to illustrate behaviour and is therefore very useful in helping clarify requirements", she says.

"And BDD is also really useful in helping people appreciate where uncertainty is and in finding a different way of addressing that uncertainty", she adds.

"The teams that are most successful at this are the ones who get feedback on the most uncertain aspects of their work before they automate the examples".

" BDD uses examples in conversation to illustrate behaviour and is therefore very useful in helping clarify requirements." Liz Keogh, BDD expert & blogger

How might this work in practice?

"Say a new regulatory requirement comes along and people start discussing what that requirement means and what the relevant applications should do. They'll frequently end up getting into arguments around what the outcome of a particular example ought to be. You can tell from the uncertainty in that argument that this is something new that doesn't lend itself well to analysis. So the right approach in this sort of situation is to try something out, to experiment and to get feedback. Talking through examples can help you to spot that", says Keogh.

"If there is a high level of domain expertise, if there is a person who is very familiar with the regulation you're implementing, then the BDD approach is great for drawing out that expertise and learning from that person. And conversely if it is something very new and there is no real expertise and people are just guessing, BDD can be used to identify and clarify that uncertainty".

[3] *http://www.itu.dk/~slauesen/Papers/PrevDefectsProceedings.pdf*

Continuous delivery

Delivering value in small iterative steps and being able to prove things work before implementing them at full scale is a key aspect of BDD, according to Kudryashov.

"In six months the business environment could be drastically different from when the project was first started", he says.

"Taking this into account, you need to treat the specific software as constantly evolving, because the features that deliver value today might not deliver value in the future. So you need to constantly deliver small, incremental improvements and measure the impact on the business. That's what the essence of BDD is. Basically it drives your company towards your business goals in small steps, constantly measuring and making sure that you're going in the right direction". Keogh agrees that the ongoing, continuous nature of BDD is an important factor when it comes to delivering value.

"If your requirement is to ensure that you have some limit on the amount of risk associated with a particular counterparty for instance, knowing that that's what you're trying to do with the underlying functionality and having that vision very clear in everyone's heads helps you, after you've made all of the functionality work, actually check on an ongoing basis that you're really meeting that vision and those capabilities" she says.

" The first thing BDD gives a team is a clear mandate to use examples as first-class objects in the software development process." David Evans, Independent consultant

The power of examples

The key to the success of BDD lies in the power of using concrete examples when discussing requirements, according to David Evans, an independent consultant specialising in agile quality.

"The first thing BDD gives a team is a clear mandate to use examples as first-class objects in the software development process" says Evans.

"Our brains naturally latch onto concrete examples rather than generalisations - we prefer to discuss specifics when getting to grips with new or complex ideas. Yet traditional analysis approaches tend to dismiss examples as little more than a means to an end. Unfortunately, that means potentially useful examples tend to be discarded and reinvented multiple times in the analysis, implementation and testing of a feature. BDD can short-circuit that by allowing discussions to centre on a small set of key examples that are meaningful to all."

Evans agrees with Liz Keogh that discussing the examples is the most important part of BDD, whereas many teams focus only on the automation of those examples as test scenarios.

"BDD is first and foremost about improving communication between those responsible for what a system must do and those responsible for how it will do it," he says. "Most of your quality assurance comes from preventing misunderstandings, hidden assumptions and undiscovered exceptions leaking into the code. The examples have immediate value as discussion points. Later they will have additional value as automated acceptance tests. Ultimately they will also have ongoing long-term value as documentation of that feature's behaviour."

" If teams successfully focus on the requirements and communications aspects of BDD first, the tools add more value." Alan Parkinson, CEO of Hindsight Software

More than Test Automation Toolkit

Alan Parkinson, CEO of the specialist BDD firm Hindsight Software, is keen to point out that BDD is much more than a set of tools.

"Many people only equate BDD with test automation and immediately think of tools like Cucumber, Behat, SpecFlow etc. So they adopt BDD for test automation and don't discover the requirements benefits until much later, if at all", he says.

According to Parkinson, the consequence of this can be profound.

"Because the business Stakeholders haven't been involved in writing the valuable Given-When-Then statements that express the business requirements, the statements end up looking like test scripts. As a result, they don't add any value. If teams successfully focus on the requirements and communications aspects of BDD first, the tools add more value", he says.

Parkinson points out that when legacy systems have to be maintained across organisations, due to mergers and acquisitions within the financial sector, it often results in 'duct taped' integration between systems.

"One problem that firms face, when dealing with legacy systems or integrating systems due to an acquisition, is the lack of documentation", he says.

"This poses a significant problem when the system needs updating, say for a new regulatory requirement". An example of this is the recent introduction of the seven-day bank account switching obligations, which has been problematic for a number of banks.

"BDD is really useful in documenting the current understanding of an existing system. And then automation tools like Cucumber and SpecFlow can be used to check if this understanding is correct. This creates living documentation for the existing systems, where you only have to run the automated checks with the BDD scenarios to discover if the requirements are being met. As and when you need to modify or replace the legacy system, you can use that living documentation to confirm that no regressions have taken place and that new functionality has been successfully implemented", he says.

" Business people can very easily understand a structure like Given-When-Then, because it's very straightforward, semantically clear English."

Chris Matts, original BDD core team member

Specifications in plain English

One of the members of the original BDD core team and co-creator of the 'Given-When-Then' format is Chris Matts, also known for his innovative work on Feature Injection and Real Options.

"Business people can very easily understand a structure like Given-When-Then, because it's very straightforward, semantically clear English", says Matts.

"So rather than having a business analyst or subject matter expert come up with an example that's abstracted into something that only they understand, the BDD approach forces them to express things in such a way that the developer doesn't have to do any translation," he says.

According to Matts, BDD gives business people the ability to frame their knowledge and domain expertise into a series of examples that developers and testers can understand.

"Traditional business analysis is all about extracting and creating abstract models that are passed to the developers for implementation. With BDD, any businessperson who understands a requirement is able to describe and explain the context in the Given-When-Then format. This makes it very easy for the business users to express requirements, because they're talking about examples rather than abstract models. And it becomes additive as well, meaning the examples can be expanded and built upon all the time, which is particularly useful in finance because there are so many intricacies", says Matts.

Feature injection

Working with examples is all very well, but how can you ensure you are working with the right examples? The answer lies in an evolution of BDD created by Matts called Feature Injection, which consists of three steps, as he explains.

"The first step is identifying where is the value. The value in any IT system is generally in the output, but that's not normally the way the value is expressed. So with regulatory requirements, you would first identify what output you want. What are the reports that you're going to give to the regulators and what sort of data will be on those reports?

"Based on that, you work out what data you need in the system and what calculations you need to do on that data. Also what needs to be pushed upstream into trading and risk management systems for example. That's the second step, identifying what inputs you need and what processes need to be run.

"The third step is actually identifying the examples. That's where you need the business domain knowledge, and this is where BDD really fits nicely because it helps you specify all the different concrete examples. When you do that, you can go to the whole organisation and ask if anyone can think of an example that is going to break the set of rules you've defined, which means you can now start working on exceptions, whereas previously you would try to work on the general case and maybe ignore exceptions".

According to Matts, this can really speed up the development cycle, from initial requirements through to the delivery of some kind of minimum viable product or the first iteration of a working system, because it makes it very easy for developers to code up examples that have been clearly defined using the Given-When-Then template.

BDD on the trading desk?

Dan North, the original inventor of BDD, has followed an interesting path since originally coming up with the concept when working at ThoughtWorks in 2003. Since then, he has spent a couple of years working on ultra-low latency trading systems at a proprietary high frequency trading firm and is currently working with a large US investment bank.

How do the BDD principles translate to a high frequency trading desk?

"At ThoughtWorks, I had a reasonable sized team of business analysts, programmers, testers, project managers, team leaders, etc. There would be an array of various stakeholders and we'd be delivering in that context where there were typically a lot of corporate constraints", explains North.

"Whereas at the HFT firm, I was working with very small, tightly integrated teams where the developers are literally on the trading desk with the traders. The speed at which you can iterate on software in that kind of environment is remarkable. I was used to operating on a timescale of a few weeks or more for a release, but now I was delivering significant amounts of functionality in days or less.

"Part of this was because the team members were very experienced but part of it was because of the immediacy of valuable feedback from the traders. I'd be sitting next to traders who would explain what they were doing and why, and more importantly they could explain the rules by which they operated. We would go off and build that, come back, they would test the results against the rules and that would be the entire development cycle, of course within the constraints of the usual rigorous risk checks. We dispensed with anything we deemed unnecessary in terms of writing down scenarios and other typical 'agile' artefacts. And there was no ambiguity because we had the trader as our feedback mechanism", says North.

Complexity and scale

Is there a level of complexity at which BDD breaks down?

"The level of abstraction changes as the system becomes more complex, but it's fractal", says North.

"You can describe at a detailed level within code what the code should do, you can describe at an application level what this particular system should do and you can describe at a portfolio level what a suite of applications should do. For each of those things you're likely to use slightly different vocabulary because you're in different domains. One domain might be risk management, one level below might be netting and settlement, the level below that might be getting data in and out of databases.

"The mechanism doesn't change. It's description by example, that's the core part. At the lowest level you're coding by example, at the middle level you're doing analysis by example, and at the top level you're almost doing governance by example. But it's always doing things by describing observable behaviour. What changes is the tooling at each level. For example, Cucumber[4] is a really good tool for the middle level, but at the level below that you'd probably just use code to describe other code. You can still write Given-Then-When structured code using any open-sourced testing framework, you don't need a dedicated tool for that.

[4] A tool for running automated acceptance tests written in a BDD style

Likewise at the higher level, those guys aren't interested in something as fiddly as Cucumber. But you still want them to describe, by example, what they want across their portfolio" explains North.

BDD beyond software

Although BDD was developed as an approach for software development, the concept is bringing value to other areas, according to Liz Keogh.

"I'm increasingly finding that people are starting to apply the philosophy of BDD to areas outside of software" she says.

" The BDD approach to analysis and identifying a process in terms of its observable behaviour is something that can be used outside of software and could have enormous impact on value stream mapping and that kind of thing." Dan North, Original inventor of BDD

"Particularly where people are using examples to clarify transformation requirements. This might be when a firm is adopting Agile for the first time, actually getting examples from people and getting ideas of what that means to them and what might happen in their environment and their context".

North agrees with this.

"The BDD approach to analysis and identifying a process in terms of its observable behaviour is something that can be used outside of software and could have enormous impact on value stream mapping and that kind of thing", he says.

"In trading, from pre-trade through to post-trade, settlement, analysis and so on, describing all of that in terms of scenarios and examples is an enormous clarifying exercise and could help in a number of areas, like regulatory & compliance, market surveillance, or even bringing new traders up to speed".

It is clear that BDD is an extremely powerful mechanism and is likely to become more and more widely adopted, not just within the financial markets but also across many other industries.

"BDD changes the way that everyone looks at software", says Konstantin Kudryashov.

"At Inviqa, when we've introduced BDD to organisations, from start-ups to FTSE 250 firms, the overwhelming response from everyone who gets involved, both technical and non technical, is the same - that it's so logical, and so obvious. Developers who get into BDD generally say that they wouldn't be able to go back and look at software engineering in any other way. It changes their perception".

Chris Matts agrees.

"Anyone who wants to develop software properly will eventually be using BDD because it's such an obvious way to do things. Instead of creating a model, you create examples and you drive your development from those examples. Sadly, I don't foresee a day where everyone does things this way because people are entrenched in the old ways. Whenever you've got a legacy system, people always use it as an excuse not to do this kind of thing. And getting software developers to do the same thing is a bit like herding cats!", he says.

Are the financial markets facing a requirements crisis? Possibly.

Does the answer lie in Behaviour-Driven Development? Probably.

For more details, visit:

Hindsight	**www.hindsightsoftware.com**
Inviqa	**www.inviqa.com**
Liz Keogh	**www.lizkeogh.com**
Dan North	**www.dannorth.net**

Financial Markets Insights from The Realization Group, is a series of interviews with thought leaders in financial and capital markets. The purpose of the series is to provide exclusive insights into industry developments, through in-depth conversations with C-level executives and key experts from banks, exchanges, vendors and other firms within the financial markets ecosystem. For more information, please visit www.financialmarketsinsights.com

The Realization Group is a full service marketing and business development services company specialising in the capital markets. Our team contains industry practitioners from both the trading and post trade disciplines and we have expertise equally in the on-exchange and OTC trading environments. We apply our comprehensive set of marketing programs and wide-ranging media and business networks to complement the business development requirements of our client organisations.

www.TheRealizationGroup.com

Produced by The**Realization**Group