CHAPTER • 3

Overview of the Trading/Investment System Development Methodology

Any business process can be managed and trading/investment system design and development is no different. It can and should be managed. Too often, though, the poor quality of execution dooms projects. What financial firms need is a more systematic approach to conceive, design, develop, and manage new trading/investment systems. Prior to implementation, though, the process of development of such a system should follow a well-defined, well-documented flow of steps according to a development methodology. The organization should establish controls over the sequence and interactions, of how the output of one process acts as the input for a subsequent process; in total, forming the product realization process.

While at times new trading/investment systems may be designed, developed, built, and tested using a ready-made process, more often a new process will be needed for each new system. Learning from experience, the organization can determine what new or additional verification, validation, monitoring, inspection, and testing processes will be needed for a new trading/investment system. Top management must also determine acceptance criteria for the product, the basis or test results upon which the product will be accepted or rejected, and define what documented evidence they will expect.

The problem is, as trading/investment system development progresses, traders and financial engineers like to make changes. No matter how hard they try to freeze the design, product teams may not be able to do so. The ability of any methodology to respond to change often determines the success or failure of a trading/investment system design and development project. Plans and methodologies must be flexible and ready to adapt to changes in the market and technology. If something changes during design and development, our methodology allows product teams to revert to a previous stage and rapidly progress again.

We have designed our incremental development methodology so as to ensure a rapid development cycle with quick deliverables and consistent quality standards. Before development can begin, however, the product team must raise money to fund the development. To raise money a product team needs a Money Document.

3.1. The Money Document

Trading and money management is a business and in order to succeed, product teams need to raise research and development capital as well as trading capital, from either inside the firm or outside it. Either way, the product team will need to describe in a persuasive manner why a system can potentially be better than competing systems and worthy of seed capital. As with most business proposals, a focused, professional business plan is essential, especially in a start-up stage.

As a template, we present the Money Document, the primary deliverable before fully laying out a trading/investment system's business plan. A well-done Money Document serves as a Vision and Scope Document by outlining the business goals of a proposed trading/investment system in a clear and concise fashion in order to persuade management or outside investors (that is, collectively, seed capital investors) to provide the initial capital needed for research and subsequently for development of a trading/investment system according to our four-stage methodology. The Money Document answers the fundamental question, "Is this a business worth investing seed capital in?"

While the Money Document comes before entry into our methodology, its own development should follow an iterative process, forcing the product team to focus on building a business. The resources allocated to a project, subsequent to the delivery of a Money Document and management approval, will permit entry into our development methodology.

Our methodology owes its structure to a combination of the traditional waterfall of Royce[1] and spiral methodology of Boehm[2] from software, the Stage-Gate® methodology of Cooper[3] from new product development, and Lean Six Sigma and Agile development, combining aspects of these well-known methodologies, seeking to gain from their respective strengths and overcome their respective weaknesses.

3.2. Waterfall Methodology

Royce's traditional waterfall methodology for software development consists of four stages—analysis, design, implementation, and testing—that vaguely map to the four stages of our methodology, as you will see.

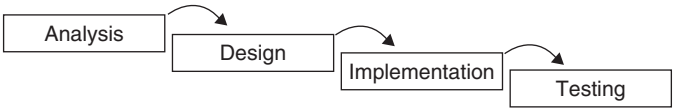

FIGURE 3-1

In sum, the waterfall methodology forces a development team to plan before building and requires a disciplined approach to development. Using the waterfall methodology, teams avoid the pitfalls of creating systems before project plans are precisely defined and approved. But, the waterfall methodology has at least two drawbacks.

The first drawback is that the waterfall methodology tends to put too much emphasis on planning; all details must be defined up front before design and implementation can begin. As a result, it leaves no room for error and no process for handling feedback on problems that inevitably occur during design and development. In the fast moving financial markets, where trading opportunities come and go, the waterfall methodology on its

own is not flexible enough to react to new information and knowledge. (Project managers and IT professionals sometimes misunderstand that trading system specifications are rarely, if ever, fixed up front.) To overcome this shortcoming of the waterfall methodology, the spiral methodology was developed.

The second drawback is that prior to progression to each new stage, the waterfall methodology does not include a gate process—a management decision as to whether to or how to continue development based upon the system's potential. For this reason, our methodology includes gates after each stage.

3.3. Spiral Methodology

In the spiral methodology, a smaller amount of time is initially devoted to the four stages—research, planning, implementation, and testing—followed by several iterations or loops over each. As the loops progress, and the spiral gets larger, development teams add more detail and refinement to the project plan. At some final level of detail, each stage will be complete.

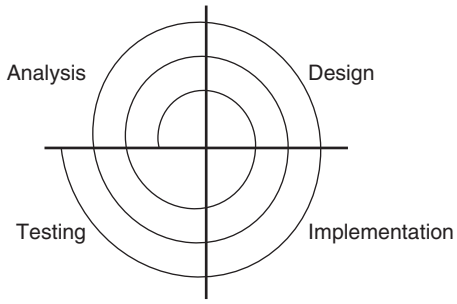

FIGURE 3-2

In this way, the spiral method allows for feedback as problems arise or as new discoveries are made. Problems can then be dealt with and corrected, unless they are fatal. So in a spiral project view, intermittent or prototype implementations provide important feedback about the viability and profitability of a trading/investment system.

As with the waterfall method, though, the spiral method is not without drawbacks. In the spiral methodology, the loops can grow without end and there are no constraints or deadlines to terminate iteration. This can lead to scope creep, a loss of project focus, messy logic, and unnecessary digressions, where the project plans may never contain a clear and concise design. (This can lead to a fuzzy front end of never-ending spirals, where researchers pursue new and better knowledge instead of working systems.) For example, spiraling has no criteria for transition from one tool set to another, say from Excel to C++. So, the looping process demands clear conditions for termination. K|V borrows from the waterfall and Stage-Gate® methodologies to overcome this weakness.

3.4. Stage-Gate® Methodology* [AQ1]

Our methodology applies concepts from the science of new product development, including idea generation and screening, business analysis, development and testing, technological implementation, to trading/investment system development. Unlike new products, though,

there are no external customers of the trading/investment system or enabling software itself. Rather, investors buy the results, or track record, of the system and its interaction with the trading team. Nevertheless, we can all benefit from an understanding of new product development methodologies, in particular Dr. Robert G. Cooper's Stage-Gate® method, from which we borrow the concept of gates. Gates are management meetings with product team members where go/kill decisions are made and where weak projects are weeded out and scarce resources are reallocated toward more promising projects.

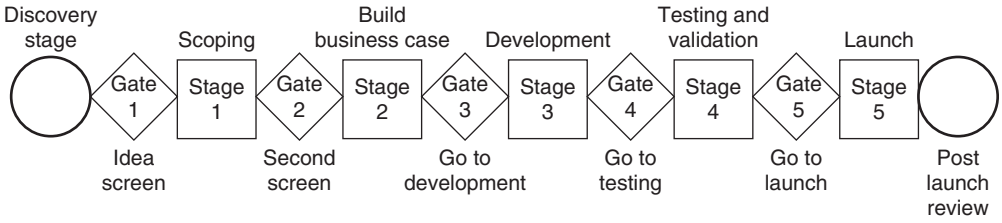

FIGURE 3-3

Gates act as checkpoints or screens along the new product development process. In our methodology, gates are an opportunity to check whether or not the business reason for developing the trading/investment system is still valid and whether additional seed capital is warranted. New in our methodology, however, is the set of gate outcomes. Where traditional gates allow only for two outcomes—either a go or kill decision—K|V allows for five potential outcomes:

1. **Go.** Go on to the next stage of the waterfall.
2. **Kill.** Kill the project entirely.
3. **Hold.** Hold development at the current stage for reconsideration at a future date.
4. **Return.** Return to a previous stage for additional research or testing.
5. **Trade.** Trade the existing prototype. (Should only be allowed for short-lived strategies, e.g., event trading algorithms.)

Well-organized gate meetings will each have a unique set of metrics or criteria for passage to the next stage. If the project is allowed to continue or is sent back to a previous stage, the gate meeting should also outline the plan for moving it through the next stage and define the deliverables expected and the criteria for evaluation at the next gate meeting. The criteria for each gate should include a check on the deliverables, minimum standards, potential for profitability, competitive advantage, technical feasibility, scalability, and risk. Gate decisions give rise to risk of error:

- **Type I gate error.** Allowing to "go" a system that either cannot be built or will not succeed in its competitive advantage.
- **Type II gate error.** Killing a system that can be built and can achieve a competitive advantage.

We consider gates to be real option expirations. When management chooses to invest seed capital in a new trading/investment system development project, it is essentially buying a call option on that project, an option that expires at the next gate meeting. At the gate meeting, management can choose to exercise the option and fund the next stage, or

allow the option to expire worthless, killing the project. This decision should be based on estimates of future cash flows and probability of those cash flows. So, gates predefine incremental releases of capital. An optional release structure limits seed capital providers' loss potential by tying capital to deliverables, predefined real option valuation techniques, and gate-passage criteria. Essentially, at each successive gate, management must make a progressively stronger commitment to the trading/investment system development project. In the end, well-organized and well-run gate meetings will allow losing projects to expire worthless, and allow worthwhile projects to proceed to the next stage.

3.5. Six Sigma, Lean, and Agile Development

Our methodology also benefits by including concepts from Six Sigma, Lean, and Agile development as well. As with Design for Six Sigma, the goal of our methodology is to drive investor needs into the product design, increasing performance and decreasing process variation. The by-products of Six Sigma are a reduction of defects and an improvement in profits and employee morale and the quality of trading/investment systems. Also, as with Lean principles, our methodology focuses on reducing waste in order to reduce production time. Like Lean Six Sigma, our methodology combines both Lean and Six Sigma to focus on both speed and quality. The goal is to build a better trading/investment system at a better cost in a shorter amount of time.

Lastly, as with Agile software development methodologies, our methodology attempts to compromise between too much process and no process, welcoming change by allowing for reversions to previous stages and minimizing risk through iterative prototyping. Furthermore, our methodology emphasizes real-time face-to-face communication through team-based development.

3.6. Trading/Investment System Development Methodology

As we have said, in order to overcome the respective shortcomings of each methodology, we combine them into a single paradigm for trading/investment system design, development, and management. Our four stages progress in a waterfall design-test-implement-manage (DTIM) framework, but within each stage four steps are connected in a spiral structure. The activities of each timeboxed spiral are organized into a four step plan-benchmark-do-check framework focusing in order: quantitative methods, data, technology, and risk management, respectively.

Note that our plan-benchmark-do-check framework differs from the traditional plan-do-check-act methodology from quality due to the heavy research component in trading system development. Benchmarking consists of critical comparison of available quantitative methods, data cleaning algorithms, technological implementation, and risk management methods that will yield a competitive advantage. Unlike standard software or manufacturing models, where there are clearly defined methods and goals up front, in trading system development the methods and goals are fuzzy, or poorly defined, and solutions will be highly complex. Most of these solutions need to be researched and replicated prior to moving forward. Furthermore, without benchmarking, firms cannot know if methods either derived in-house, or those provided by vendors, are correct.

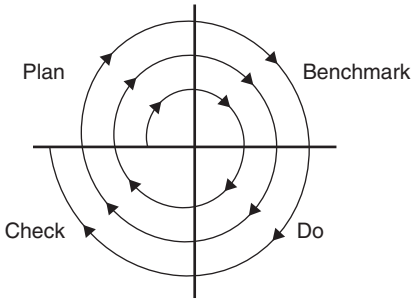

FIGURE 3-4

- **Plan.** Determine the problem to be solved, gather information, and then plan and document a course of action to solve it (i.e., what do we need to do?).
- **Benchmark.** Research and compare alternative solutions to arrive at the best practice (i.e., what is the best way to do it?). Over the four stages of our methodology, we will benchmark quantitative methods, data cleaning and optimization algorithms, technology, and portfolio and risk management processes.
- **Do.** Carry out the best practice course of action (i.e., we do it).
- **Check.** Check to see if the desired results were achieved along with what, if anything, went wrong, and document what was learned (i.e., how did we do?). If results are not satisfactory, repeat the cycle using knowledge gained.

The four-stage spirals consist of, for the purposes of this book, three loops. (In practice, they may consist of more or fewer as needed.) Each loop consists of one pass over each of the steps in the stage spiral.

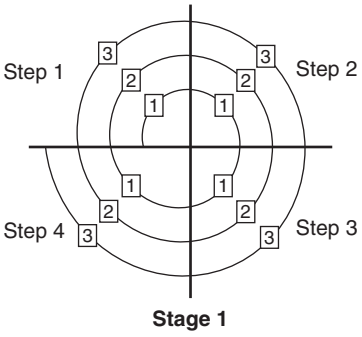

FIGURE 3-5

(We will refer to the activities of each stage, each step, and each loop in the following way:

K|V *Stage Number.Step Number.Loop Number*

For example, the shorthand K|V 1.3.2 would refer to Stage 1, Step 3, Loop 2.)

Again, at the completion of each stage is a gate that will allow top management to kill the project, return to a previous stage, hold until some future time, continue to the next

stage of development, or trade the existing prototype. After completing the fourth and final stage, the methodology requires that we repeat the entire four-stage waterfall for continuous improvement.

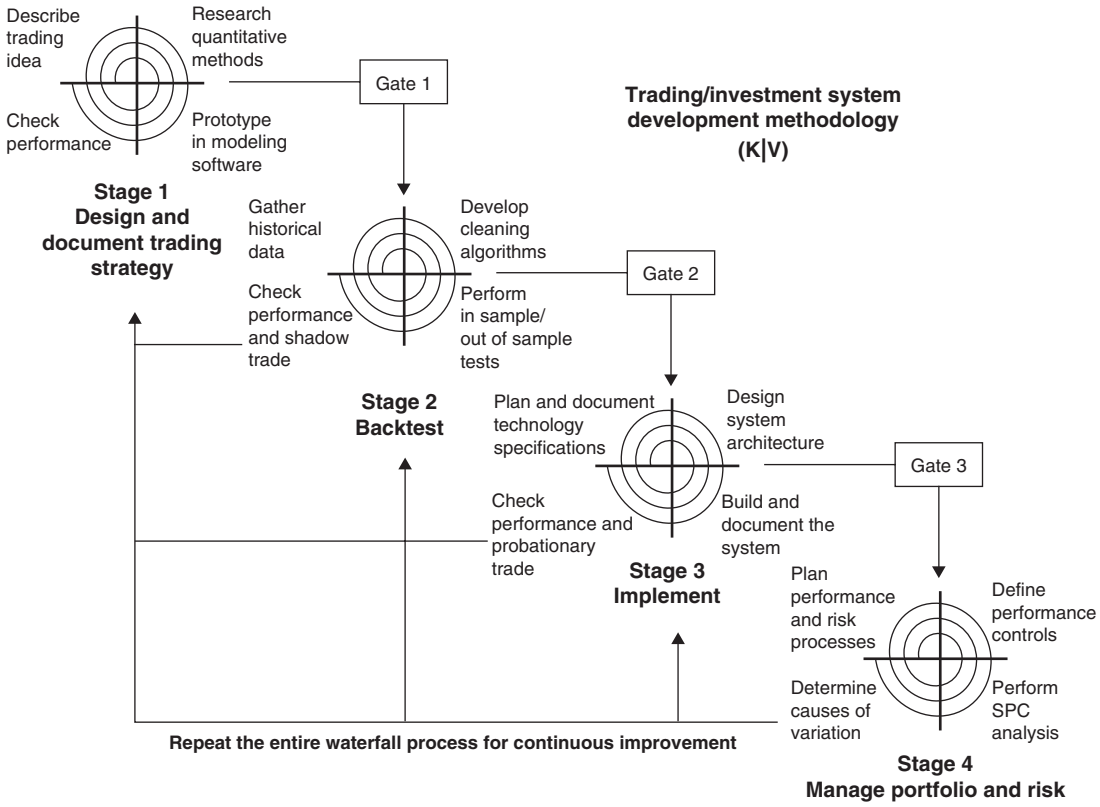

FIGURE 3-6

Here are the four stages of our methodology and their respective components:

Stage 1. Design and document trading/investment strategy

1. Describe trading/investment idea (K|V 1.1)
2. Research quantitative methods (K|V 1.2)
3. Prototype in modeling software (K|V 1.3)
4. Check performance (K|V 1.4)

Gate 1
Stage 2. Backtest

1. Gather historical data (K|V 2.1)
2. Develop cleaning algorithms (K|V 2.2)

3. Perform in-sample/out-of-sample tests (K|V 2.3)
4. Check performance and shadow trade (K|V 2.4)

Gate 2
Stage 3. Implement

1. Plan and document technology specifications (K|V 3.1)
2. Design system architecture (K|V 3.2)
3. Build and document the system (K|V 3.3)
4. Check performance and probationary trade (K|V 3.4)

Gate 3
Stage 4. Manage portfolio and risk

1. Plan performance and risk processes (K|V 4.1)
2. Define performance controls (K|V 4.2)
3. Perform SPC analysis (K|V 4.3)
4. Determine causes of variation (K|V 4.4)

Repeat the entire waterfall process for continuous improvement

The rest of this book is dedicated to developing trading/investment systems according to these 16 steps, but first we will provide a brief overview of each.

3.7. Design and Document Trading/Investment Strategy (Chapter 7)

It is more fun to do than to plan. This very human trait is only driven out by years of schooling and experience. The problem with planning in the financial markets is twofold; one, most traders prefer to trade rather than plan; and two, most planners never get the opportunity to trade since the management in financial firms generally rises from the trading ranks. Consequently, proprietary trading and money management firms tend to optimize for the short term, and wind up building trading/investment systems that are not unique, making it difficult to persuade investors to invest in the new strategy and not existing ones with long track records instead.

These types of systems, however, do not generally result in maintainable excess returns and are very often not scalable. This could explain why the vast majority of money managers underperform their benchmark index and why so many hedge funds close each year for underperformance. Complex systems that will generate long-term returns are built one step at a time and evolve along the way as new knowledge is gained.

3.7.1. Describe Trading/Investment Idea (Chapter 8)

This first step is often the hardest one. The more complex the trading idea, the more difficult it becomes to plan and communicate it clearly. Before any development on a system begins, we must first be able to fully articulate the business logic and quantitative methods

of the system. We have, though, already started the description process—the description created for the Money Document will serve as a starting point. Over the iterative research process, planning each loop will require team members to define goals and set boundaries for that research.

3.7.2. Research Quantitative Methods (Chapter 9)

Very rarely do we dream up completely new trading/investment ideas. Rather, we build on ideas of the past and add new twists. Many times new trading strategies are essentially copies of old ones that proved to be successful.

The next step is to research and benchmark the relevant mathematical and logical models, which may include deriving proprietary algorithms and/or applying publicly available research from journals, books, the Internet, or white papers. The research process may also include gaining an understanding of the trade selection and execution methodologies of other successful systems. The goal of research is to speed and refine our path to the best algorithms, which will form the basis of a Business Rules Catalog for the system. Building and maintaining a proprietary library of unique quantitative methods is key to the long-term success of a firm.

3.7.3. Prototype in Modeling Software (Chapter 10)

As we have said, despite its shortcomings, Excel as well as other modeling software (such as Resolver, MATLAB, Mathematica, or Zack's) are rapid development environments for testing trade selection and execution algorithms. The goals of prototyping are to quickly build several generations in order to evaluate whether a particular idea warrants further investigation and to promote risk-based iterative development, where the hardest pieces of the project are investigated first.

Currently, many trading/investment ideas are immediately built as working systems if confidence is high that the trader and/or the system will make money. This type of development often proceeds without proper definition prior to implementation despite understood software risks. Prototyping algorithms enables us to clearly define algorithms, GUI and data requirements, and develop a baseline application for regression testing. These prototypes will form a foundation of the Technology Requirements Specification document in later stages. Baseline regression with prototypes enhances defect detection and removal more than any other strategy.

3.7.4. Check Performance (Chapter 11)

We will use prototypes as a starting point in the discussion of how we are going to measure the performance of the system. Without a clear plan on how to measure performance and define success and the variability of the trading system, we cannot proceed to the next stage.

We define three types of trading/investment systems and each type will require a unique mechanism for performance testing. The three foundation types are:

1. **Trigger systems.** A good example of trigger trading/investment systems are ones built on technical or valuation indicators.

2. **Filter systems.** This is the traditional system for mutual funds.
3. **Signal strength systems.** These systems are typically highly quantitative in nature and use blending or regression algorithms.

(In reality, few trading/investment systems fit neatly into a single, simple category; more complex systems combine aspects of two or even all three. Nevertheless, our methodology applies to all types, simple and complex.)

If at this first checkpoint performance measurement shows system failure, the methodology necessitates a looping back to previous stages. The goal is to quickly stop development on trading/investment systems with a low probability of success.

3.8. Gate 1 (Chapter 12)

In order to pass through a gate, several questions need to be answered and as we will see Gate 1 has several such questions. If the questions are answered to the satisfaction of the customer (usually top management), development will be allowed to proceed to Stage 2.

This gate will prevent development of the trading/investment system from moving to the backtesting stage until the required activities and deliverables have been completed in a quality manner. Furthermore, at the gate meeting we will chart the path ahead by ensuring that plans and budgets have been made for the backtesting stage.

For the remainder of the book we will use the term "well defined" to mean that a trading/investment system has passed through this first gate. The implicit assumption is that the methodology has been rigorously followed in a quality fashion.

3.9. Backtest (Chapter 13)

Complete system analysis necessitates research into and optimization over past market movements as a way to analyze and validate the system—a process called backtesting. A backtest is a simulation and statistical analysis of a trading/investment system's inputs and outputs against historical data and would be a unique process for each system. A backtest will prove the capability of the trading/investment system to meet investor requirements and is based on statistical measures. Such proof will demonstrate that the system exceeds the traditional buy and hold sample paths.

3.9.1. Gather Historical Data (Chapter 14)

Once the initial prototype has shown a system to be worthy of further investment of time and resources, the real task of backtesting begins. Prior to building and implementing the system, we must test it over a relatively large set of historical data and preferably for a large sample of instruments. As a result, firms build a customized database of historical data and purchase or build a software tool that allows for proper backtesting of the system.

While it may seem elementary, planning and investigating the availability of data is very important. Required data may either not exist at all or is prohibitively expensive based upon the prospective returns of the trading/investment system.

3.9.2. Develop Cleaning Algorithms (Chapter 15)

One of the major obstacles to building a profitable trading/investment system is dirty data. Virtually all data contains errors, outliers, which may or may not be errors, and point in time problems. Even data purchased from reliable vendors has errors. As a result, the identification and removal or correction of errors and known issues in the calculation data prior to optimizing the trade selection and position management algorithms is very important. Development of a Data Transformation Management System (DTMS) that will scan data for errors and irregularities is essential.

A DTMS should implement data cleaning algorithms that can operate on live-time as well as historical data. Algorithms that cannot be run in real time prior to trade selection should not be used on historical data or the cleaned, historical data will skew the results. The benchmarked cleaning algorithms will be added to the Business Rules Catalog begun in Stage 1.

3.9.3. Perform In-Sample/Out-of-Sample Tests (Chapter 16)

Performing a proper in-sample/out-of-sample test is perhaps the most critical step in the process. Financial engineers are keenly aware of the extent to which in-sample results may differ from out-of-sample results and trading/investment algorithms must be examined against both before progressing to the implementation stage. A well-developed, optimized system will perform similarly out of sample as it does in sample, so it is of utmost importance to hold back some historical data for out-of-sample testing. Such a test will result in one of three outcomes for a trading/investment system:

1. Profitable both in sample and out of sample.
2. Profitable in sample, but not out of sample.
3. Unprofitable both in sample and out of sample.

If the system is profitable both in sample and out of sample, it will very likely receive capital to begin implementation and trading as soon as possible. If a system is profitable only in sample, it will likely be allocated additional resources for continued research. If, however, the system proves to be unprofitable both in sample as well as out of sample, it will likely be scrapped altogether.

3.9.4. Check Performance and Probationary Trade (Chapter 17)

At this point, we acknowledge that traders may very well demand to trade the prototype. However, it should be understood that the only purpose this should serve is to fully understand the performance monitoring tools that will be required by management in the later stages. Be aware that the performance of shadow trading may not be indicative of the performance of the completed system, largely due to the lack of SPC controls placed around the prototype system. The lack of controls at this stage can lead to, for example, overweighted sector bets.

Shadow trading may also occur in order to more fully understand the behavior of the system and as mentioned to understand what reporting tools management will need. As before, checking the performance of the system will prevent additional time and

resources from being spent on unprofitable projects. We may need to loop back to the initial research stage and reassess the quantitative methods and algorithms.

3.10. Gate 2 (Chapter 18)

As with Gate 1, Gate 2 has several questions we must answer before proceeding to Stage 3. If the passage criteria are met, to the satisfaction of seed investors, they will release capital and development will proceed to Stage 3.

This gate will prevent development of the trading/investment system from moving to the implementation stage until the required backtesting activities and deliverables have been completed in a quality manner. Furthermore, at the gate meeting we will chart the path ahead by ensuring that plans and budgets have been made for the implementation stage.

3.11. Implement (Chapter 19)

Building a fully or even partially automated trading/investment system is in many respects a software development project. As a result, implementation will require:

1. Connectivity between and interoperability with disparate software and hardware systems for trade execution.
2. Object-oriented software design to encapsulate trading logic.
3. Other processes such as optimization and data storage.

This will require creation of plans and blueprints before programming in a language like C# or C++ begins.

3.11.1. Plan and Document Technology Specifications (Chapter 20)

The purpose of technology specifications, such as the IEEE Software Requirements Specification (SRS), is to fully define the functionalities and performance requirements of the trading/investment system. The specification documents allow the team of hardware engineers and programmers to quickly build the system with the correct functionalities and to the proper specifications. As with all high-level planning documents in engineering, we expect these to be revised in an orderly process as we spiral through the development. (This is no different to the revision process of quality management using ISO 9000.)

Prior to constructing a trading/investment system we will have essentially completed a project Vision and Scope Document in the form of a Money Document. The specifications include all of the required features and functionalities and all of the items that hardware and software engineers will need to design, build, and modify the production system. Further, the specifications should clearly define the steps for the project along with documenting all of the detailed information about the network and hardware requirements, trading algorithms, including data dictionary, data flow maps, GUI requirements, error handling maps, and report generation maps. Fortunately, much of this work will

already be done and the specification documents will largely be based upon the prototypes and descriptions created in earlier stages of development.

3.11.2. Design System Architecture (Chapter 21)

Architecture documents are blueprints of the hardware and software that will form the architecture of the trading/investment system, including the financial calculations, real-time data and user interfaces, order routing connections, reporting functionalities, and any other necessary processes.

As with architecture, the bigger and more complex the building or trading system, the more important blueprints are to the success of the project. The more complex a trading/investment system becomes, the more important it is to create detailed architectural plans, using an agreed-upon set of notations, as with the Unified Modeling Language (UML), which enables project designers and programmers to communicate the details of system design. Through the use of a common notation, problems can be solved in an object-oriented way before programming begins. As much of the technological implementation depends on interoperability and connectivity, glue code, computer programming code that is written to connect reused commercial off-the-shelf applications, should be well documented in the architecture documents.

3.11.3. Build and Document the System (Chapter 22)

Once the hardware is built and network connections are completed, the process of construction will be a step-by-step march through the architecture documents. Since the architecture documents themselves will be evolving as the team spirals through implementation, the ideal solution is to continue to add and refine it as the system is built.

The data maps will continue to grow along with adding new dictionaries to clearly show both the calculation of each field and where it came from and where it is used. A GUI section also will grow to include a screen shot of each form. The error handling section will continue to grow to list all known open issues.

The product team should produce a user manual that will allow a junior trader to operate the system and a junior programmer to maintain it. (Actually, the product team should be writing this manual over the course of the entire development process. By this stage and step, the production of a user manual should be a process of assembling already-written documents.) While junior-level people are not overseeing the system, the documentation should be placed at that level. It is assumed that since the systems we are building are proprietary in nature we will not need commercial-level user documentation. Documentation should be treated as a trade secret so staff external to the product team cannot steal its intellectual property.

3.11.4. Check Performance and Probationary Trade (Chapter 23)

Once the completed system has been built, probationary trading can progress to find any design flaws in the trading algorithm or the technology prior to trading the full investment sum, officially beginning the track record period, or managing customer funds. The second

purpose of probationary trading is to allow the trader or money manager time to use the trading tools and determine what additional reporting tools need to be built to properly manage the embedded risks. Management tools should contain, for instance, displays of underlying data for calculations and trade reports. At this point, the performance of the trading/investment system will be similar to those of the final product.

3.12. Gate 3 (Chapter 24)

As with Gates 1 and 2, Gate 3 has several questions the product team must answer before proceeding to Stage 4. If the questions are answered to the satisfaction of the customer, development will be allowed to proceed.

This gate will prevent development of the trading/investment system from moving to the final stage until the required activities and deliverables have been completed in a quality manner. Furthermore, at the gate meeting, management will chart the path ahead by ensuring that plans and budgets have been made for the next stage. Approval at this gate permits full investment and trading of the system.

3.13. Manage Portfolio and Risk (Chapter 25)

Portfolios of securities and derivatives require constant monitoring and so successful implementation of a trading/investment system necessitates that periodic reports be generated to show the performance of the working system. These reports will present the portfolio statistics and performance metrics, risk calculations, and provide documentation of the attribution of gains and losses. Traders are notorious for turning off systems right before they become profitable and leveraging up after positive statistical anomalies with strong mean reversion, which is why SPC is a critical tool. Reports should present a proper determination of the root causes of variation from the expected results, that is, nonconformance, and an action plan to deal with those variations, that is, corrective actions.

3.13.1. Plan Performance and Risk Processes (Chapter 26)

Markets are stochastic and trading/investment system performance will be stochastic as well. Stochastic processes should be measured and monitored by tools built for the purpose. The product team, with risk management, must plan for a system of monitoring and reporting portfolio statistics, performance metrics, and risk factors. Essentially, these risk control techniques will help the team understand whether or not the system is working within specifications and in conformance with the backtest and/or a benchmark.

3.13.2. Define Performance Controls (Chapter 27)

To monitor a working system, automated processes should keep track of the system's performance metrics. These will be valuable when reevaluating the underlying premise for the system relative to the performance experienced during backtesting.

Furthermore, every system should have a benchmark. Using a portfolio attribution system the product team can clearly identify the performance relative to the benchmark. So, to make sure that the trading/investment system consistently outperforms its benchmark the team must perform attribution analysis on the portfolio as well as on the benchmark itself.

3.13.3. Perform SPC Analysis (Chapter 28)

Risk management is effectively the same as statistical process control in manufacturing industries. Are the underlying market distributions the same as the ones that were used to generate returns for the backtest? Is the system in conformance with the backtest and the benchmark index? Or are the inputs or outputs of the process different? Risk calculations and reports will give top management a snapshot of performance, and potential losses and drawdowns both on an absolute basis and relative to the benchmark over a given time horizon. However, of course, while methods for dealing with extraordinary occurrences may be built into a trading/investment system, gaps may render them useless.

3.13.4. Determine Causes of Variation (Chapter 29)

After the attribution analysis is completed and the product team understands all of the bets the system is placing to beat the benchmark, the team must build one final set of tools. These tools are based on process control theory—namely, quality, statistics, ANOVA, and design of experiments.

The goal of these tools is to determine the causes of variation, or nonconformance, in the trading/investment system's performance. If a process goes out of control, then a cause for that condition can be found. If the team can find the cause of the process being out of control, then they can fix the process and theoretically experience less variance than the benchmark going forward.

Profitable trading/investment systems have life cycles. Eventually, the market will close the door on every trade. So, systems will need to be continuously tweaked and eventually scrapped. The goal is to quickly stop trading systems that lose their edge before they cause large losses.

3.14. Repeat the Entire Waterfall Process for Continuous Improvement (Kaizen) (Chapter 30)

Think of the steps in our methodology as a continuous, never ending spiral. Once the product team gets to the end, they start again to improve the system with new refinements or create new ideas altogether.

Top management is responsible for cultivating a professional environment that promotes continuous improvement through an ongoing effort to improve trading/investment system performance, increase efficiency, and reduce costs. Product teams should focus on continuously making small improvements to the trade selection, data cleaning, order management, and risk management processes as well as the enabling technologies, until the life cycle of the trading/investment system runs its course. A firmwide culture of sustained

continuous improvement will concentrate efforts on eliminating waste in all processes of a trading organization.

Through small innovations from research and entrepreneurial activity, trading and money management firms can discover breakthrough ideas. These include, among other things, the creation of new trade selection algorithms, application of existing systems to new markets, and the implementation of new technologies for more efficient trade execution. Through continuous improvement, trading firms can extend the maturity stage of the trading/investment system life cycle.

Author query

[AQ1] what does the asterisk refer to?