Driving Business Value in Industrial Innovation

By Augustine Tibazarwa

ISA™
Notice

The information presented in this publication is for the general education of the reader. Because nei-
ther the author nor the publisher has any control over the use of the information by the reader, both the
author and the publisher disclaim any and all liability of any kind arising out of such use. The reader is
expected to exercise sound professional judgment in using any of the information presented in a particu-
lar application.

Additionally, neither the author nor the publisher has investigated or considered the effect of any
patents on the ability of the reader to use any of the information in a particular application. The reader is
responsible for reviewing any possible patents that may affect any particular use of the information pre-
sented.

Any references to commercial products in the work are cited as examples only. Neither the author
nor the publisher endorses any referenced commercial product. Any trademarks or tradenames refer-
cenced belong to the respective owner of the mark or name. Neither the author nor the publisher makes
any representation regarding the availability of any referenced commercial product at any time. The
manufacturer’s instructions on the use of any commercial product must be followed at all times, even if
in conflict with the information in this publication.

The opinions expressed in this book are the author’s own and do not reflect the view of the Interna-
tional Society of Automation.

Copyright © 2017 International Society of Automation (ISA)
All rights reserved.

Printed in the United States of America.
10 9 8 7 6 5 4 3 2

ISBN: 978-1-941546-78-9

No part of this work may be reproduced, stored in a retrieval system, or transmitted in any form or by
any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written per-
misson of the publisher.

ISA
67 T. W. Alexander Drive
P.O. Box 12277
Research Triangle Park, NC 27709

Library of Congress Cataloging-in-Publication Data in process
# Contents

## Chapter 1 Introduction
- Innovation Agility ......................................................... 4
- Why Disciplined Agility? .................................................. 8
- Why Product Development? ................................................ 9
- Engineering Management Perspective .............................. 9
- Summary .............................................................................. 10

## Chapter 2 Process Manufacturing Industry
- Manufacturer Business Perspective of Automation and Control 13
  - Demand-Side Threats from the Manufacturing Industry ........ 13
  - Supply-Side Threats within the Automation Supply Segment 14
  - Manufacturer Requirements for Automation ....................... 15
  - Deficiency in Domain Expertise ........................................ 16
  - Un-Met Manufacturer Needs ............................................. 17
- Obtaining Manufacturing Business Value from Innovation ....... 18
  - Increasing Revenue .......................................................... 18
  - Reduce Operating Cost .................................................... 20
  - Improve Competitiveness .................................................. 22
  - Reduce Liability Exposure ............................................... 24
- Summary .............................................................................. 25

## Chapter 3 Automation and Control Business Drivers for Manufacturing
- Economic Performance ....................................................... 31
- Analysis of Financial Ratios ............................................... 31
- Business Value .................................................................... 33
- Return on Investment ......................................................... 37
- Automation of Enterprise Processes .................................... 37
Chapter 4 Innovation Supply Chain .............................................. 41
  Design and IP ................................................................. 42
  Technology Supply Chain Management ................................. 45
    Value Chain .............................................................. 46
    Value Capture .......................................................... 48
    Value Creation .......................................................... 51
  Ecosystem Management .................................................... 55
  Safety-Critical Supply Chain ............................................. 57
    Product Lifetime .......................................................... 63
    R&D Cost ................................................................. 64
  Summary ................................................................. 66

Chapter 5 Innovation Framework .............................................. 67
  Innovation Delivery Engine .............................................. 68
    Value Alignment .......................................................... 70
    Value Realization .......................................................... 70
    Value Instrumentation .................................................... 71
    Value Optimization ...................................................... 71
    Value Creation ............................................................ 72
  Innovation-Centric Engineering ....................................... 73
  Engineering for Manufacturing Innovation .......................... 75
  System Architecting: Design for Business Value ...................... 78
    Good System Architecture Principles .................................. 84
    Principle: Empathic Design .............................................. 85
    Principle: Complexity .................................................... 87
    Principle: Fidelity to Dominant Design ................................ 89
    Principle: Value Chain Protection ..................................... 91
    Principle: Robustness ..................................................... 93
    Principle: Product Quality ................................................ 94
    Principle: Reliability and Availability .................................. 96
  Principle: Extensibility, Maintainability, and Upgradeability .... 98
  Principles of System Architecting ....................................... 99
    Principle: First Principle of System Architecting .................. 99
    Principle: Needs Analysis ............................................... 100
    Principle: Level N Analysis for Business Flows .................... 103
    Principle: Concept Synthesis and Selection ......................... 104
    Principle: Simplicity, Unity, Clarity ................................... 106
    Principle: Designing for Robustness .................................. 108
    Principle: Branding and Brand Positioning .......................... 110
    Principle: Liability and Warranty Accounting ....................... 112
  Principle: Innovating .................................................... 114
  Summary ................................................................. 116
Chapter 6  Case Study: Product Development Practices ............................. 117
  Background ................................................................................. 117
  Process Practices ...................................................................... 119
    Case Study Design .................................................................. 119
    Product Development Practices ............................................. 120
  Knowledge and Process Architecture ....................................... 124
    Case Study Design .................................................................. 126
    Results: Explicit Formal Knowledge ....................................... 127
    Knowledge Loops .................................................................... 128
  Conclusion .................................................................................. 132

Chapter 7  Agile Industrial Innovation .............................................. 133
  Anatomy of Delivery Excellence ............................................... 133
    Throughput Contribution Analysis ....................................... 134
    Demand Velocity and Delivery Velocity ................................. 138
    Innovation Agility ................................................................. 150
    Organization for Agility ......................................................... 152
    Agile Principles for Process Automation ............................... 153
    Agile Practices for Process Automation ................................. 156
  Scalable Agility ......................................................................... 161
    Agile Practice for Large Organizations .................................. 161
    Agile Practices for Large Systems ......................................... 163
    People Aspects of Agility ....................................................... 165
  Risk Balancing for Agility ....................................................... 166
    Iterative and Plan-Driven ....................................................... 167
    Waterfall and Plan-Driven ..................................................... 169
    Waterfall and Agile .............................................................. 170
    Iterative and Agile ............................................................... 171
  Summary .................................................................................... 172

Chapter 8  R&D Agility Transition .................................................... 173
  Alignment for Agility: Metrics .................................................. 176
    Metrics Tetra Transforms ...................................................... 178
  Effectiveness and Efficiency .................................................... 185
    Process Change Management: Adopting Agile Methods ........ 191
    Analysis of Effective and Efficient Teams ............................ 195
  Summary .................................................................................... 198

Chapter 9  Executive Leadership for Innovation ............................... 199
  Market ....................................................................................... 200
  Technology ............................................................................... 201
  Knowledge Workers ................................................................. 201
  Managing Uncertainty .............................................................. 202
  Technology Strategy ................................................................. 204
  Institutionalized Innovation ...................................................... 204
  Products versus Services ........................................................... 204
<table>
<thead>
<tr>
<th>Chapter 10 Disciplined Agility</th>
<th>211</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: Select Business Drivers for Change</td>
<td>213</td>
</tr>
<tr>
<td>Step 2: Select the Most Suitable Agile Practices</td>
<td>216</td>
</tr>
<tr>
<td>Step 3: Total Agility Transformation</td>
<td>218</td>
</tr>
<tr>
<td>Step 4: Prerequisites for Agility</td>
<td>221</td>
</tr>
<tr>
<td>Step 5: Operational Agility</td>
<td>222</td>
</tr>
<tr>
<td>Summary</td>
<td>222</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 11 Emerging Frontier for Industrial Innovation</th>
<th>225</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital Plant: Automation Value Proposition</td>
<td>228</td>
</tr>
<tr>
<td>Internet of Things: Capital Equipment Life Cycle</td>
<td>230</td>
</tr>
<tr>
<td>Capital Investment Analysis</td>
<td>233</td>
</tr>
<tr>
<td>Integrated Engineering (Multi-Discipline)</td>
<td>234</td>
</tr>
<tr>
<td>Capital Equipment Life-Cycle Management</td>
<td>235</td>
</tr>
<tr>
<td>STAMP: Eliminating Safety Incidents</td>
<td>237</td>
</tr>
<tr>
<td>Systems Theoretic Accident Modeling</td>
<td>237</td>
</tr>
<tr>
<td>Incident Simulation and Cognitive Learning</td>
<td>237</td>
</tr>
<tr>
<td>App Store: Continuous Value Delivery</td>
<td>238</td>
</tr>
<tr>
<td>Summary</td>
<td>240</td>
</tr>
</tbody>
</table>

| Appendix A Engineering Standards Knowledge Clusters | 241 |

| Bibliography | 247 |
Manufacturer needs for automation system capabilities are evolving quickly due to a fast-changing manufacturing business environment, while the value proposition of manufacturing automation systems is constantly being questioned. The maturity of the industrial technology sector means that process automation suppliers must continually shorten development cycles, continually release automation enhancing features, and relentlessly address production costs. Manufacturing automation suppliers need high-velocity innovation, powered by agility and sustained by operational excellence, to deliver business enabling products and services to manufacturers. Figure 7-1 captures the principal business considerations and factors for agile innovation of industrial technology, such as manufacturing automation systems.

**Anatomy of Delivery Excellence**

The contemporary landscape of technology innovation is governed by one overarching factor: the ability to deliver, measured in terms of speed and feature set. For automation suppliers in a mature market, delivery excellence bestows a critical competitive advantage. Existing customers can be lured away from competitors with fast response to unfulfilled needs, while new customers from emerging economies can be reached first.

While the importance of delivery excellence is well appreciated, poor understanding of this presumed-simple concept has led many product development organizations to fail in their initiatives to speed up innovation. Other indus-
tries, such as the automotive industry, have succeeded in making significant strides improving their speed and quality of innovation. A careful look at some of these success stories reveals the following about the improvement systems responsible for the turnaround in innovation performance:

- Data-driven, with a focus on understanding a specific company (Toyota for Toyota Production System, Motorola for Six Sigma)

- Knowledge-driven and observation-based, with process redesign informed by perspective from individuals involved directly in all steps of the innovation throughput stream

**Throughput Contribution Analysis**

Product development throughput at a major process automation supplier was analyzed by looking at the contributions of design, development, and testing for a project that was primarily software development, with limited embedded software for a hardware module. It was a multi-year, multi-geography (three continents), multi-time-zone project involving almost 100 people, including engineers, managers, and personnel from Marketing, Quality Assurance, Customer Service, and Training. The analysis looked at the labor hours that had been charged by everyone against the project, and included
hours for contract labor. The project used a plan-driven and waterfall development process.

Figure 7-2 summarizes the cumulative progress made by the major effort-contributing disciplines: Implementation/Construction, Functional Testing, Integration Testing and System Testing. Figure 7-3 plots the cumulative effort contribution within the development team, comparing the contribution of design/implementation/coding/construction versus functional testing. Figure 7-4 shows the count of people participating in the project over time.

**Synopsis #7.1.1:** Effort on implementation/construction is constant throughout the project. Effort on functional testing seems to track implementation/construction closely.

**Synopsis #7.1.2:** System testing occurs in steps, with the largest increase in effort occurring just after the mid-point of total elapsed time.

**Synopsis #7.1.3:** Effort on integration testing shows step increases that coincide with those for system testing.

*Figure 7-2. Multi-Discipline Work Progress*
Synopsis #7.2.1: About 30% of Development Team's effort goes to design/implementation/ construction/coding, and about 40% is spent on testing.

Synopsis #7.2.2: Rework (reproducing bugs, fixing bugs and validating fixes) takes about 30% of the total effort.

Synopsis #7.2.3: About 40% of Development Team's effort occurs in the second half of the project.

**Figure 7-3. Progress Development Team**

Synopsis #7.3.1: Participation headcount peaks early in the project, and is followed by a downward trend.

Synopsis #7.3.2: Participation headcount shows spikes that do not necessarily coincide with increase in total effort.

**Figure 7-4. People Participation**
RESULTS #1:

- The overall functional testing effort is similar to the implementation/construction effort. Moreover, efforts for both disciplines are near constant throughout the project. Although a waterfall development process was used, developers and functional testers worked fully in parallel throughout the project, which is typically expected in Agile development.

- System testing and integration testing are done in two steps.

RESULTS #2:

- The second (and largest) step change in the system testing and integration testing effort occurred when implementation/construction and functional testing were at 75% of the effort. This suggests that the surge in system testing and integration testing effort may have uncovered deficiencies that significantly lengthened the project duration, perhaps by as much as 33% (= 25% / 75%). Higher project throughput could have been achieved with iterations that ensured multiple opportunities for release readiness and that forced some system testing and integration testing to be executed and completed sooner and in multiple steps.

RESULTS #3:

- In this project, 30% of product development time was spent on rework with significant effort occurring during the second half of the project. Higher project throughput could have been achieved by avoiding rework.

- Design, implementation, and functional testing were concurrent, but they were not concurrent with integration testing or system testing. Any substantial improvement in project throughput could therefore have come from making system testing and/or integration testing concurrent with implementation/construction and functional testing.

The data shows that concurrency already exists between developers and functional testers. Further concurrency would require making additional investments in system testing for more resources and a higher-capability infrastructure to manage the team sharing and collaboration with customer-
facing stakeholders. In addition, just as with Lean transformation, the investment would have to be ongoing to preserve the higher level of performance.

To improve project throughput, the project team attempted to manage elapsed time by identifying fewer tasks to be done and by reducing the effort needed for tasks, for example, through the reuse of code. While reusing code is fraught with many problems, reusing higher-level intellectual property (e.g., use cases or system engineering design) has quantifiable benefits. Valerdi et al. (2009) describe the benefits of system engineering reuse, and provide a cost-model, COSYSMO 2.0, that can be used to estimate the cost savings from reuse.

Figure 7-2 gives some insight into why managing elapsed time might not always lead to the expected outcome. Some key contributions to project elapsed time are nonlinear. The expectation (philosophy) had been that the start of system testing marks the end of design and development. However, design, development, and functional testing show no decline in effort until system testing is complete.

**Demand Velocity and Delivery Velocity**

Business objectives for manufacturing process automation suppliers constantly exert pressure for higher development throughput. This is often conveyed as a requirement for shorter time-to-market or a faster development cycle; and it has led to a focus on doing design and development in less elapsed time. However, focus on either speed alone or elapsed time alone may not lead to sustainable market success. The delivered features and capabilities must also be aligned with the market expectations. Emphasis on speed alone may result in a focus on internal indicators, with poor alignment to market feature needs. Emphasis on elapsed time alone may result in misalignment with market availability expectations and organizational readiness for release.

The suppliers’ executive leadership requires that the product development organization delivers products to meet the business objectives based on demand. This requirement can be formalized as:

\[
\text{Demand Velocity} \leq \text{Delivery Velocity}
\]

- **Demand Velocity** – This is the demand rate of features and capabilities by the market. This rate is ratified by executive leadership for alignment with the company’s business objectives. Executive
leadership must ensure that this rate takes into account the upper limit on resources available and that it has the correct reading of the market’s needs, market direction, and critical time-windows to launch product offerings.

- **Delivery Velocity** – Formally, this is defined as the number of features developed divided by elapsed time. Product development delivery has three key components: R&D speed, integration speed, and release readiness speed.
  
  - **R&D Speed** – The rate at which a small, cohesive, and empowered team of peers can design and develop a defect-free significant feature or capability.
  
  - **Integration Speed** – The rate at which R&D combines features and capabilities to create subsystems and a system with greater business value than that of the individual features and capabilities.
  
  - **Release Readiness Speed** – The rate at which the enterprise works cross-functionally to validate new offerings and prepares to release them to market, including the creation of market-launch collateral materials.

Table 7-1 summarizes the formal relationship between delivery velocity and R&D speed, and that between loss and recovery in delivery velocity.

When signs of delays in delivery emerge, executive leadership must step in to ensure there is a recovery plan to bring the delivery trajectory back on track. Any loss in delivery velocity requires a recovery velocity matched with the appropriate recovery timeframe. The delivery-velocity loss-recovery rule is defined by Equation 7-1, based on \( V_0 \), the demand velocity driven by the company’s business objectives.
Executive leadership is responsible for any business impact resulting from loss in project velocity. Using Equation 7-1, executive leadership can confirm whether the proposed recovery timeframe \( T_1 \) is supported by a recovery velocity \( V_1 \) that is supported by the company’s capacity for all business objectives, not just those of the project. Figure 7-5 provides additional insight into how senior management can intervene to help recover from a loss in delivery velocity. The lines for \( t=4 \) and \( t=8 \) show the necessary recovery velocity and time adjustments when intervention takes place at \( t=4 \) and \( t=8 \), sometime before the expected time-to-market. As shown, the required recovery velocity and time-to-market adjustments are much less the earlier the intervention. While this should not come as a surprise, in many cases senior management does not normally intervene early in projects, leaving project teams to resolve early wrinkles. This is understandable in the context of empowered teams in which team members take ownership of delivery success, but may deny senior management a pivotal low-pain, low-cost tool to preserve delivery velocity.

Consider a project that was launched with a commitment to deliver 10 features in 10 months, or a business-driven demand velocity of one feature per month. However, the actual delivery velocity is measured as 0.8 features/month. After 10 months (original time-to-market), the company will be two...
features too short for the market. Figure 7-5 depicts the required recovery velocity and recovery timeframe, depending on the point in time that recovery intervention occurs to detect velocity loss and make delivery adjustments. When intervention occurs at t=10, the original timeframe to release to market, then a 100% increase in delivery velocity is necessary to deliver the remaining two features in an additional 2 months (point X). If the company is resource-constrained and can only increase its R&D capacity by 30% (point Y), then the recovery timeframe will have to be 6 months after the original time-to-market.

For each project, executive leadership must know the critical points in time beyond which any loss in delivery velocity could jeopardize the company’s business objectives beyond the company’s ability for corrective intervention. Collaborative executive leadership involvement at these critical points with delivery teams is pivotal.
Product development managers are adept at masking bad progress, especially in a corporate culture of retributive senior management. Executive leadership must therefore effect key cultural changes that include:

1. **Executive Leadership Mandatory Injection of Additional Capacity**
   For many product development organizations, a project team requesting additional resources reflects negatively on the project team due to the presumed suggestion of poor planning by the project leadership. This has the effect of discouraging project leadership from asking for additional capacity unless the project is really in trouble. Project leadership’s initial response is to squeeze more capacity from existing team members. By the time project leadership resorts to requesting additional capacity, project delays may be inevitable, with senior management unprepared and unable to leverage enterprise capacity to on-track the project.

   Getting additional resources usually constitutes a painful process, with project leadership driven by behavior not aligned with business objectives and a disruptive on-boarding phase for project team members. Executive leadership should regard the injection of additional capacity to in-flight projects to be strategically critical for ensuring on-time delivery to market. Executive leadership can eliminate the stigma associated with requesting additional capacity by insisting that every project team regularly submit a case for additional resources.

2. **Executive Leadership Oversight of Delivery Velocity**
   On-time, on-quality delivery is one of the most important high-impact challenges for process automation suppliers. Executive leadership customarily steps up its project involvement near the end, or when the project is in serious trouble. In the former case, executive leadership’s involvement does not contribute to delivery excellence. In the latter case, the involvement is usually adversarial, if not retributive. In both cases, there is little strategic leveraging of executive leadership to preserve delivery velocity.

   Executive leadership can be involved in specific ways without micromanaging through patronizing two or three principal workflows. Not only will this bring greater enterprise visibility to the value-add of a delivery stream, but it will also enable executives to become better evangelists for the offerings under development.
Executive leadership is responsible for setting demand velocity and for ensuring that enterprise activities are aligned with, and support, the required delivery velocity. Executive leadership has the complete picture of the business objectives, and it should review the delivery velocity at calendar points derived from time-to-market rather than on project milestones. Executive leadership must ensure that business commitments will not be jeopardized by providing additional resources early enough in a project. Delivery velocity review must be based on business-oriented indicators rather than on project-oriented metrics.

3. **Senior Management De-Bottlenecking Shared Resources**

Some key activities, such as system integration testing and system validation testing, are resource-intensive, and use resources shared across delivery streams and across projects. These critical activities create inter-project dependencies, creating business risk not based on any market objectives. The daily use of shared resources is determined by the competing project teams, but senior management has ultimate control. As adjustments are made to respond to velocity loss, schedule competition for these shared resources may become disruptive. Senior management must continually work to eliminate bottlenecks resulting from shared resources, typically by redirecting those resources or by empowering project teams. Resource redirection is a zero-sum approach whereby higher priority projects get the bottleneck resources at the expense of lower priority projects, with senior management ultimately responsible for the project time-to-market. Project team empowering is an investment approach whereby senior management ensures that all delivery bottlenecks remain under the control of the project team and supports the necessary funds to access alternative resources. For example, emerging technologies, such as cloud computing, should be used to eliminate the bottleneck of the limited availability of test equipment needed to validate complex software.

Any strategy to increase delivery velocity must therefore look at the two demand-driven constraints: the number of features and capabilities, and time-to-market. Figure 7-6 summarizes how those constraints drive delivery requirements. There are four demand scenarios (with check marks) that require increasing delivery velocity and one additional scenario (S1) that could successfully increase delivery velocity if combined with a strategy for co-creation. Most product development organizations focus on reducing time-
to-market, rather than on achieving alignment with time-to-market. To have an impact on delivery, velocity requires changes beyond R&D: key enterprise processes such as product compliance and certification, key players such as customer training, and key organizations such as manufacturing and marketing. If executive leadership sets a demand velocity that requires reduced time-to-market, then senior management must ensure that not only is R&D able to perform at a higher speed, but also that the enterprise has the capacity to support the higher speed and has the delivery engine required for a higher throughput of features and capabilities.

Some processes involving regulatory authorities, for example, are usually difficult to speed up in any deterministic way. If the time-limiting processes cannot be changed in any meaningful way, then total elapsed time cannot be

<table>
<thead>
<tr>
<th>Same Time-to-Market</th>
<th>Fewer #Features</th>
<th>More #Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same #Features</td>
<td>Same delivery velocity</td>
<td>Slower delivery velocity</td>
</tr>
<tr>
<td></td>
<td>Same time-to-market</td>
<td>Same time-to-market</td>
</tr>
<tr>
<td></td>
<td>Expected value-add</td>
<td>Reduced value-add</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reduced Time-to-Market</th>
<th>Fewer #Features</th>
<th>More #Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same #Features</td>
<td>Same delivery velocity</td>
<td>Same delivery velocity</td>
</tr>
<tr>
<td></td>
<td>Same time-to-market</td>
<td>Same time-to-market</td>
</tr>
<tr>
<td></td>
<td>Expected value-add</td>
<td>Reduced value-add</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Increased Time-to-Market</th>
<th>Fewer #Features</th>
<th>More #Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same #Features</td>
<td>Slower delivery velocity</td>
<td>Slower delivery velocity</td>
</tr>
<tr>
<td></td>
<td>Same time-to-market</td>
<td>Same time-to-market</td>
</tr>
<tr>
<td></td>
<td>Reduced value-add</td>
<td>Reduced value-add</td>
</tr>
</tbody>
</table>

| Figure 7-6. Delivery Velocity Options |

- Same #Features: Same delivery velocity, Same time-to-market, Expected value-add
- Fewer #Features: Slower delivery velocity, Same time-to-market, Reduced value-add
- More #Features: Faster delivery velocity, Same time-to-market, Higher value-add

- Same Time-to-Market: Speed and value add remain constant
- Reduced Time-to-Market: Faster delivery velocity, Better time-to-market, Expected value-add
- Increased Time-to-Market: Slower delivery velocity, Worse time-to-market, Expected value-add
reduced effectively and higher delivery velocity can only be achieved by increasing the number of features or capabilities to be delivered.

While it may be obvious that delivery velocity can be increased by shortening elapsed time or by increasing the number of features delivered, product development organizations still struggle to achieve sustainable higher velocity. Figure 7-7 offers a guideline on the best delivery velocity improvement strategy depending on the company’s business objective or targeted business performance improvements. A company may be targeting more than one of these objectives or improvements. The product development organization may therefore have to execute multiple strategies related to improving delivery velocity. These organizations should take extra care to make sure there are no conflicting strategies. For example, a product development strategy to replace obsolete components has a delivery time conflict (must be done quickly) with a strategy for features for new workflows, which require more time. A single project attempting to achieve both will therefore have in-built tension due to the different timeframe requirements. These two product development strategies should not be combined in same project.
The chosen delivery velocity strategy should be driven by the business value targeted by the company. Figure 7-7 depicts how the major delivery performance strategies are aligned to four revenue-increasing technology strategies. That alignment is achieved through the selection and execution of the most effective product development strategy. The eight product development strategies shown in Figure 7-7 are described below.

**Design for Scalable Production**
The primary objective is to scale up production while reducing unit cost. No new features or capabilities are added, but the existing design is improved to get greater efficiencies in production, logistics, and aftermarket support. These improvements should have a recurring positive impact on operating cost and cash flow, and so they must be targeted for completion as quickly as possible. Changes are cross functional, affecting R&D, manufacturing, quality, and procurement. In order to make sure there is adequate coordination and stakeholder involvement, Lean should be used.

**Design Refresh for Obsolescence**
The principal objective is to maintain production as components become obsolete, or as a need arises to change a supplier or the specifications of components. No new features or capabilities are added. There is no design change, but Quality Assurance must be involved to ensure product, system, and regulatory compliance. Procurement must also get involved to confirm component sourcing. These changes are not driven by demand, but are usually forced on the automation supplier, affecting the supply chain. The absence of a direct positive economic net benefit should bias the project toward quick completion in order to contain and minimize impact beyond the module under consideration.

Furthermore, to avoid supply chain disruptions and cost hikes, there should be a strategy and processes to manage obsolescence, based on proactive cross-functional collaboration. Close coordination with Sourcing (for supplier agreements) and Procurement (for last-time buyouts) is usually necessary, as these functions have lead times outside the control of product development.

**New Features for Current Customer Business Processes**
The recommended delivery velocity strategy is *fewer features in less time*. The objective is to enhance customers’ performance or productivity for current business processes, through either incremental or enhanced workflows. Users
should perceive a better experience in executing their business tasks, and incremental business value. For executive leadership, the new features show commitment to continually improving customers’ business processes. Delivering these new features to market in a short period of time strengthens the company’s image as being responsive to customers’ dynamic business needs, and fends off competition.

**Features for New Customer Business Processes**
The recommended delivery velocity strategy is *more features in less time*, especially for early adopter customers. The business objective is to take to market a set of features and capabilities that enable new customer workflows for business processes that are either new or not previously supported by the automation supplier’s offerings. The new workflows are implemented to increase the customers’ competitive advantage. Automation suppliers must therefore balance between aggressive time-to-market targets to beat out competitors and a customer-readiness timeframe that allows for customers’ change management processes.

For executive leadership, features and capabilities that support new business processes represent the willingness to move the market in a new direction and to grow the technology value chain. To be able to influence eventual success, executive leadership needs to get market feedback quickly, from the right set of customers, and mobilize the company to respond quickly. Executive leadership should therefore push to have the new features and capabilities delivered quickly to early adopters (Christensen 2003), while allowing a longer timeframe for the majority of customers.

**Localization of Features**
The business objective is to be able to sell offerings more effectively in new geographical markets and to be more competitive in key growth markets. No new features or capabilities are expected, except to the extent required to support local specifications and norms. However, executive leadership should expect all existing features and capabilities to be made available in localized form within a short timeframe in order to fend off competitor counter-actions. Moreover, the supply chain must be re-optimized to minimize the cost of supplying and supporting the new geographical markets at the expected service levels.
Compliance for New Regulations
The likely delivery velocity strategy is more features in less time. The business objective is to eliminate liability and potential class action lawsuits against the automation supplier. Legal implications related to alleged or actual product safety can be crippling to a company, since the punitive cost is based on society’s perspective and bears no relation to costs related to the product itself. For regulations on product safety, all existing features and capabilities must be analyzed, and sufficient time must be allowed for recertification. Executive leadership must ensure that any changes to regulations are implemented in the product offering during the grace period usually permitted by the new regulation. Failure to adhere to the timeframe could leave the company legally exposed or render its products unsellable in the market.

New Features for Higher Performance
The recommended delivery velocity strategy is fewer (selected) features in less time. The business objective is to sell offerings that have performance characteristics and improvements that customers expect. A focus on improving performance shows a commitment to not only improve customers’ workflows but also to preserve the offerings’ value proposition. Improving performance in a short timeframe provides an important competitive advantage. Senior management should identify a selected set of features, substantial enough to have a tangible business process or business performance impact for customers, but limited in number to provide the opportunity to deliver to market in the shortest possible timeframe.

Features for Customer Conversion
The recommended delivery velocity strategy is fewer (selected) features in less time. The business objective is to sell offerings into a competitor’s customer base by lowering the business-related switching cost for those customers. The assumption is that the current offerings already have a compelling value proposition for competitor’s customer base, so only a limited set of features needs to be provided to enable migration away from the competition. Since the strategy is to switch customers from the competition, and the normal tactic is to execute the conversion before the legacy competitor can respond, executive leadership must target the delivery of these custom features in a short timeframe. Increasing delivery velocity can be achieved by:

• Speeding up how components are developed and tested, to increase R&D speed
• Speeding up how quickly completed components can be combined to create complete workflow capabilities that increase integration speed

• System design strategy to optimize the system architecture in order to minimize the integration cost and time, and to facilitate validation and release readiness

This is summarized in Equation 7-3 of Figure 7-8, modeling the time it takes to develop and integrate features. The relationship between the number of features and the cycle time is nonlinear (quadratic). Companies normally know how many features to target, and need to estimate the time-to-complete. Equation 7-4 provides a useful form to estimate time-to-complete for a given number of features to target, and the R&D speed.

Good system decomposition practices mean the effort to integrate any two features is much less than the effort to design and develop a single feature. Although not all binary combinations of features are functionally meaningful, process automation systems must include rigorous consideration of failure modes to make sure there are no undesirable interactions between features and components that could make the system dangerous, unsafe, unprotected, or unable to protect or safeguard. For process automation systems, the “time to integrate” factor (α) (see Figure 7-8) can therefore not be ignored. For large systems, it quickly becomes the dominant factor in determining the delivery velocity.

Executive leadership defines the business objectives, including the business-driven value-adding features and capabilities to be delivered within a market-driven timeframe, setting up the business-driven demand velocity. The product development organization must then make sure its R&D speed is high enough to power a delivery velocity that meets the demand velocity. The quadratic relationship between time-to-complete and number of features, modeled in Equation 7-4, shows the more features developed, the slower the effective delivery velocity. In fact, a demand velocity (V₀) will dictate the maximum possible number of features (F) and time-to-market (t), for a given R&D speed (1/p). Senior management must recognize that project teams have limited degrees of freedom to meet both the demand velocity and the number of features, due to R&D speed and system integration efficiency. R&D speed must be greater than the demand velocity adjusted for integration, as shown in Equation 7-6. The integration adjustment factor is a function of the number of features, capabilities, and components to be integrated. R&D managers can
Innovation Agility

The case for agility is rooted in both the business demands bearing on process automation suppliers and the nature of software development. According to Kelly (2008), “If we look at the definition of knowledge workers, it is clear that it includes (software) developers” where “development activities—specifying, designing, coding, and testing new software—are themselves knowledge activities.” A knowledge worker’s individual knowledge can make an immediate difference to the end product. Today’s business environment is constantly changing as a result of both new competitors and changing customer expectations. New knowledge acquired in technology can ideally be used to create new opportunities, thereby providing a “feed-forward” effect to the changing business environment.

An enterprise consists of four systems: human, economic, natural, and knowledge. Kelly (2008) states that Agile is based on the view of software development as consisting of people, internal politics, and emergent behavior. This view and that of software developers as knowledge workers suggest that Agile methods focus primarily on the human system and the knowledge system. The scope of agility is extended to include the economic system (as a

use Equation 7-6 to determine the minimum target R&D speed, and to discuss with executive leadership the number of features for committed delivery.

Figure 7-8. R&D Speed

\[ V_0 = \frac{F}{t} \]

\[ F/t > V_0 \Rightarrow F/t < V_0 \]
source of business drivers) and the natural system (physics, materials science, mechanics, and the packaging of components).

Ulf Henriksson, former CEO of Invensys plc, defined agility as an enterprise-wide core value necessary to “adapt, grow and swiftly change for a sustainable future.” According to Leffingwell (2007), the aim for Agile methods is to create “reliable software more quickly and doing so while eliminating unnecessary waste and unproductive overhead.” Lean and Agile form a “virtuous cycle”: Lean eliminates waste, increasing the opportunity to “identify and deliver value” (enterprise-wide). This captured value will enable additional resources and the motivation to eliminate more waste. The additional resources can then be deployed to create new value using Agile methods. Lean can then be leveraged to capture and maximize the created value.

At the enterprise level, Murmann et al. (2002) and Kelly (2008) suggest that agility can be viewed in the context of a lean enterprise, as depicted in Figure 7-9.

While agility can be considered at different levels of an enterprise, Agile methods can be considered at the narrower program level. At this level, Agile methods can be contrasted with plan-based methods.

There are many Agile methods, each with particular emphasis and strength. Some common Agile methods, shown in Figure 7-9, are summarized in Table 7-2.
A process automation project, with its typical technical complexity and range of stakeholders, will typically require a combination of Agile methods, such as:

- XP (for software development)
- Scrum (for project management)
- FDD (for modeling incremental innovation + build management)
- RUP (for architectural innovation)

**Organization for Agility**

Based on Murmann et al. (2002) and Kelly (2008), the prerequisites for agility are:

- Organization is lean and/or is practicing Lean principles
- Individuals are empowered to act on what they have learned
- There is trust and honesty among individuals
- Organization must know how to tolerate failure

According to Kelly (2008), learning and change happen at many levels: individual, team, and organizational. An important level of learning not included in Kelly is the systems level, where systems thinking principles focus on:

- Emergent relationships among components
- Emergent patterns
- Emergent interactions and inter-operations
In addition, Leveson (1995) provides perspective into the corollary concept of system complexity. Leveson suggests that the challenges of building complex systems may be rooted in the limited ability of humans to manage intellectual complexity. Given that the basic human ability is not changing, it is necessary to augment human ability “both in terms of system designers and system users.” Leveson recommends turning to cognitive psychology and social sciences for ideas to augment engineering foundations.

Kelly (2008) suggests that traditional development organizations use predefined processes as a defensive mechanism whereby failures can be attributed to flaws in the process. For process automation suppliers, stakeholders usually expect an individual in the supplier organization to be accountable for the process, so failures are not casually attributed to the process. Kelly also discourages having a common pool of developers because such an arrangement undermines a team vision. However, it is also possible that a strong culture centered on predefined processes obviates the need for a strong common vision and facilitates the establishment of a common pool of developers. Agile practices discourage the use of a common pool of developers because such an arrangement undermines a team vision to bind and focus the team-members. However, it is also possible that a strong team vision is necessary in the absence of a common pool of developers guided by a predefined process. For process automation technology, the following organizational practices should be included:

- Transitioning to Agile to ensure the non-disruptive coexistence of legacy processes and Agile practices
- Investing in Agile capability, a corporate-sponsored strategic investment in capability (personnel, infrastructure, and tools) to enable and sustain a higher level of operational performance

**Agile Principles for Process Automation**

The Agile Manifesto (2001) is specific about what is more valuable during product development:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan
Agile methods are founded on 5 principles of Lean, 4 Agile values, and 12 Agile principles, all fixed. Of the Manifesto’s 12 principles, 7 have to be re-assessed for process automation offerings, as follows:

1. “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”
   For process automation customers, continuous delivery of modified automation offerings is likely to be burdensome because of the cost of deploying changed software and of potential disruption to manufacturing operations. However, this challenge can also present an innovation opportunity to (and a potential competitive edge for) an automation supplier. Focus should be on customer needs, rather than on quality improvement or (internal) productivity enhancement.

2. “Deliver working software frequently, from a couple of weeks to a couple of months, with preference to the shorter timescale.”
   This principle, like the previous one, has limited benefits for automation offerings. Both the rigor for system change readiness and the potential disruption to production prevent many manufacturers from adopting frequently changed software. Any benefits for an automation vendor will be primarily internal to development operations. For process automation systems, regression testing, integration testing, and system testing often take months. However, this principle may spur innovation to complete regression testing in weeks.

3. “Business people and developers must work together daily throughout the project.”
   This principle is especially challenging in the case of automation offerings because of the gap in the backgrounds between developers skilled in fast-changing technology stacks and manufacturing business professionals steeped in well-established principals of industrial production, supply chain management, energy management and occupational hazards. Experts say that development personnel from automation suppliers are generally insular, with little knowledge about customers, and narrow and decreasing knowledge about manufacturing. While this principle will help redress the empathy gap, developers may struggle to work directly with manufacturer representatives. Automation suppliers may instead rely on product managers and program managers as intermediaries.
4. “The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.”

The system size and complexity of automation offerings, as well as the multidiscipline composition of automation development teams, undermine reliance on face-to-face conversation for “most effective” communication. While this principle reinforces Allen’s Law, data does not support face-to-face to be the only efficient and effective way. In fact, results in Chapter 6 show engineering standards to be effective in pinpointing errors (essential for high quality).

5. “Working software is the primary measure of progress.”

This principle is inadequate for process automation offerings. High-quality working software can be accepted as a measure of success together with high-quality hardware, due to the high dependency between hardware and software for automation systems. Requirements for automation systems to ensure and enforce the safety of humans and the environment mean the software must be the highest quality possible; low quality working software is so disruptive as to be unacceptable and therefore a poor measure of progress.

6. “Simplicity—the art of maximizing the amount of work not done—is essential.”

This principle needs re-adaptation for automation offerings in order to focus on a good architecture rather than the amount of work. For process automation, offerings are typically components (modules or subsystems) of a complex system. Simplicity in the architecture might require considerable effort and collaboration. This principle should therefore be practiced as advised by Crawley (2007) in accordance with the key system architecting principles of “reducing complexity, resolving ambiguity, and focusing creativity.”

7. “The best architectures, requirements, and designs emerge from self-organizing teams.”

Applying this principle to automation requires careful consideration of the role of automation systems in the manufacturing value chain. The capacity, capability, and track record of a self-organizing team must be calibrated to confirm its competency and empowerment in producing requirements. The number of stakeholders, diversity in customer types, and the chain of users for an automation system makes the gathering of requirements particularly challenging. Requirements are often articulated to provide contractual form to an (unarticulated)
intent. It is not clear how willing customers will be to contractually accept anything less formal than traditional requirements. During architectural design, access to intent may provide a better design. Access to experts in stakeholders’ needs is essential for the best architecture.

While 7 of the 12 Agile principles need rigorous reevaluation for process automation, they may also provide opportunities for innovation. These principles have already demonstrated value outside process automation, and many practices associated with the principles are well documented.

While the Manifesto has fixed Agile values and Agile principles, different literature emphasizes different practices. Table 7-3 summarizes practices from three different books. Practices that are common between books are noted with a linking identifier.

**Agile Practices for Process Automation**

Kelly (2008) asserts that Knowledge = Learning + Action. Moreover, businesses today obtain a competitive advantage through access to knowledge and by acting on that knowledge. The specific objective of learning is to improve the products, the understanding of customer needs, and the processes in use. Focusing on quality improvement, productivity enhancement, and cost cutting is unhelpful, although making developers more productive is an important goal for Agile methods. Kelly (2008) identifies nine Agile practices, listed in Table 7-3. These practices, and all other Agile practices, are required to be consistent with one or more of the 12 Agile principles without violating any of those principles.

Kelly (2008) also recommends the following *attitudes* to accompany Agile practices:

- Accept that requirements will change.
- Organize software development activities as projects.
- Focus on incremental innovation.
This activity is explicitly recommended by Leffingwell (2007), although not listed there as a practice.

When Kelly’s (2008) practices (shown in Table 7-3) and the three recommendations above are further evaluated with respect to the key manufacturing process stakeholders (see Chapter 4), the following gaps become apparent:

- **Test Driven Development**
  This approach works well for positive testing, for example, to show that a function produces the correct output when given a known input. However, it is inadequate for negative testing, for example, when testing how robust software in the presence of out-of-spec conditions.

  Test-driven development is inadequate for negative testing since that must include test cases for behavior that emerges from the implemented system, independent of requirements.

<table>
<thead>
<tr>
<th>Table 7-3. Agile Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enterprise Agility</strong></td>
</tr>
<tr>
<td>- Define/build/test component team</td>
</tr>
<tr>
<td>- Two-level planning and tracking</td>
</tr>
<tr>
<td>- Mastering the iteration</td>
</tr>
<tr>
<td>- Smaller, more frequent releases</td>
</tr>
<tr>
<td>- Concurrent testing</td>
</tr>
<tr>
<td>- Continuous integration</td>
</tr>
<tr>
<td>- Regular reflection and adaptation</td>
</tr>
<tr>
<td><strong>Complex Long-Lived Systems</strong></td>
</tr>
<tr>
<td>- Intentional architecture</td>
</tr>
<tr>
<td>- Vision + roadmap + JIT elaboration</td>
</tr>
<tr>
<td>- Agile release train</td>
</tr>
<tr>
<td>- Impact on customers and operations</td>
</tr>
<tr>
<td>- Measuring business performance</td>
</tr>
</tbody>
</table>

\( (L^*) \) This activity is explicitly recommended by Leffingwell (2007), although not listed there as a practice.
• **Pair Programming**
Although this approach works well for individual software modules, it is unclear how it would work for a system. Moreover, Leveson (1996) cites interfaces and software complexity as being the primary cause of software failures:

> “The problems in building complex systems today often arise in the interfaces between the components... where the components may be hardware, software, or human.”

Agile methods do not include systems thinking considerations. In particular, the practice of pair programming does not address the systems level needs.

• **Quality before Features**
Given the stakeholders identified in Chapter 4, the various definitions of quality should be enumerated. Although considering quality before implementing features is an important practice, considering quality after features have been implemented is also critical for process automation systems. Furthermore, large systems, such as automation systems, have emergent properties that can have a direct impact on the quality and perceived cost of ownership of a system. An example emergent property is system complexity, which is not considered in the Agile practices covered by Leffingwell (2007), Kelly (2008), or Elssamadisy (2009). The following are recommended for adoption as Agile practices for coping with large-scale system complexity:

  – *Managing critical system parameters* to ensure that architectures and designs preserve essential (emergent) system properties expected by stakeholders (PDSS 2006).

  – *Engineering system robustness* to confirm that the implemented (constructed) architecture fulfills “intent associated with the externally delivered processes which are traceable to customer needs” (Crawley 2007, Frey 2007).

• **Malleable Design**
The emphasis on a malleable design and the avoidance of reusable code are together a promising solution to the problem of reusable code. Leveson (1996) argues that the practice of code reuse has been a dominant source of catastrophic accidents. This is particularly true of software that controls machinery or energy sources, and it is
fundamentally rooted in how software can cause interactions among physical components that would be impossible in the natural world. Since software is developed, tested, and qualified in a defined context, it is risky to generalize that context to all other situations. However, emphasizing malleable design over reusable code will end the misplaced expectations of reusable software.

Valerdi et al. (2009) provide a comprehensive treatment of the value of system engineering reuse, providing some important observations that should guide this recommended practice:

- Products, processes, and knowledge are all reusable artifacts.
- Reuse is knowledge that must be deliberately captured to be beneficial.
- The benefits of reuse do not scale linearly with the size of the system under consideration. The larger the system, the more likely new system modes will emerge, thereby reducing the reusable system engineering footprint.

### Refactoring

Because of the high switching costs for manufacturers, a process automation vendor usually has a sizeable installed base. The software in that installed base is usually controlling dangerous machinery or elements, often after approval by representatives of the manufacturing process stakeholders identified in Chapter 4. Concerns over applying refactored software to an application controlling a running plant can be compared to concerns over reusing software, covered by Leveson (2004). (Refactoring changes platform software and assumes no change to the application, while reuse changes the application and assumes no change to platform software). Releasing refactored software to be used to control a manufacturing plant will create apprehension among manufacturers and stakeholders.

### Release Often

This practice was identified as undesirable by at least one of the operations management experts interviewed for this research. Most process control systems are not connected to the network, and any change to a deployed automation system requires human supervision. Producing too many releases will be burdensome, and potentially costly, to manufacturers. It may also increase the risk of system upsets.
Elssamadisy (2009) also recognizes release often as being a problem for enterprise customers. However, no fundamental technical reasons are offered to disqualify this practice for the automation industry. The capability for painless multiple migration paths for customers may eliminate the burden to customers.

• **Accept that Requirements Will Change**

Requirements could change for many reasons. Kelly (2008) seems to assume that changes in the business environment are the root cause of a requirements change, but changes may also be due to poorly understood needs or to regulatory changes. It is not clear whether Agile methods should have different practices for the different root causes for requirements changes. This is echoed strongly in a different form in LaFon (2008) who states, “Sources of change are evolving technologies, determinations and discoveries made during feasibility analyses, discovery of missing or poorly designed system specifications, or discovery of incorrectly specified components and technologies.”

Expecting requirements to change is at the heart of agility, but changes in requirements introduce considerable uncertainty to a project, especially since there could be many different root causes for a change in a requirement.

• **Organize Software Development Activities as Projects**

According to Pressmann (2001), projects are generally required to have a schedule and plans for resources. A project is therefore not appropriate for activities with considerable technological and organizational uncertainty and that cannot, therefore, be scheduled effectively. However, these types of projects can be classified as research projects, and a separate set of Agile recommendations should be provided for research projects. Pressman, for example, accounts for prototyping in traditional development processes as an approach to remove technical and organizational uncertainty.

Project management best practices assert that a highly effective strategy that includes a mission statement, top management support, and planning is important for a successful project, while Agile methods encourage only a good team vision.
• **Focus on Incremental Innovation**
  As established in Chapter 3, manufacturers are conservative and cautious, and therefore prefer incremental innovation. Agile methods are most suitable for incremental innovation. However, the complex value chain established in Chapter 4 and manufacturer complaints about gaps in offerings may require breakthrough innovation, and perhaps even innovation disruptive to manufacturer processes.

Katz (2004) characterizes projects with a high potential for implementation success as having both a highly effective strategy (planning phase) and highly effective tactics (implementation and execution). Projects with a low-effectiveness strategy but highly effective tactics are more likely to have “high acceptance misuse.” Work will be done when there is no solid justification for it, or the wrong problem will be solved. Agile methods address the need for highly effective strategy partially through emphasis on a team vision, while highly effective tactics are addressed by the *leaning + action* cycle. While it is the actual practices that will determine the extent to which an Agile implementation meets the Katz standard for high potential, the principles of Agile *expect a high standard*.

**Scalable Agility**

**Agile Practice for Large Organizations**

Leffingwell (2007), while noting that “XP is not sufficient for managing requirements of large-scale systems” goes on to emphasize the following benefits of Agile development that are also sought after for large systems, including automation systems:

- More likely to be first to market
- Delivering solutions that directly address customers’ pain points
- Delivering solutions that have the requisite quality and functionality
- Adapting more rapidly to business and technology changes

Leffingwell (2007) recommends 14 practices for a large-scale system, identified in Table 7-3, intended to address the challenges of organizing for agility in large organizations. Some of the practices, however, require further evalua-
tion given the characteristics of the process automation sector and the general requirements of a process automation offering.

**Intentional Architecture**
The defined practice mentions the importance of working code, but there is insufficient emphasis on the need to confirm early on that the architecture meets the intent.

**Lean Requirements: Vision, Roadmap, Just-in-Time Elaboration**
This practice is intended to ensure that performance, reliability, and scalability needs are understood by all teams. Moreover, a vision is intended to be a container for nonfunctional requirements. However, it is not clear if the style and brevity of a good vision statement will be sufficient to satisfy automation stakeholders who have real concern and justified paranoia about possible loss of life as a result of system failures.

While the roadmap provides a good tool to prioritize needs, it is not clear who defines the roadmap and how the assumptions underlying the prioritization decisions are captured and checked. The observations in Chapter 5 suggest that automation suppliers often make wrong guesses about what manufacturers actually need.

**Managing Highly Distributed Teams**
Applying Agile methods may in fact make distributed development more challenging. In plan-driven development, people-to-people communication protocols can be defined to mitigate the effects of distance. Multi-location, multi-time-zone development may make daily meetings intrusive, while relying on ad hoc communications routinely extends response time to a problem to more than 24 hours. Multi-location, multi-time-zone development may make daily stand-up meetings intrusive. It may be necessary to plan for a communication mechanism that would ensure that response time to queries is not stretched to one or more days.

**Changing the Organization**
This practice serves to place focus on continual reorganization to identify roadblocks, bottlenecks, and cultural impediments. It does not, however, address the reorganization necessary to transition from classical plan-based development to Agile. The challenges of coexistence between the two approaches to development must be carefully managed to ensure that the
quality of offerings is not affected. It should be noted that the practice does not address whether more personnel and resources will be required to implement Agile. The need to fit system testing and regression testing into an 8-week timeframe will likely require additional investment.

Measuring Business Performance
The defined practice provides helpful metrics about an ongoing project. There is, however, no connection to the Agile framework in place and the Agile practices chosen.

Leffingwell (2007) recommends the release management train (RMT) as the mechanism to use to coordinate releases from multiple teams. Over the duration of a project, the project team must adapt strategic focus and operational tactics for RMT to ensure:

- Each RMT has a vision with themes and end-to-end use cases.
- Each RMT release is customer-ready, even if it is just an internal release. A release will be preceded by three team iterations and one hardening iteration. Each team must therefore have a primary plan and a fallback plan.
- RMT is treated as a project artifact, subject to the expected best project management practices of configuration management and risk management.

Agile Practices for Large Systems
Elssamadisy (2009) emphasizes the following business values of Agile:

- Increase product utility (value to market)
- Increase quality to market
- Reduce cost to develop
- Increase product lifetime

Automation offerings are becoming increasingly commoditized. These offerings already have a long lifetime (20–30 years). Increase product lifetime is therefore unlikely to be a priority for process automation suppliers. New offerings enable suppliers to ask for higher prices as existing offerings face downward pressure on price.
Elssamadisy (2009) explores patterns (practices and guiding principles) for scaling up Agile methods to large development organizations such as a major process automation supplier. These patterns are founded on focusing development on delivering business value and recognizing project problems. Three major types of patterns are proposed:

- **For rapid feedback**: iteration, kickoff, stand-up, demonstrations, and retrospectives

- **For fostering team development**: colocation, self-organization, cross-functional roles, and customer-part-of-team

- **For facilitating technical tasks**: testing, refactoring, continuous integration, simple design, collective code ownership, and pair programming

Elssamadisy (2009) makes specific recommendations for managing Agile projects. The recommendations are based on the following premises:

“Software development professionals are highly trained knowledge workers. They are more qualified to make decisions regarding how best to do their jobs than (are) their managers.”

“It is demoralizing and frustrating to professionals when they can see a solution to their problems but official processes and procedures impede that solution.”

Elssamadisy recommends that management communicate goals rather than mandate tasks. Documentation should be kept to the very minimum, and stand-up meetings should be the principal forum for discussing and resolving problems. Moreover, the purpose of any document should be to evoke discussion.

While Agile values and principles are fixed, Agile practices are not. Different Agile thought-leaders offer different practices. Some practices are common; some are conflicting. For example, while Leffingwell (2007) recommends a hardening iteration to remove bugs from software, stabilize the software, and ensure readiness for internal release or customer assessment, Elssamadisy (2009) views hardening as a bad and inefficient practice.
People Aspects of Agility

It is possible to infer from the literature that there are different types of agility: personal, team, organizational, enterprise, and market. The researched literature covered team, organizational, and enterprise agility. There is little mention of personal agility: how individuals make themselves more agile. (Personal agility also applies to personnel in stakeholder organizations.)

Leffingwell (2007) asserts, “New teams and new personnel in these product-driving roles are a primary factor in the success of the Agile enterprise.”

While Elssamadisy (2009) states that “deliver(ing) value to the customer is the main driver for all Agile development practices” and adds that it is therefore important to “know what business value is delivered by the software development practices we use.” Personnel differ in what they consider to be the most important value of their work, based on role. While this diversity of work value is probably beneficial in general, entrenched differences in what is most important (e.g., defect count versus product performance) are likely to be a challenge for a cross-functional Agile team that must be guided by a single vision and governed by 6–8 week iterations. Organizations adopting Agile methods should also pay attention to personal agility: how individuals can train themselves to be successful practitioners of Agile methods. It is important to all individuals in a team to commit to a common customer (or business) value, given entrenched differences among personnel types about what is most valuable in a product.

Elssamadisy (2009) advises, “Individual responsibility is the bedrock of personal agility.” Individuals must know they are responsible for recognizing and responding to the need for change. In established development organizations, it is not unusual for an engineer to detect a problem but to take no initiative for corrective action because he may feel unauthorized, inadequate, or intrusive in pointing out the problem.

Agile development emphasizes communication in every aspect of the software development process. More person-to-person communication is expected for Agile projects. For distributed teams, this means more traveling for team members (not just managers) and ready access to the full complement of long-distance communication media. Established Agile methods however, do not cover practices for the following market leadership factors:
• **Ensuring domain expertise** to ensure that all assumptions and expectations about domain knowledge are understood, assessed, and challenged, and to access the latest in state-of-the-practice.

• **Clarifying the company’s role in the ecosystem** to ensure that the team’s vision and other selected practices reinforce and support either good platform leadership or good platform contribution.

**Risk Balancing for Agility**

There are two dominant means for assessing software development methods: waterfall versus iterative and plan-driven versus Agile. Considerations for waterfall versus iterative are well documented, for example, Pressman (2001). The key considerations for choosing between plan-driven and Agile are covered by Boehm (2004). The landscape of software methods is depicted in Figure 7-10, showing the position of a sample of popular software development methods. While it may be adequate to consider either plan-driven versus Agile or waterfall versus iterative for software-only application development, both considerations are necessary when software development includes complex platforms and hardware.

![Figure 7-10. Landscape of Software Methods](image)

Boehm (2004) identifies five risk factors that must be independently assessed for each project. The “five critical factors involved in determining the relative suitability of Agile or plan-driven methods in a particular project situation” are project size, criticality, dynamism, personnel, and culture. Established
Agile methods, however, do not cover risks introduced by project complexity. The following practices, if adopted and adapted to be Agile practices, could be effective for managing project complexity:

- **Continuous system validation** to ensure that the required project regression testing and system testing can all be completed within the 6–8 week time frame for an iteration.

- **Balancing risk** to select the appropriate balance of agility versus plan-driven methods based on the risks due to project size, criticality, dynamism, personnel, and culture.

Each delivery stream must assess its combined risks to determine the appropriate balance, based on the risks of the chosen development approach and on the technology, demand, and supply chain. Using the dimensions of software methods in Figure 7-10, there are four types of development approaches: (1) iterative and plan-driven, (2) waterfall and plan-driven, (3) waterfall and Agile, and (4) iterative and Agile.

**Iterative and Plan-Driven**

This approach is suitable for projects in disruptive technologies for breakthrough applications. An overarching plan provides a framework for establishing the business case, product requirements, and resource needs, as well as for managing financial and market commitments. Iterations allow technology and market uncertainties to be managed prior to the innovation’s release for general availability in the market. An example of iterative and plan-driven development is the spiral model for software (Pressman 2001), where the objectives of each spiral can be scoped out, and the deliverables of each phase of a spiral can be planned. Table 7-4 summarizes the risks that can arise for innovation that is iterative and plan-driven.
<table>
<thead>
<tr>
<th>Table 7-4. Risks with Iterative and Plan-Driven Innovation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Innovation Risks</strong></td>
</tr>
<tr>
<td>Inadequate robustness may emerge for products being developed that have technology components never before applied in an industrial setting. With the iterations lacking experimentation in their plan-driven methods, errors in requirement specifications risk being passed on to design and possibly being released to customers at the conclusion of iterations. There is also risk of poor customer readiness when the product has novel technology components. The customers’ facilities, environment, and infrastructure may not be upgraded in time for each iteration to ensure that the operating conditions are within the specifications for the new technology.</td>
</tr>
</tbody>
</table>

| **Demand-Side Risks**                                    |
| There is risk of reduced sales for existing offerings due to cannibalization or customers perceiving uncertainty in product portfolio strategy while iterations are still ongoing. New value proposition may be poorly understood by customers until enough iterations of the offering have been validated by a sufficient base of customers, and market segmentation has been adequately confirmed, to ensure the plan-driven functional requirements cover the complete range of requirements. Competitors may gain market share by finding entry opportunities for the segments least-served by completed iterations. Early market exit threatens offerings whose growth relies on network effects but whose plan-driven design does not adequately reflect user expectations, undermining the iterative build-up of critical mass of connected users. |

| **Supply-Side Risks**                                    |
| Production cost may increase and profitability drop due to a suboptimal supply chain for new technology components, or for production of the new product offering, until a critical number of iterations are complete. Service levels may be affected by poor order fulfilment readiness if existing distribution channels and aftermarket support functions cannot be adequately stocked with the new offering for all iterations. |
Waterfall and Plan-Driven

This approach is used in large-scope product development projects with high product or project complexity. An overarching plan provides the blueprint to manage complexity and risks, and to coordinate the large number of teams and dependencies between teams. The waterfall approach is aimed at containing runaway project costs and delays by insisting on due diligence in all preceding steps. For projects that span multiple financial periods, a detailed plan with disciplined sign-off provides a basis for interim payments based on progress and milestones. Table 7-5 summarizes some of the risks that need greater attention for innovation that is waterfall and plan-driven.

Table 7-5. Risks with Waterfall and Plan-Driven Innovation

<table>
<thead>
<tr>
<th>Innovation Risks</th>
<th>There is risk that emergent complexity will result in an offering with limited market acceptance if the number of components being combined in the offering is high with considerable interdependencies between components and functional dependency based on design. Reliance on plan-driven capture of product and system requirements means there is risk that the system design will not adequately account for variations in customer personnel operational practices.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand-Side Risks</td>
<td>For offerings based on RFQ, the value proposition is based on the business and functional requirements being properly captured early in the development life cycle, typically using plan-driven methods. The waterfall approach, with focus on internal process criteria for continuing investing in the innovation, creates risk that the offering’s ROI will be below business target.</td>
</tr>
<tr>
<td>Supply-Side Risks</td>
<td>There is risk that the new offering, based on rigid set of plan-driven product requirements, does not have features or capabilities that scale to other customer business processes. There is also risk that the waterfall approach cannot adequately deliver products for the voice of both early adopters and majority customers.</td>
</tr>
<tr>
<td></td>
<td>If the plan-driven business opportunity is used as the basis for material planning and volume purchasing, there is risk of higher than expected direct material cost due to incorrect supply-chain change assumptions or to overstocking on raw materials. Waterfall methods largely treat the exchange between R&amp;D and the supply chain as transactional (exchanged information assumed to be accurate and correct).</td>
</tr>
</tbody>
</table>
Waterfall and Agile

This approach is particularly useful in a target system release with multiple delivery streams with key sync points. This can happen during a major upgrade of a product line that has a complex set of products and offerings, or when the upgrade requires a combination of elements from different engineering disciplines (e.g., hardware, software, mechanical). Agile development can be applied to the incremental innovation of individual subsystems or delivery streams. Waterfall can be used to manage the overall scope of the release, including engaging key stakeholders to confirm and coordinate subsystems permitted to move forward in the release. Table 7-6 summarizes risks that need closer attention for innovation that is waterfall and Agile.

Table 7-6. Risks with Waterfall and Agile Innovation

<table>
<thead>
<tr>
<th>Innovation Risks</th>
<th>Different subsystems have different technology risk profiles. There may also be considerable variation in subsystem interdependencies. With different Agile methods used for different subsystems, there is risk of significant variation in the time-to-complete of different subsystems, resulting in suboptimal resource allocation for each iteration, and significant reduction in effective R&amp;D speed. There is risk that actual delivery velocity is difficult to determine, making it difficult for credible commitments to launch the overall system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand-Side Risks</td>
<td>With different Agile methods used for different subsystems, there may be significant variation in the perceived progress of development for different subsystems. There is risk that the delivered features will not be released or made available to customers in iterations sequence or on a timeline that is acceptable to critical mass of customers. This creates a sales impact risk as lead customers could delay plans to upgrade legacy systems.</td>
</tr>
<tr>
<td>Supply-Side Risks</td>
<td>In cases where the delivery includes both new technologies and existing technologies, Agile delivery of subsystems may treat supply chain changes as incremental, not requiring optimization. With waterfall methods for delivery of the overall system, there is risk that supply chain costs are not optimum, resulting in higher production cost and lower profitability.</td>
</tr>
</tbody>
</table>
Iterative and Agile

This approach is suitable for incremental innovation where reference designs exist, component interdependencies are well understood, and integration can be managed incrementally and complemented with robust regression testing. An iterative approach allows features and capabilities to be release-ready prior to time to complete, while an Agile approach allows requirements to be validated with customer representatives. Table 7-7 summarizes risks that should have greater focus for innovation that is iterative and agile.

Table 7-7. Risks with Iterative and Agile Innovation

| Innovation Risks | The features and capabilities under development are incremental to an existing offering, but Agile methods allow for additional requirements to be confirmed during development. There is risk that insufficient time will be spent to develop additional requirements, or that planned iterations will not cover enough needs for broad customer base and diverse customer business processes. The features and capabilities may have poorly understood requirements. With Agile methods, there is risk that starting development before stabilizing requirements may result in instability during development, poor convergence on product acceptance, and considerable rework. Moreover, iterative releases may not adequately distinguish between the needs of early adopters and early majority adopters. |
| Demand-Side Risks | With Agile methods, developed features and capabilities will focus on a subset of products that are of interest to the partnering customers. With iterative releases, there is risk of implemented features and capabilities not being adequately ported across the entire product line. There is risk that the target features and capabilities will require greater design complexity for market success than initially assumed or than is suitable for current Agile methods. As a result, executive intervention will be required to restart the project with a different approach, to reset market expectations, and to recalibrate the business objectives. |
| Supply-Side Risks | With iterative releases, there is risk that there is insufficient time to set up suppliers or to procure critical components in a timely fashion at optimum supply chain cost. |
Summary

*Delivery excellence* is taking products to market at the delivery velocity required by the demand velocity driven by business objectives. Executive leadership must play a more active role more often during product development to ensure that there is no loss in delivery velocity. Product development can raise the delivery velocity by increasing R&D speed. Agile methods provide an effective and proven set of practices to increase R&D speed, although additional practices are necessary for process automation suppliers for the unique opportunities and risks related to manufacturing. Since many of the key opportunities and risks for process automation systems are generally common across industrial equipment for manufacturing, the recommended delivery excellence and Agile practices are applicable to industrial innovation in general.