Agile's Role in Developing Robust Software Competency at Precor

Brent Barton  
Agile Consultant  
Agile Advantage, Inc.  
Redmond, WA USA  
brent.barton@gmail.com

Brent Brooks  
Vice President  
Precor, Inc.  
Woodinville, WA USA  
brent.brooks@precor.com

Abstract— Precor has a history of innovation as a leader in the fitness equipment industry. Market forces caused new technology innovations, and traditional outsourcing strategies failed to deliver on the new vision. The decision to develop these new product capabilities in-house created the need for software competency that didn’t currently exist.

Cost of software was historically considered “free” because it was very simple and ROI was far less than a year. Now, software is driving many business decisions and strategies. And today the software group has about 40 people.

Starting a software competency in an existing business with material non-software businesses is not straightforward. Sustaining this competency requires successfully addressing challenges with the entire organization and is a long-term effort. This experience report provides a multi-year reflection and includes business decisions that drove key behaviors.

Keywords-component; complexity, embedded software, innovation, Long-lead times, Manufacturing, Precor, Scaling, Scrum, Software Competency

I. INTRODUCTION TO PRECOR

Established in 1980, Precor sets the standard for quality, innovation, and performance in exercise equipment. Precor has distinguished itself as a worldwide industry leader, providing state-of-the-art fitness equipment to health clubs, hotels, spas, fitness centers, and private homes all over the world. By 2007 a variety of forces were combining to put pressure on Precor to do something significantly different in pursuit of long-term sustainable growth.

Within the fitness industry the rate of new club openings was beginning to slow, meaning future growth would need to come at the direct expense of competitors fighting for a greater share of the same pie. Where opportunities for growth did exist they tended to be within “vertical channels” such as hospitality and education. Indeed, fitness was beginning to play a more important role within vertical markets but individuals were often making purchase decisions in these environments that had less familiarity with fitness equipment and traditional fitness equipment brands. The overall result was a slow commoditization of an industry where opportunities to innovate by inventing a new category of fitness equipment (e.g. treadmill, elliptical, bikes, etc.) were increasingly rare and where innovation within a category of equipment (e.g. a new style of seat for a recumbent bike) were incremental at best. This is a classic case where innovation needs to disrupt an industry [1].

Meanwhile people everywhere were struggling to make and keep a regular fitness routine. Obesity rates across the globe, and in particular the United States, continued to accelerate rapidly. And within the industry, statistics showed that on average 1 out of 3 exercisers would drop out within their first year of joining a fitness club. For a company like Precor whose president was quoted as saying “we’re in the dream fulfillment business” these were concerning trends.

II. EMBRACING CHANGE

Beyond the fitness industry there were forces at play that would provide additional motivation for change. A revolution was underway that was reshaping the way people rely on technology over the course of their daily lives and companies like Apple were redefining user expectations on what great technology-driven experiences could be. The fitness industry however was falling further and further behind the technology curve, and the limited technology based products that had been introduced failed to captivate exercisers or fitness operators in a significant way.

If we could integrate new user experience paradigms (like the ones from Apple) seamlessly within a fitness context, we could change the fitness industry. Limited resources were applied to pre-development concepts. Geoffrey Moore describes this as a Horizon 3 initiative because it is not intended to compete with current product lines for some time [2]. We developed simple prototypes and started product design.

Our concept became bigger and bigger. We wanted to change the fitness industry with the new user experience paradigms. We had a new grand concept, code-named Atlas. We set the budget and plans, based on previous experience and initial estimates. Our new technology would be a material part of our business revenues within two years. This meant we needed to equip the entire organization, from development to sales to support.
III. OUTSOURCING THE SOFTWARE DEVELOPMENT

Precor has a strong tradition as an equipment manufacturer, and the reputation of Precor for product innovation and quality is arguably second to none. This legacy of world-class hardware design and development established a strong “product-centric” culture and innovative spirit. However, the strength of this heritage also created both practical and cultural challenges.

Through much of Precor’s history, manufacturing practices relied heavily on vendor relationships. We assumed we could outsource the software since we did not have enough software expertise. We chose specific software vendors based on reputation and expert recommendation. We did our due diligence.

Far more complexity exists in the Atlas related product line than in previous comparative products. We designed (and ultimately built) an integrated console that combines real-time multi-function motion controls, networked connectivity, remote upgrades, highly interactive displays with overlay, gesture-based controls, television, video and a 15” HD touch screen. All of this must work interchangeably on our newly redesigned cardio fitness product line. Attention to good design provides the right experience without degrading the fitness effort or causing safety issues.

We also needed a web-based system for managing units in the field, delivering content and providing software updates. As we made progress on the design and prototyping of the next generation of our physical products, the software lagged further and further behind. Within the core technologies for embedded software, it seemed impossible to evaluate progress. The web-based platform capabilities got so far ahead in particular areas while waiting for the console development that the value of the platform work became highly questionable.

It took six months to acknowledge that software was a huge problem and take effective action. Now, the whole initiative was behind schedule and over budget. Further, Precor lacked sufficient internal software expertise to create the technology required to meet the vision.

IV. BRINGING AGILE SOFTWARE DEVELOPMENT IN-HOUSE

Feeling a sense of loss of control, Precor decided it needed its own software expertise if it were to be successful, similar to the expertise in mechanical engineering and manufacturing. In Q1 2009, we reduced the number of software vendors to two: keeping the Platform team (within 30 miles of our HQ) and a small China-based media development team and started planning to hire key staff.

We hired software development expertise and the team started adopting agile methods. The outsourced Platform team was using Scrum/XP to support this effort. We needed a better software process and this seemed to deliver better results than our other vendor’s processes. We embedded agile developers, and hired an agile team coach. Scrum was put in place with a small team. By the time the first beta boards became available we hired the development manager. He has great software engineering depth and experience and was interested in working in a Scrum environment. Agile was less important than the right engineering depth to prove or disprove the technical feasibility. A concerted effort began in earnest to hire high-quality software engineers and testers. Progress was starting to be made and visibility was increasing.

Still, we did not have the business answers we needed.

- Do we have enough budget?
- What is it going to take to deliver?
- How do we roll this new technology out?
- What can we deliver?
- When can we deliver?
- Is it still worth doing?

We were introduced to Brent Barton by one of our product managers to discuss how we might address the situation and with him we learned about Agile, Scrum and multi-level planning from the business perspective. We mapped out a way to integrate the business planning and agile planning. On March 21, 2009 we agreed on the following deliverables and responsibilities sufficient to generate a Statement of Work (see Table 1: Initial Responsibilities and Deliverables). Brent Barton emphasized, “Regardless of what we write in the statement of work, what I will do is drive transparency into the organization and give you tools to deal with what comes along with transparency.” This started the integration of Agile into the organization as a whole.

V. SOFTWARE TEAMS START PROGRESSING

In addition to hiring and implementing Scrum basics, we focused on build automation and unit testing for the many languages already in play. Quality assurance was beefed up to include software testing, not just mechanical and electrical engineering testing. The team size grew to seventeen people by the fourth Sprint (using two week Sprints), and was oversized for a few weeks until one day the team size mattered. The day before it didn’t seem a big, noticeable problem. Weak signal indicators [3] like this are key areas to be attentive for. We split the group into three Scrum teams (Note from Brent Barton: In

<table>
<thead>
<tr>
<th>Deliverables</th>
<th>Precor Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executive Visibility in the Plan</td>
<td>Give authority to spend time with leadership</td>
</tr>
<tr>
<td>The Plan is a holistic view of releaseable hardware and software product</td>
<td>Authorization to form, reform and refocus teams</td>
</tr>
<tr>
<td>Implement 5 levels of planning (enumerate) across scaled teams (multi-level planning)</td>
<td>Apply Just in time learning</td>
</tr>
<tr>
<td>Establish integrated roadmap and release plan in accordance with multi-level planning techniques, including integration milestones</td>
<td>Responsible and accountable to make time to implement change</td>
</tr>
<tr>
<td>Increase insight into value based decision process</td>
<td></td>
</tr>
<tr>
<td>Implement scaled product ownership team for integrated decision making</td>
<td></td>
</tr>
<tr>
<td>Organizational impediments can cause well-intentioned folks to locally optimize to their detriment. Establish a mechanism to deal with this.</td>
<td></td>
</tr>
</tbody>
</table>
times of forming or storming [4] I personally find two Scrum teams of 5-6 people similar in overhead as one too-large team so recommend splitting when other inefficiencies are removed instead of when dogma says to split).

VI. TENSIONS

VII. FROM BUILDING A SOFTWARE COMPETENCY

From a practical standpoint the focal point of Precor’s established product development processes and capabilities focused on hardware development and manufacturing of hard goods. Compared to mechanical engineering planning, software has less predictability and visibility.

TABLE I. INITIAL RESPONSIBILITIES AND DELIVERABLES

The processes that Precor used to manage new hardware-based product development proved to be inadequate for software. Building a parallel competency in software development to rival the company’s expertise in hardware design and manufacturing created tension across the organization because it required a build-up of software related expertise during a two year span where Precor suffered the most significant business decline in its history, all the while funding the biggest R&D investment ever.

The term competency has several somewhat conflicting meanings. For the purpose of this paper, we will expand on the following definition: “A competency is a grouping of the knowledge and skills necessary for the performance of a job task. Competent workers have the knowledge and skills they need to perform their jobs.” [5]. Further, we expand this definition by stating an organization has sufficient software competency when the processes enable knowledge workers to work in a coordinated fashion with sufficient progress that the system(s) under development can be inspected for qualities required, such as described in ISO 9126 “Evaluation of Software Quality” [6]. Further, this competency seems rather hollow unless the products delivered attain market adoption. Culturally, the new level of attention the company was placing on software along with the significance of the departure from the company’s historical focus generated frustration and fear across the organization. Fear was justified because this occurred during a time when the company was forced to cut back in traditional design and manufacturing. This is because it is outside the current core competencies. Geoffrey Moore describes this as a “Horizon 2 initiative” in his book *Escape Velocity* because it has to compete with existing material lines of business [7]. The book parallels our experience in an eerily accurate way.

VIII. ESTABLISHING INTEGRATED PLANS WITHIN A PROGRAM

The existing electrical and mechanical processes had a difficult time understanding how work can flow through software teams. Software delivery needed to be more predictable for obvious reasons. Manufacturing had long lead times. Mechanical Engineering needed long haul testing in harsh environments. Electrical Engineering must balance cost of goods, current configurations, number of board turns, and soon-to-be-available technologies. All of this added significant complexity to software development. Everyone waited impatiently for software schedules to be delivered. Software now drove hardware schedules.

Clearly, we needed to manage this as a program to address:

- Multiple pieces of fitness equipment
- Multiple commercial product lines
- Software platform
- Embedded software
- Internal process changes to support platforms and new levels of complexity

Agile program management in embedded software with platform dependencies on mechanical hardware in a hostile operating environment did not have a well-documented recipe available in 2009. We also had many people new to agile methods and unfamiliar with software.

We needed to establish a planning system that integrated and responded to the adaptive needs of software as well as the long lead times required for manufacturing, materials management and procurement. We decided to integrate multi-level planning at the roadmap level. We adopted and extended the “Five levels of Planning” [8] to support Programs. By adding a sixth program roadmap level, we had six planning levels:

- Vision
- Program Roadmap
- Area Roadmaps
- Release
- Sprint
- Daily

We mapped roles and responsibilities to this multi-level planning structure and created a modified RACI chart (see Table II: Modified RACI chart mapped to Multi-Level Planning). Notice we allowed multi-person accountability at the team level but assigned single-point accountability at the Program level.

Brent Brooks became Chief Product Owner. As Chief Product Owner, Brent was responsible for the Program vision and roadmap. If the release plans or the Product Area roadmaps were out of alignment with the Program Roadmap, the Chief Product Owner had to decide how to re-align this. The CEO maintained accountability for the roadmap and thus had to be part of the decisions when the Program Roadmap changed.

As Chief Product Owner, Brent Brooks collaborated with many people and created a compelling program vision:

*The soul of Atlas is about helping people. More than a display, the Atlas platform sets a new standard for fitness experiences. Atlas captivates and connects exercisers, and gives Precor’s customers new tools to manage their businesses. By delivering experiences that bond people, places and partner services, the Atlas platform will be a catalyst for a fitter world.*
The first line of the vision connected the company mission statement. Line two emphasized that we were creating a platform that would change how we think about products. The vision further identifies users intended experiences and delineates customers as facility managers like club owners. Finally, goals and success criteria are referenced.

We developed and reconciled roadmaps for each “Component” area: Platform, Console and Customer Care. The hardware part of the organization would integrate at the program roadmap since they had their detailed plans. We then created a top down view of the entire Program with a program roadmap (Roadmaps in Brent Barton’s point of view are top-down operational instantiations of a product vision).

We built roadmaps and Product backlogs of user stories. The Program Roadmap was 18’ x 5’ and was in the room where executives met (see Fig. 1: Program Roadmap). Everyone was able to access this room. We had to invent ways for people to interact with the Program Roadmap to address long lead times (using string), issues (color coded stickies), and impediments (orange stickies). We addressed these items each Meta-Scrum meeting. When a Project Manager digitized this roadmap, it became less useful, less modified and less likely to be current. Something about hand-made information radiators gives permission to interact with it.

We established a Meta-Scrum meeting every week to reconcile delivery between different groups and updates to plans [9]. The Meeting Purpose was established with a goal of having one place where schedule decisions were made, whether the decision was continue, change or take action items for the next meeting. This established a new adaptive rhythm in the organization because it recognized that plans could change frequently and the organization had to have the skills to respond to changes:

This meeting has accomplished its purpose when we evaluate and understand changes to the release plan since last week, agree on this week’s release plan, and we know our actions that are required. We will not need to make further decisions about releases until this meeting reconvenes (unless a top-the-line event requires it).

As could be expected, we still had no way to connect Sprint plans to manufacturing plans and long range planning. We had to establish release plans for the software teams and decided we needed a facilitated release-planning event.
IX. MULTI-TEAM RELEASE PLANNING

In Jun 3, 2009, we engaged in multi-team agile release planning. We now had roadmaps, Product Backlogs, and teams with enough evidence of delivery (measured using velocity) that we felt we could develop an integrated plan. Our primary constraint was schedule since we targeted a trade show to launch. We involved the software teams, design, Product Ownership, representatives from manufacturing, mechanical, electrical engineering and management. This allowed us to “outlaw” assumptions and turn them into decisions, action items or risks. We had a three-part meeting objective:

1. Create a release plan that provides the information needed, including risk factors, for the teams to be able to agree on the plan, external teams understand our needs, goals, and objectives, AND executives can make informed decisions in support of OUR plan.

2. Our agenda included an executive kickoff by the CEO and roadmap level discussions at the program and area levels. We mapped user stories on multiple tracks and used breakout rooms for the console and platform teams. The line up of user stories spanned over 100 feet (see Fig. 2: Multiple track release plan). We used the velocity of each team to determine where functionality is likely to become available. Day one gave us data we didn’t want to hear: we weren’t going to deliver by the trade show with this plan. The console Product Owner asked the teams to re-prioritize to see if something could improve. This opened up some possibilities. We ended day one without achieving our objectives and created Day Two. Management got together from product, engineering, and senior levels. We found some opportunities by learning from the teams the day before. We decided to scale more. Money would have to come from somewhere. We proposed the plans to the teams and they agreed to it.

X. MANAGING THE RELEASE

The meta-Scrum served its purpose in the sense that nearly every week plans needed to change. When engaging in something new and also more complex than before, change is expected. This meeting helped us stay aligned.

At one point, however, transparency was lost and conflicting stories emerged about whether we were going to make it or not. The violation of a splitting rule for story points turned out to be the root cause. A whole collection of stories around a particular technology were estimated at 5 points and on average were splitting into five 5-point stories causing a 5x growth. Had this been allowed and communicated broadly, the loss of transparency at the release level would not have happened. Instead, someone told the teams that these stories could only be allocated 5 points total, regardless of how many times they split. This event caused the estimating model of relative sizing to be inconsistent and velocity declined so rapidly that no long range forecasting could be done. In any case, the launch was postponed. If transparency wasn’t lost, the decision could have been made sooner with less angst. This caused trust issues and increased tension in the organization.

On reflection, the meta-Scrum had the right people in the room. The conversations included perspectives whether they would make the release on target. The lack of transparent evidence in the updated release plan caused lack of alignment. This is always dangerous because most of us have a tendency to put too much belief in a plan.
XI. PRODUCT LAUNCH
We launched at the industry’s main US trade show in 2011. The market responded well. The product platform is well on its way to becoming a growth engine for the company.

XII. FUNDING THE PRODUCT
Had the sheer size of the final price tag been known from the beginning this product would have likely never seen the light of day. In this way uncertainty and a level of unpredictability became an ally. While the teams never intentionally mislead senior management in regard to estimates, the reality is that by not fully understanding what it would take from the beginning allowed the company time to “approve” the necessary funding incrementally. In a sense, it was an unplanned “eating of the elephant, one bite at a time” that made the cost of the entire meal more palatable.

XIII. CONTINUOUS IMPROVEMENT
Our team always struggled taking vertical slices all at once Integration was a big challenge. Teams felt too pressured to complete stories and were over-committing with carryover every two-week Sprint. We significantly improved by going to three week Sprints. We also created some waterfall-like targets. We required a code complete date two weeks into the Sprint to give QA time at the end of the sprint and allowed lower priority tests to be completed one week into the next Sprint. Specifically it resulted in better QA/Dev integration and, somewhat counter intuitively, in smaller stories. Prior to this we spent a lot of effort trying to avoid this type of pipelining and longer Sprints. After the change the QA and development teams began to migrate to negotiating smaller stories, earlier collaboration and completing all testing within the Sprint.

Starting from a process that was less restrictive than strict adherence to a short Sprint cycle, our experiments resulted in a Kaizen evolution [10] to agility that was previously unachievable. Software competency and agility is now at a higher level.

We limited Scrum of Scrums to executing on the Sprints and included Product Owners and managers. The primary activity for the top level Scrum of Scrums was removing impediments. Still, the Scrum of Scrum’s value has been limited. We recently integrated a Kanban board [11] to focus on upstream and downstream work in the value stream. We’re hopeful this will provide more efficient communication and collaboration but it is still unproven.

XIV. BUSINESS LESSONS
It took the company time to understand the difference between a “product” and a “platform.” In the past, product development initiatives typically had a predictable beginning, middle and end. These discrete efforts could be planned relatively accurately from a scope, schedule and timing aspect and even if they went awry the average duration of 18-24 months limited the extent of problems a single project could create. By contrast, as the company began to appreciate the difference between a product and platform reality began to set-in. For starters, a “platform” effort requires a big beginning and a long middle. If successful, there really is no “end”, at least not in the traditional sense of the project.

Finally, the customer’s ability became a crucial consideration. Very early on it became necessary to rein in the sales organization from the momentum generated by an enthusiastic market place for this new product. In some ways the product became a victim of its own success by creating a strong interest among customers who wouldn’t or couldn’t tolerate the early hiccups associated with a disruptive new technology. This forced Precor to dial back in year 2 a little in terms of both expectation setting with customers and with their selling strategy. It was determined that a focused effort on more visionary customers early on would provide for a more forgiving early installation base that could be used to work out the kinks and validate the value of the platform before accelerating scale and moving into platform monetization.

XV. LOOKING FORWARD
By building an in-house software competency we can continue to upgrade our platform efficiently. Investing through the recession has potentially put us years ahead of our competitors, but we retain a high level of urgency in our future plans.

More business opportunities have become possible because of the move to a software platform. This makes it more important that we know the difference between what we could do, should do and don’t do.
REFERENCES


