

March 27, 2009

Case Study: Microsoft Speeds Tool Delivery With Agile Development

by Dave West

for Application Development & Program Management Professionals



March 27, 2009

Case Study: Microsoft Speeds Tool Delivery With Agile Development

This is the first document in the “Software Development Process Best Practices” series.

by **Dave West**

with Mile Gilpin and David D’Silva

EXECUTIVE SUMMARY

Microsoft’s TeamArch, which is responsible for the Visual Studio Team System 2010 Architect product, faced multiple challenges that interfered with its ability to rapidly deliver a plug-in tool with the right capabilities. Some of the challenges included fuzzy requirements, a large development organization, and widely varied target customers. How did the team respond? It transformed the way it worked, introducing tighter customer relationships, Agile development practices, and more-transparent stakeholder communication. These changes, coupled with an almost manic focus on delivery, enabled the team to challenge the status quo, delivering faster and with higher quality. The team’s practices also have relevance in other technology contexts, and many application development professionals can apply the lessons Microsoft learned.

TABLE OF CONTENTS

2 **On Time And On Budget Is Easy; Building The Right Software Is Hard**

2 **The Team Incrementally Improves Its Development Practices**

TeamArch’s New Two-Phase Process Supported A Change In Developmental Focus

Feature Crews Deliver Frequently

The New Process Reduces Feature Bloat

Transparent Reporting Leads To Greater Focus, Improved Communication, And Efficiency

Retrospectives Enable Improvement

RECOMMENDATIONS

6 **Introduce An Agile Development Process**

NOTES & RESOURCES

Forrester interviewed Cameron Skinner, partner product unit manager of one of three SKUs in Team System, a part of the overall Visual Studio and Team System development organization.

Related Research Documents

[“The PMBOK And Agile: Friend Or Foes?”](#)
January 22, 2009

[“Make Agile Lean To Boost Business Impact”](#)
December 17, 2008

[“Agile Product Management Makes Agile Technology Companies”](#)
October 7, 2008

ON TIME AND ON BUDGET IS EASY; BUILDING THE RIGHT SOFTWARE IS HARD

TeamArch resides within a 500-person development organization tasked with delivering Microsoft Visual Studio and Team System. TeamArch is responsible for enabling customers to use Visual Studio to deliver well-architected systems. The team comprises 60 people performing roles including quality assurance (QA), program management, and product management. Microsoft measures the team on delivering products that customers love in a timely fashion. Trying to meet this goal while developing the Visual Studio Team System 2010 Architect product, the team encountered challenges including:

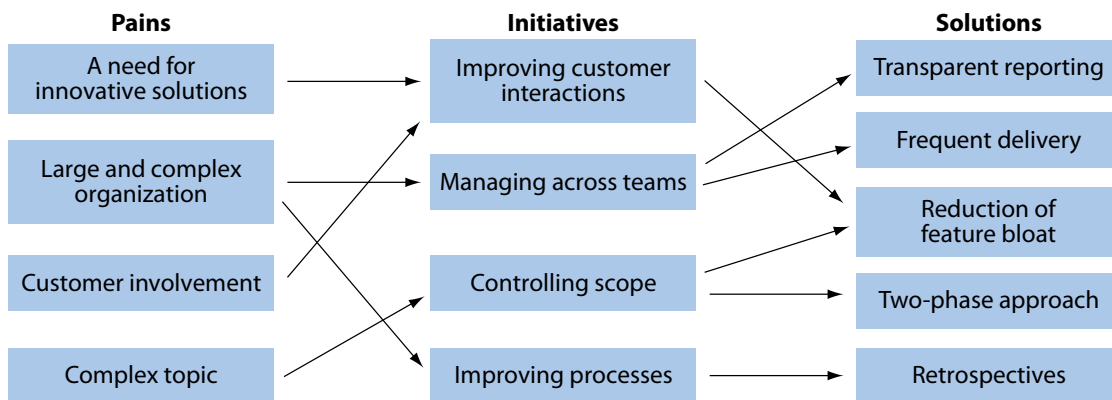
- **Delivering innovative solutions the business seeks.** Microsoft considers it a given that the team will deliver on time and within budget. To be successful, TeamArch needs to be able to deliver added value via innovative solutions. But innovation is difficult when the varied customer base is calling for contradictory features and other internal development teams are calling for integrations to support their needs. Partner Product Unit Manager Cameron Skinner recounted, “It was clear to us that delivering what customers want was not enough — the team needed to deliver something special to really make a difference.”
- **Involving many teams and many components.** Within the past 10 years, the product capabilities of Visual Studio and Team System have grown exponentially. This increased functionality has brought both architectural and organizational complexity, complicating team communication and collaboration. Development teams are located around the world, with developers in Europe, India, and the US; therefore, the team must contend with the additional complexity of working with people from different cultures in different time zones.
- **Fostering customer involvement.** Working closely with a varied and geographically dispersed customer base has always been a problem when delivery dates and cross-product integrations are pressing. This becomes even more acute when a team must consider new features or capabilities that are outside the current scope of the product it’s developing.
- **Building consensus on the nature of good architecture.** Thought leaders throughout the industry agree neither on what makes a great architecture nor on what critical components an architectural solution must include. Working through many different agendas and stakeholder groups is one of TeamArch’s key challenges.

THE TEAM INCREMENTALLY IMPROVES ITS DEVELOPMENT PRACTICES

Faced with increased developmental complexity and tight delivery deadlines, TeamArch focused on four main imperatives. This focus enabled the team to continue to deliver working code, improve the code it was delivering, and increase the overall quality of that delivered product. These imperatives were (see Figure 1):

- **Improving customer interactions.** This sounds like an easy task, but in the past talking to customers led to long, time-consuming processes as well as unrealistic expectations and actually reduced the value of the final product. Getting the right customers involved for the right amount of time focused on the right things was the team’s No. 1 priority.
- **Managing integration across teams.** With nearly 450 other people working hard and delivering many components on which architecture add-ins might depend, understanding what is important and what is not was a key priority. Considering that the team was delivering an architecture product, it’s not surprising that the architecture team defined this as “external impact on our architecture.”
- **Controlling project scope by attacking feature bloat.**¹ The maxim “less is more” became the development team’s mantra: It focused on delivering just enough capability to keep the customer happy, keep the solution simple, and deliver as fast as possible.
- **Motivating and challenging the development team to improve processes.** Making the process the responsibility of the development team and providing that team with the time and power to change it encouraged the development organization to focus on process improvement.

Figure 1 Mapping Pains To Initiatives To Solutions



54034

Source: Forrester Research, Inc.

TeamArch’s New Two-Phase Process Supported A Change In Developmental Focus

TeamArch decided to begin using Agile development practices to better connect with its customer, improve the planning process, and increase predictability and quality.² But it needed to combine that Agile approach with a more traditional life cycle in use across the 500-person development team. Thus, the team combined an iterative approach with a milestone-focused, risk-driven life cycle. In its new two-phase process, TeamArch:

- **First focuses on understanding project scope and mitigating risk.** The first set of milestones focuses the team on understanding the overall value, scope, and architecture of the release. The iterations are more experimental than intended for delivery; at this point in the process, the team operates with an appreciation of the fact that delivering throwaway code is OK as long as developers increase their understanding and mitigate risks.
- **Subsequently focuses on delivering features and managing quality.** The second phase takes the scope and solution defined during the first set of milestones and adds capabilities and functionality incrementally. This phase delivers to an overall release stream with continuous integration testing and review.

Feature Crews Deliver Frequently

A feature crew is a cross-functional, self-managed group of developers and testers. The process they follow resembles Scrum, with the product manager acting as a product owner and a Scrum master replacing the team leader position. The size of each feature team varies between six and 10 members depending on the maturity of the team and the size and complexity of the feature set, and the number of feature teams depends on the size of the release and the number of teams that can work without adversely affecting each other. Within feature crews:

- **Product owners describe features in terms of value, not function.** Instead of describing a feature in terms of how the product will support it, product owners describe a feature in terms of what the end user wants to do. An example of such a description is “the customer wants to be able to understand the architectural impact of a change” rather than “we need to deliver robustness charts and impact assessment reports for each component.”
- **Developers use continuous integration and build to improve quality.** The feature crew continuously delivers working software to an integration stream; thus, all other external teams use the latest version of the software, which means that existing issues or problems will come to light more quickly.

The New Process Reduces Feature Bloat

Historically, feature lists at the start of the project were massive; they listed and described every possible customer requirement. The development team would plan and deliver around that list until it became obvious that it would not be able to deliver all the requirements within the required timeframe. Once the team had come to that realization, features were “culled,” with some being prioritized and some dropped. This often led to confusion, as different groups did not know which features were in or out of a release until the last minute. Microsoft’s TeamArch incrementally *evolves* a release, *adding* and *refining* rather than *deleting* features. This means that:

- **The team keeps the scope of features it will deliver in each sprint to a bare minimum.** The team keeps the list of possible features to a minimum, preferring to add capabilities during a sprint rather than take them away.
- **The team continuously delivers working software, allowing rescoping and refocus.** Instead of spending hours prior to a sprint working out the details of a feature, the team describes the feature in a conversational manner and then further elaborates the feature during the sprint. The result of this elaboration is not long documents but working software that the product manager — as well as end customers — can touch.
- **The backlog runs the show.** Nothing happens within a release without those features being on the backlog with associated work items and development and test assets. The backlog not only runs the project, with developers taking work from the backlog; it is also the mechanism that team members use for reporting and communication.
- **The team includes the customer review board throughout.** Rather than just including a group of representative users at key milestones, TeamArch involves customers throughout the development process, providing them access to all builds and candidate releases. Building strong working relationships with the customer base allows the feature crews' product managers to provide rapid validated customer feedback.

Transparent Reporting Leads To Greater Focus, Improved Communication, And Efficiency

Different stakeholders within the development organization have different reporting needs depending on their responsibilities and focus. Often, supporting many different stakeholders leads to overly complex reports and very complex progress reporting. To avoid this, TeamArch uses the automated reporting capabilities of Team System and keeps additional information to a minimum, relying on automatic capturing of status information. This method tracks status in a way that ties back to the original user stories:

- **Each team breaks stories into features.** Teams deliver the high-level features that support the stories. Status reports for each feature show completeness, issues, and test coverage. This enables teams whose features depend on one another, or who are delivering different aspects of the same feature, to know the status of the feature from the other team's perspective. Building reports around individual features enables describing a release in terms of the functionality that it will deliver and the status of that functionality. Features therefore provide the basis for all communication outside of each team.
- **Each team associates work items with each feature.** Each feature crew plans in detail the tasks it will do for a particular sprint each day. Knowing the status of these work items enables the team to mark progress and highlight issues that are stopping progress. Reporting this sprint burn down is a key artifact for supporting the team in a sprint, as it provides evidence of progress and status.

- **Teams also report quality and test coverage at the feature and story level.** Knowing the status of each story and feature, feature crews can report on overall release integrity to provide upper management with a view of the release and its progress. Managers add more quality criteria — such as internationalization and performance — to this analysis.

Retrospectives Enable Improvement

Historically, managers imposed the development process on the development teams. With the introduction of a more Agile approach, teams own the development process. This ownership comes to a head in the process of the team retrospective, a meeting that happens at the end of each sprint wherein the team asks questions about many facets of how well the team is working. Questions include:

- **Does the solution work?** This provides an opportunity to review the technology the team used and highlight areas that could be improved in the next sprint. This is also the opportunity to kick off “skunk works” efforts to review possible technical innovation outside of the sprint, or to contribute to the Microsoft body of knowledge.
- **What should we do differently next time?** Reviewing the team’s past sprint activities provides the team the opportunity to add, delete, or change the activities it will undertake in the next sprint. The retrospective is also the time when the team discusses its training and support requirements for the next iteration.
- **How can we involve the broader Microsoft community?** Microsoft has slowly introduced communities of practice to support the feature crews. During the retrospective, the team has the opportunity to bring areas of improvement or quality back to the community for its betterment.

RECOMMENDATIONS

INTRODUCE AN AGILE DEVELOPMENT PROCESS

TeamArch incrementally introduced Agile practices in direct response to the challenges the development team faced. To use the team’s experiences to revamp your own development process:

- **Focus requirements on intent rather than solution.** Describing a feature in terms of what the customer is trying to achieve enables the development team to deliver a solution that focuses on value. The development team could also surprise the customer with a solution that is better than the customer imagined when first identifying the feature.
- **Introduce self-managed, self-directed teams that are empowered to improve.** Each of TeamArch’s feature crews has the flexibility and freedom to decide on the best way to deliver

the features for each sprint. Teams also take the time after each sprint to review *how* they worked and improve those practices for the future.

- **Provide transparent communication that varies to fit different stakeholders' needs.**

Because TeamArch's automated information-capturing process centers around the decomposition of story, feature, and task, development leads and managers can report progress and quality at many different levels of the process, which enables different stakeholder groups to understand the project from their particular perspective and take appropriate action. For example, the product management group would like to see how a feature is progressing in terms of test coverage and outstanding defects, whereas executives are interested in the complete status of the release looking at overall coverage and status.

- **Deliver frequently, and involve the customer.** By supporting a process that continuously delivers working software, teams can deliver capabilities more frequently to the customer. This also allows the team to change direction more frequently based on customer feedback. Finally, by writing requirements in a more flexible form rather than investing massively in a detailed requirements process, product owners reduce the impact of change.

ENDNOTES

- ¹ For a more detailed description on how bloat can undermine software development organizations, see the December 12, 2008, "[Lean Software Is Agile, Fit-To-Purpose, And Efficient](#)" report.
- ² To better understand how Agile development practices can help you improve your software development practice, see the August 29, 2007, "[The Truth About Agile Processes](#)" report.

FORRESTER[®]

Making Leaders Successful Every Day

Headquarters

Forrester Research, Inc.
400 Technology Square
Cambridge, MA 02139 USA
Tel: +1 617.613.6000
Fax: +1 617.613.5000
Email: forrester@forrester.com
Nasdaq symbol: FORR
www.forrester.com

Research and Sales Offices

Australia	Israel
Brazil	Japan
Canada	Korea
Denmark	The Netherlands
France	Switzerland
Germany	United Kingdom
Hong Kong	United States
India	

For a complete list of worldwide locations, visit www.forrester.com/about.

For information on hard-copy or electronic reprints, please contact Client Support at +1 866.367.7378, +1 617.613.5730, or clientsupport@forrester.com.

We offer quantity discounts and special pricing for academic and nonprofit institutions.

Forrester Research, Inc. (Nasdaq: FORR) is an independent research company that provides pragmatic and forward-thinking advice to global leaders in business and technology. Forrester works with professionals in 19 key roles at major companies providing proprietary research, consumer insight, consulting, events, and peer-to-peer executive programs. For more than 25 years, Forrester has been making IT, marketing, and technology industry leaders successful every day. For more information, visit www.forrester.com.