Agile Project Management

InfoQueue
eMag Issue 18 - September 2014

Why the Agile Project Manager is the Secret Sauce for Development Projects

The Agile project manager is sometimes referred to as the "secret sauce" for software development projects? Leo Abdala describes a recent development project at a Fortune 50 company where the Agile PM instilled confidence with and produced a value-generating product for the client. PAGE 5

Contents

A Letter from the Editor

Shane Hastie is the Chief Knowledge Engineer for Software Education (www.softed.com) a training and consulting company working in Australia, New Zealand and around the world. Since first using XP in 2000 Shane's been passionate about helping organisations and teams adopt Agile practices. Shane leads Software Education's Agile Practice, offering training, consulting, mentoring and support for organisations and teams working to improve their project outcomes.In 2011 Shane was elected as a Director of the Agile Alliance (www.agilealliance.org)

Project management is a crucial and often maligned discipline. In the software world, project management is mainly about coordinating the efforts of many people to achieve common goals. It has been likened to herding cats – a thankless undertaking that seems to engender little or no respect from the teams who are being managed.

The project manager needs to understand the multiple constraints under which projects operate, far more than the triple constraint of scope, time, and money, which is the simplistic view of the discipline.

They are responsible for balancing the needs of the organisation with the ability of the team(s) to deliver, coordinating multiple conflicting stakeholders, and shepherding the successful delivery of a product that delivers the desired business benefits within a realistic and achievable timescale for a realistic and viable cost.

Project management is about trying to predict the unpredictable, making promises about an uncertain future, and having the skills and knowledge to know when and how to adapt plans to the evolving reality that is the complex world of software engineering.

This eMag looks at project management in agile projects, challenging the myth of "we don't need project management in agile projects."

Leonardo Abdala asserts that the agile project manager is the secret sauce for development projects, and explains why he feels this is the case. He looks at the activities of a project manager in agile environments and discusses how they add value, especially as the project ecosystem becomes more complicated, with multiple streams of work and distributed teams whose efforts need to be coordinated.

Vinay Aggarwal discusses the role of project managers in agile. He addresses five aspects that indicate a need for project management, shows how agile approaches address parts of the problem, and suggests where and how project managers can add value to the development process. He tackles three common myths about managers and management in agile projects and shows the flaws in the arguments often put forward for the removal of project management on agile projects.

Peter Schmidt tackles the integration of agile into the project-management office. Often seen as a bastion of cumbersome process and heavyweight compliance rules, the project-management office is changing, becoming more nimble, supporting organisational agility and taking a leadership role in new ways of working. Peter says "The redefined project management office has begun to integrate itself into this approach by providing resource support where necessary, by acting in the role of change enablement, and by clearing roadblocks in the progress of projects and programs by incorporating elements of the servant-leadership model into day to-day operations."

Robert Pankowecki answers questions about his book Developers Oriented Project Management, which aims to help programmers, product owners, project managers, and agile companies improve their project-management practices and move towards flatter organization. He talks about how many of the traditional tasks of project management can be undertaken by team members, freeing project managers to view a bigger picture and focus on removing the organisational obstacles that so often get in the way of effective outcomes.

In his talk at QCon London in 2014, Tim Lister examines the need for risk management on any type of project and shows how many of the agile techniques are about reducing and mitigating risk. He discussed what risk management is, what it isn't, and how project-management approaches need to incorporate risk management at their very core.

Rolf Häsänen and Morgan Ahlström tackle the endemic problem of over-commitment – promising too much and consistently failing to deliver. They explore the many reasons at many levels why teams and individuals become overcommitted and provide some concrete suggestions about how to manage the flow of work to achieve more consistency in results.

Finally, in his presentation at Agile Cambridge 2012, Tony Willoughby talked about the common roles on an agile/Scrum team and how the project manager is conspicuously missing. He identifies the activities and tasks that are left out of the Scrum framework and shows how a project manager fills the gaps, especially in organisations whose structure doesn't yet accommodate an agile framework and where third parties are involved in delivery of the product.

He explains how the traditional project-management responsibilities need to change and how some aspects of the role stay the same.

In bringing these articles together, we hope to provide you with a view of what project management on agile projects can become and show where the real value lies in having someone take the responsibility for managing the project.

Why the Agile Project Manager Is the Secret Sauce for Development Projects

by *Leonardo Abdala*

According to the out-of-the-box Scrum framework, there is no agile project manager (PM) role. There are other agile methodologies, such as feature-driven development (FDD), that rely on the role of a PM but the PM role is reduced to someone responsible for the administrative aspects of the project, and not necessarily for helping to coordinate the development team and their activities or dealing with resource issues (and also far from the complete traditional PM described in A Guide to the Project Management Body of Knowledge). For example, according to FDD, these are responsibilities of the development manager, not necessarily the PM. An agile PM goes beyond the tactical PM role, entering into project coordination and strategy. The agile PM takes the multidisciplinary skills of the traditional PM and brings a unique familiarity with the fast-paced, change-embracing context of agile projects and frameworks.

The agile PM can be better understood as a unique professional with a very particular set of skills that allows him or her to own part of the responsibilities of two or more roles at the same time, as needed. In the Scrum framework, these roles can include client product owner (PO) and Scrum master, i.e. in one project the agile PM can act more as a Scrum master then switch to a stronger PO role in the next engagement, according to each project's needs – not fully replacing but instead complementing the job of the Scrum master or PO. The agile PM uses project-management expertise to assist both sides (client PO and Scrum master plus development team), filling in the gaps while always aiming for the best outcome for the project and walking the extra mile.

During the course of an agile software project, the Scrum master is typically viewed as the agile PO, who makes sure that the team is using the Scrum and agile frameworks correctly. Yet when considering near-shore projects, the leadership role is most beneficial when it is held by more than one person because most of the time the development team and PO are based in different countries, and working with leaders in each location can ensure the parties stay on track to meet the project's goals. For example, a Scrum master and an agile PM can work collaboratively to lead a project and direct the geographically distant teams, with the agile PM taking ownership of some of the tasks that are typically owned by the business partner to ensure the project moves along smoothly when the client is not physically present to meet with the development team. By taking on these responsibilities on behalf of the client, the agile PM becomes critical in ensuring the progress and success of a near-shore development project.

The presence of an agile PM also works well with colocated groups. However, because distributed teams do not always benefit from osmotic communication since not everyone is sitting in the same room, having the agile PM helping to fill in this communication gap is crucial for the success of the initiative.

Tasks that the agile PM can be responsible for include (but are not limited to): allocating team members (staffing); providing mentoring and coaching; coordinating the development of the product backlog with the PO and the sprint backlog creation process with the project team; developing, executing and monitoring the project's schedule cost/budget; project cash flow/invoicing; communications and risk response plans and procurement management.

Specifically on the PO side, the agile PM can be responsible for holding the kickoff meetings and also for scheduling and facilitating other project meetings as needed. In this situation, the agile PM will take charge of preparing and communicating written and verbal status reports for the team members and stakeholders, as well as for updating and archiving the project documentation as needed (e.g. if a company's PMO requires certain documents to be generated for the project to be compliant, there will be stories in the backlog for creating such documents).

The agile PM is capable of assisting the client PO in properly conveying the client vision to the development team (e.g. by creating/maintaining a prioritized product backlog using value-engineering techniques) while helping the Scrum master ensure that the project has someone playing the appropriate PO role, and that the Agile process is being followed on the client side and not only within the development team.

An agile PM at work

To fully understand the job of the agile PM, consider a recent project for a Fortune 50 pharmaceutical company. The company undertook a software-development and migration initiative with a near-shore development team consisting of a Scrum master and the team responsible for burning the backlog items towards the sprint goals. The team included three strong programmers with different backgrounds (coding, Web designing, database knowledge, etc.), one tester, and one software architect. In addition to the remote development team, the PO and the agile PM, both onsite, were also part of the core project team. The project lasted 66 days and consisted of five cycles.

The project began with a four-day warm up that the team referred to as Sprint 0. During this phase, the team reviewed the requirements created by the client and established the estimated timeline while discussing and outlining the application's infrastructure. One goal of Sprint 0 was to include the client in discussions to clarify questions associated with the business rules so that the client, the agile PM, and the development team were on the same page.

While working through the 16-day sprints, the agile PM played the critical role of facilitating constant communications, including arranging daily meetings with the team, morning meetings with the client, and checking the backlog to ensure on-time completion of the project segments at the theme level, The agile PM also coached the Scrum master to try to anticipate unknown roadblocks. Traditional Scrum practitioners believe that the development team is capable of tracking the backlog and bringing tasks to completion. However, this near-shore team had learned from experience that with geographical distance separating the client and team, the process is better streamlined with a manager in place to track all tasks.

It's important to mention that the agile PM did not have assign specific tasks during the project, as the development the team was self-organized enough to handle that. For example, the developers did not feel comfortable handling some of the complex backlog tasks alone and they themselves requested the software architect's assistance. Also, even though the developers were performing tasks besides coding, such as unit tests and system and regression tests (testing each other's code), they naturally asked the tester to assist them in these activities. This scenario strengthened the buy-in of the team members and reinforced the "power to the edge", which is aimed at achieving organizational agility.

The first day of each sprint included planning sessions, during which the development team worked on the breakdown of the user stories (from the product backlog) into tasks, estimating the hours required for them and assigning team members to each task (creation of the sprint backlog). The client worked with the agile PM and development team to discuss and define each sprint goal, which was then written on the whiteboard in the room where the development team worked.

During the next 14 days, the development team moved forward with implementation and participation in daily 15-minute stand-up Scrum meetings every morning. The agile PM, through a

webcam, participated in the review of what was done the day before and what would be done that day, and communicated any impediments to the sprint goal. The agile PM also participated in a separate daily 30-minute call with the PO to discuss and find solutions for any impediments discussed in the daily meeting. The last day of each sprint was marked by a one-hour demonstration session. The development team presented the working functionalities developed during the sprint to the client and any other stakeholders within the organization.

Bridging geographical boundaries and enabling communication

With the development team and client in different places (in this case, the development team was in Brazil and the PO in New Jersey), final responsibility for each sprint fell to the agile PM. It's worthwhile to note that the agile PM's job was made easier during this project because the development team was in a near-shore location, only one time zone away from the client, as opposed to the eight-plus-hour time difference often associated with offshore projects. To better facilitate communication, the teams set up several ***Live Meeting*** and ***GoToMeeting*** sessions and used a 1-800 conference number provided by the client organization.

The constant communication facilitated by the agile PM complemented the work style of the development team. A high-performance team, the developers placed priority on regular communication with the client, ensuring that both parties were focused on the most current business goals throughout the duration of the project. Combining the communication priorities of the high-performance team and the strong presence of the agile PM as the go-between for the two parties, the near-shore team was able to stay on track through sprints, delivering projects on time and within desired specifications.

Learning from past sprints and building reputations

Throughout the project, the team held retrospective sessions at the end of each sprint, led by the agile PM, who was in charge of coaching the Scrum master and the development team. The goal of these sessions was to uncover ways that they could work better together, with a focus on what specifically went right and wrong during each sprint. In the Scrum framework, learning from the project is just as important as delivering the final product.

As a result of these sessions, the development team solidified its reputation with the client, providing a framework for exemplary work processes and serving as a benchmark for other teams that were working with the client on other software-development initiatives. In addition, due to the constant retrospective views into each sprint, the development team needed less time to prove its case to the client prior to making a decision, which led to less overhead – and reducing overhead is critical in a highly competitive development market.

As the development team built its reputation with the client, it also built team morale, which is very important for agile and high-performance teams. The team began to feel more confident in its work and in its ability to try new ways to perform certain tasks, including suggesting improvement in the business processes related to building the software.

Realizing success

Beyond satisfying the client, the development team also achieved internal successes. Through Scrum and following the lead of the agile PM, the team spent less time behind the curtains developing, and getting faster feedback on its work. It was able to focus on the most important elements of the project, placing priority on delivering the parts that brought business value. With the short, daily interactions, the client knew what was going to be delivered and when, and had confidence that the result would provide value. The agile PM also assisted the PO in preparing executive reports for upper management, to communicate the project value.

Looking back at this project, the development team realized that its success would likely have not been possible without the agile PM to lead the process. Without constant communication, the cost of the project would have increased because issues would have not been detected immediately, leading to larger problems, rework, and project delays. Without the agile PM ensuring that the development team and client were focused on the same business goals, the two parties could have ended up on divergent paths, leading to the team delivering a less-effective product that did not meet customer needs.

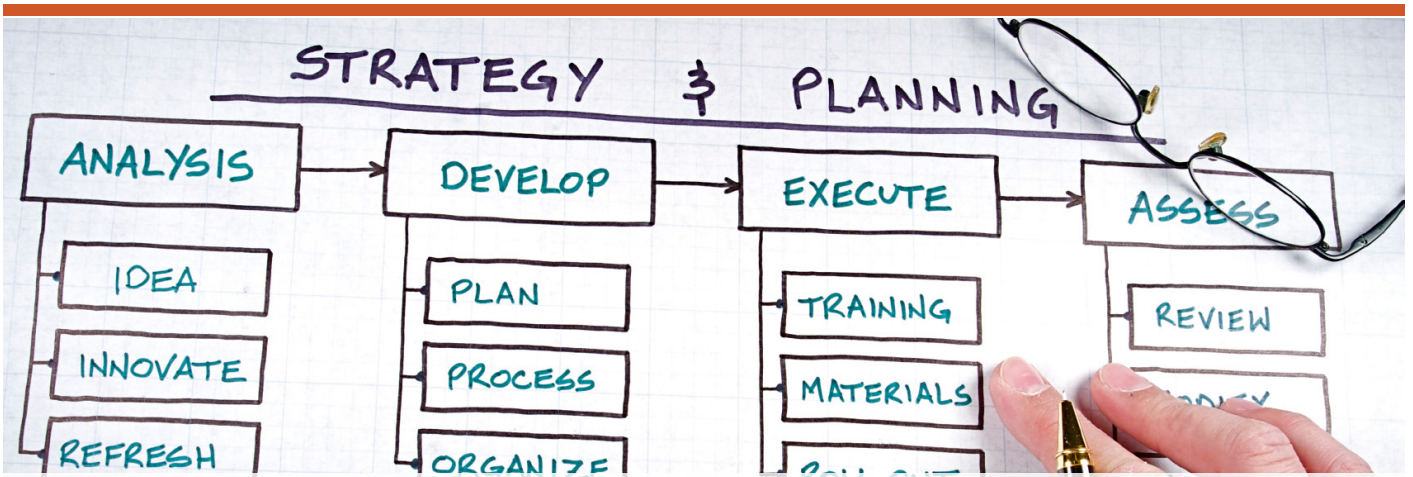
Above all, the use of Scrum and the presence of the agile PM ensured transparency during the project. The stakeholders were able to track the project on a daily basis and valued the ability of the team to rapidly react to necessary changes to

circumvent potentially impassible impediments. The transparency served to increase the confidence of the client in the development team. This combination of a high-performance team and agile PM instilled confidence and produced a truly value-generating product for the client.

ABOUT THE AUTHOR

Leonardo Abdala is a project manager responsible for agile projects involving Amazon Cloud (AWS), Microsoft, and Drupal, as well as mobile apps and mobile-friendly websites developed by Ci&T for international clients, including a Fortune 50 pharmaceutical company, a Fortune 200 marketing and advertising corporation, and a Fortune 500 organization in the healthcare LOB. He is responsible for leading Ci&T's geographically dispersed development and creative teams based in Brazil, Argentina, and China. He has been working with agile for more than five years and with the PMI framework for more than nine years. Leonardo holds several Microsoft certifications (MCP, MCTS, MCPD, MCITP), and is a Scrum Alliance Certified Scrum Master (CSM) and Certified Scrum Professional (CSP) and a PMI Project Management Professional (PMP). Leo is also a college professor (currently on leave) in Belo Horizonte, Brazil. He holds a bachelor's of science and a post-graduate degree in management information systems.

READ THIS ARTICLE ONLINE ON InfoQ



The Role of Project Managers in Agile

by *Vinay Aggarwal*

Agile books usually do not talk of the role of manager but of a coach/facilitator. This article first explains the role of project manager in general in any industry and then tries to map it to the role of coach/facilitator in agile. It also tries to widen the scope of being a coach/facilitator.

Before we discuss the role of project manager in agile, let's first see why managers are required at all in any industry.

1) People are not perfect

Working with human minds is very complex. No two people in the world think alike. Styles of work are as individual as fingerprints but business goals remain one and the same for all stakeholders. "People" in the subtitle above means all the stakeholders involved in the project, like project team members, business users, and management and financiers. People must be managed to:

Keep them aligned with project goals and fine-tune their style of work.

Bring the best out of them.

Help them stay focused and motivated.

If everyone in a project were perfect, no project in any industry would ever fail and there would be no need of any software-development methodology, be it waterfall or agile. Perfect people would always produce a perfect project.

2) Change must be controlled

Change is the only constant in life. Everything can change, be it tangible (e.g. requirements) or intangible (e.g. people).

Requirements are like a wind that always shifts.

People's experience and exposure change day to day. My level of experience tomorrow will have grown by one day compared to today's. This can alter my:

- Aspirations
- Skills
- Commitment
- Attitude
- Any other soft or hard skill

Business is dynamic and the market is changing every minute. With this, customer expectations may change.

With technological change and innovation happening every minute, the project environment, architecture and design, and development processes may change quickly.

Resource movement is inevitable in long projects.

In terms of mathematics, planning is a function of time. However perfect your planning at program level, project level, or sprint level, it may lose its validity tomorrow. Every attribute in planning at any level has an expiry date that can be as close as tomorrow. When everything is changing constantly and unexpectedly, how can yesterday's planning be valid tomorrow?

In this context, role of manager is:

To keep the people continuously motivated and engaged with the project.

Working out resource movement with a realistic transition plan with minimum impact on business.

Keep an eye on the plan, let the plan evolve with time and accordingly take extra steps to manage the impact and change.

Since team members and plan – both are dynamic, keep communicating to stakeholders about impact and mitigation.

3) Communication causes gaps and conflicts

Communication is root cause of all happiness – and of all conflicts.

It's an art that requires diligence and thought. How will the audience perceive this message? Will it offend anyone? Does it have the necessary weight to strongly communicate the message? Few people possess the skills required.

People in development are generally too focused in technology and so ignore, knowingly or unknowingly, this fine art.

A project manager controls the communication to large extent. He or she should also delegate some responsibilities to other team members as and when they are comfortable with it.

4) Processes are not perfect

No process is ideal. Software-development methodology be it agile or waterfall has gaps. No ideal process piously defines customer-supplier relations and even if one existed, it would be almost impossible to strictly carry it out.

Even if a process works absolutely fine with one person or in one situation, it may fail terribly in a different context.

A manager is expected to let the team focus on results and not worry too much about process. The saying "Processes are for us, we are not for processes" means that following process is not the goal but just a tool to get there. The manager along with the team decides what processes work best for this project and apply them.

5) Processes may not be implemented properly

Process implementation always means more work, more diligence, and more tracking, which any development team in general tends to avoid. Many people consider process overhead an evil.

It's rare that a particular process in a project has been implemented 100% faithfully for the entire duration of the project.

The manager should intervene if any process violation may lead to indiscipline and adversely impacting the project, and ensure a high degree of compliance for all good practices.

If none of the above five reasons existed then no industry would ever need managers. Unfortunately, all five do exist in every industry, every company, every project, and every sprint.

None of the five reasons is technical in nature and all are best addressed by the application of management practices. The person who brings management practices into a project is called a manager in the corporate world. Managers have no magic to make the above perfect but they help the people and processes to monitor the project, to fine-tune, and to apply lateral thinking and management concepts, all to find creative ways to make sure that those five reasons do not impact the business goals. Investors and shareholders must get good return on investment in any project so someone has to balance these things and help achieve the business objectives.

A subset of this role is described as coach/facilitator in agile terminology.

Agile coined a new term, the "self-organizing team". I am a big fan of the self-organizing team. It works

well, especially in those cultures where people display high standards of responsibility and duty in public life. These people carry their standards to work and become perfect members of the self-organizing team. To have every employee working in self-organizing mode is the dream of all agile company management.

But human beings are unique and not everyone can perfectly fit into a self-organizing team. Not every doctor becomes a surgeon or specialist but every practicing M.D. is useful to society. Similarly, it's impossible to expect everyone to work in a self-organizing manner, though individuals who do not self-organize can still contribute a lot if handled differently. This is where the role of project manager becomes useful, applying a little or a lot of supervision (depending upon individual) can extract the best work from any team member. Agile uses the term "coach/facilitator" for this kind of role.

Again, this role can work fine even when people deviate a little from self-organization. In following three scenarios, a coach/facilitator may have to widen his scope.

People who deviate too much from being self-organizing because they are highly unstructured, highly unfocused, too emotional, etc.People's soft skills do not align with business needs. They are not proactive, are afraid of speaking, have poor time management, etc. Lack of these soft skills would prevent potential from translating into performance.

People participate in corporate evils like jealousy, withholding knowledge, sycophancy, etc. These people can still be productive provided a strong manager (not coach) controls them with constant monitoring and nips these evils before they start to impact team dynamics.

We should appreciate all professionals but the way to handle and bring the best out of an individual differs for everyone. There is no one rule of thumb that can be applied to everyone. This is something organizations need to grasp. There are very good techies in all countries who can be very good contributors but may not be self-organizing. This is where a coach/facilitator would move more towards the traditional role of project manager. These workers may need guidance and supervision and may be missing the soft skills. The limited scope of agile coach/facilitator would make it a nightmare to align these kinds of people with agile and get the work done.

I have full respect for all kinds of people and strongly believe that this kind can be great contributors, but you need to widen the scope of coach and give him some kind of authority to enforce dos and don'ts. This is where the role of project manager becomes useful. The following table demonstrates a few other areas where a project manager can have impact.

A project manager can extend his or her role beyond coach/facilitator if things are going wrong. He or she can control those team members who are not agile by nature or by intentions. I would like to address three common myths prevalent in the industry. These myths are more prominent in the context of agile.

Myth #1: Managers have magic pills

Dealing with human minds is complex and most challenging. There is no science; it's pure art. Whatever you do, there will still be people who are unmanageable and changes that are uncontrollable. A good manager can:

Completely solve 50% of problemsPartially solve 15% of problemsMake 15% of problems appear to have no impact or be out of scope by making them explicit with the help of communicationAccept that 20% of the problems will always remain because some people or some changes in certain contexts can never be managedThe percentages are only an expression of my experience and are not based on any scientific study or research.

Managers are also human beings who are as imperfect as anyone else. Management is a different concept with holistic approach. It's a different profession altogether, one that is designed to manage imperfect people and processes. People with experience and study of the subject can bring a lot of value.

Myth #2: Managers always curb freedom

This may be true for some bully managers but in reality a good manager creates an environment that enhances performance, thereby bringing the best out of people. A manager with experience and vision may temporarily curb a team's freedom within the context of an objective that eventually benefits them. Sometimes, people cannot visualize that far ahead because of a lack of experience, extreme comfort, arrogance, or other reasons.

Reason	How agile helps	Where agile does not help but a project manager can	Remarks
People are not perfect	Daily status updates keeps focus and a product owner keeps requirements aligned with business.	Is focus is in right direction? Does product owner change goals every sprint? Is accountability shared? Does anyone think people with more experience or more knowledge are more accountable? Are people hiding incompetency on the name of agile? Is the team really self-organizing? Are people finding outward reasons as an excuse for not improving? Is one individual trying to take all the credit thereby disturbing team dynamics? If someone is holding knowledge and not sharing with team?	A manager with lateral thinking can devise innovative ways to manage imperfections. A coach can explain how to do things in the right manner but what if people don't follow? For example, what if a team does not take feedback from the product owner after the demo? Is it acceptable or must it be enforced?
Change must be controlled	Agile welcomes new requirements at the beginning of each sprint and the Scrum master prevents scope creep during the sprint.	Is the Scrum master playing his role properly?Is the tester doing his job at right time? Are people's soft skills and commitment changing?Has the customer stopped believing in agile? Is customer expecting unrealistic results?	Agile takes care of change in priority of requirements. A product owner using his influence can add a story even in the middle of the sprint. What if the team does not know how to handle it? Intangible changes cannot be addressed by any methodology.
Communication causes gaps and conflicts	Agile provides opportunity to communicate every day in stand-ups. Agile creates a platform that a worker can use to speak his mind during retrospectives.	Is the team really raising impediments?Is the team proactive in communication? Does the audience understand all communication?Are there language or cultural barriers to communication?Is distributed communication a bottleneck? Is user experience good?Is all email replied to per expectation with good quality?	Corporate communication is very different from knowing programming and is difficult. Management studies explain the fine art of communication. Soft skills cannot be addressed by any process.
Processes are not perfect	Agile helps in software development.	Every methodology has its limitations. It's people who have to make the project a success.	Bad agile is worse than no agile.
Processes may not be implemented properly	Agile is a process whose implementation depends on people.	Are people following processes?Can processes be improved?What subset of processes is appropriate for my project?Where are exceptions and when it is okay to deviate from process?	Is the team doing excessive pair programming?When is a best practice really the best practice for my project?

It may also be the case that incompetent people fear being exposed and hence they feel that managers curb freedom. People who have a zest to perform should raise their personal bar, and use their manager's experience to plug gaps. They can work closely wit the manager, eventually taking more responsibility and letting the manager attend to other duties.

Myth#3: A manager should not have authority

Some countries and cultures by default inculcate responsibilities and duties in public life. The authority is not required in these cases; a coach/facilitator would work perfectly fine in these kinds of environments. The concept of authority is more relevant in societies that are still evolving and have yet to reach sufficient maturity.

In order to control the five reasons mentioned at the beginning of this article, any manager has to have authority. A manager without authority would be like a car without fuel. Psychological studies have revealed that the human mind (especially in adulthood) is inflexible, like hard iron. In order to shape the iron into a beautiful vessel, you need powerful tools – and for people, you need authority. The moment the people of the world all become diligent, responsible, and highly self-organizing is the moment all management institutes would close globally.

Conclusion

Agile is a software-development methodology that helps iron out some of the wrinkles of the traditional waterfall process. But agile is not a trump card to play for guaranteed success of a project. It's the people who have to work and perform and people are always a challenge to deal with.

The world is full of problems and imperfections. Management is a profession that helps people use processes to achieve business or professional goals despite working within constraints. No methodology can make a manager redundant until and unless people reach perfection. When there are people, there are problems. And every process features deviation. To handle people and problems and control deviation or changes, every project has to ask for help from the management profession. Teams may resist t.If the role ofisn't enoughcan take over

At the same time, managers are also human beings. They belong to the same world of imperfections. Management decisions may fail. Stakeholders must accept this.

ABOUT THE AUTHOR

Vinay Aggarwal is a delivery manager with Xebia IT Architects, India. He has 11 years of experience in the IT industry. He holds bachelor's degree in engineering and is a PMI-certified project manager (PMP) and Certified Scrum Master (SCM). He has worked in companies like IBM and Accenture. He has much experience within both waterfall and agile (Scrum) methodologies. He believes in lateral thinking and applies management concepts to handle various delivery challenges.

READ THIS ARTICLE ONLINE ON InfoQ

The Integration of Agile and the Project-Management Office

by *Peter Schmidt*

Introduction

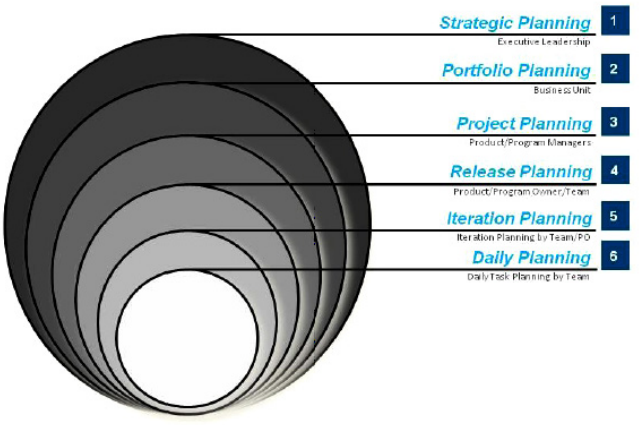
In the early days of its adoption, the agile methodology seemed to be diametrically opposed to the process-driven project-management office (PMO), which most often corresponded to a waterfall-style planning-and-delivery methodology. Agile was a more nimble approach to project management while waterfall stipulated a rigid, document-driven structure. The same is true today, but in the past few years, a growing partnership between agile practitioners and the PMO has emerged. The two are no longer mutually exclusive. In fact, the discipline of project management has evolved to actively include both methodologies in the enterprise.

Organizations often see a need for a blended approach to project delivery, moving away from the traditional project management to a hybrid agile-waterfall methodology. Selecting certain elements from the agile approach such as writing story-based requirements, holding daily stand-up meetings, or targeting shorter development cycles allows organizations to alter their original milestone-driven approach to build in more planning and feedback loops, which in turn gives them more flexibility to react to changes in project requirements. The redefined PMO has begun to integrate itself into this approach by providing resource support where necessary, by acting to enable change, and by clearing roadblocks in the progress of projects and programs by incorporating elements of the servant-leadership model into day-to-day operations.

Where agile makes sense

The agile approach is best suited for projects of an experimental nature incorporating new or untried technology, in which change or refinement of the requirements will be a necessary aspect of the defined product release. Using agile for bridge construction, for instance, makes little sense since the requirements are clear. In order for the bridge to fulfill its function, a set list of prerequisites is needed from the outset of the project. Last-minute changes simply aren't in the plan. Building a software application, on the other hand, frequently benefits from using the agile approach since the desired end-state system may be known, but the details of the technical solution will have to be determined using a sequence of tightly defined iterative loops, or possibly parallel project teams working in tandem on a variety of subsystems.

Six Levels of Agile Planning



It would be incorrect to claim that agile is simply a looser, less disciplined way of running projects. In fact, on the program level, agile teams are even more tightly controlled since the progress of one or more projects is monitored and actively communicated in real time. Unlike the traditional approaches in which reporting is done on perhaps a monthly basis, agile reporting is in fact continuous and runs in tandem with the six defined levels of agile planning.

Where the PMO can play a part

ESI research indicates that agile projects tend to be large and complex, emphasizing a particular need for specialist resource support at critical points in the project, a coordination task that the PMO can naturally fulfill. In fact, in terms of planning, the PMO is heavily involved with the top three levels of agile project planning (strategic, portfolio, and project planning), while the project team itself provides the basis for the release, iteration, and daily planning cycles. The illustration above depicts the top-down and bottom-up interplay of these planning activities.

According to a recent ESI survey on the global state of the PMO, 80% of all agile projects were medium-to-large in scale with a medium-to-high risk profile. Over half of those surveyed said their agile projects were complex in nature. At present, the PMO's major role in agile project management appears to center around coaching and mentoring-support for agile teams. The PMO has some way to go in most organizations before fully integrating itself in the agile landscape and will most likely take on increasing importance as a resource warehouse, inter-project coordinator, and translator of strategic direction into actionable project objectives.

The research

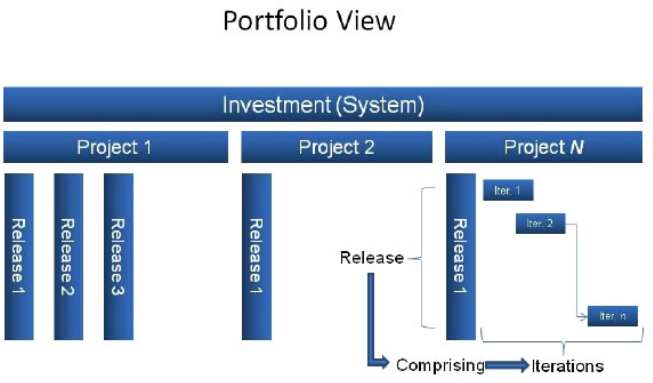
According to a PMI/Forrester survey in 2011, about three out of four PMOs still favor A Guide to the Project Management Body of Knowledge (PMBOK) as their primary methodology. Reflecting traditional planning practices, the PMBOK provides an adaptable framework for a wide variety of project needs. Nonetheless, it is not nearly as flexible as agile. The 2011 survey also showed that only one out of three PMOs fully supports agile, Scrum, or lean practices.

ESI's global PMO study from 2013 revealed that 42% claimed their organization delivers projects using agile methods while 40% claimed they do not. It appears agile usage is on the rise, even if it is not as pervasive as many think. All told only 9% claimed they used agile in over half their projects. Agile is clearly not a silver bullet for all projects.

Not surprisingly, agile projects are typically IT-related in nature with an even distribution of projects among the entire enterprise (38%), division (24%), or department (29%). In the Middle East and Africa (35%), agile was used more commonly for process-improvement projects as compared to the global mean (13%).

In many cases, the PMO acts as a centralized coordinating function for a cluster of agile projects. The management of a group of projects under agile offers improved risk management due the nature of its real-time reporting, offering better insight into the project's status than traditional project-management methods.

The PMO often aggregates information such as the velocity and burn rate of each project, thereby treating complex projects with many sub-projects like an entire program. Because of its position as a supervising and coordinating body, the PMO can reallocate resources as necessary in a nimble, agile fashion. By helping to determine what the team needs, the PMO acts as a partner instead of being viewed as an executive body with little idea as to what is happening on the ground. In this way, the PMO can function as a go-to resource that points teams in the right direction, gathers resources, and brings in specialists as required. The PMO has evolved into a body that orchestrates just-in-time management of critical resources that it can shift between projects, thereby taking a specific agile approach to its resource allocation.



Interdependencies between projects are a focal point for the PMO. Whether for an agile or more traditional project, the PMO is responsible for

program and portfolio cost management and planning. It acts as the financial intermediary and financial buffer, setting aside necessary reserves to cover potential risks.

Hybrid projects such as the US Coast Guard's Response Boat-Medium development program represent a great example of the concert between agile and waterfall. While the hull of the ship was built based upon a set of fixed requirements with a traditional project-management approach, the onboard electronic systems were developed using agile approaches. This tactic helped in saving development time and cost as the parallel approach to the sub-projects within the overall program led to faster overall results in the development of this vessel.

The solution

If agile is so essential for certain types of projects, how can the PMO optimize its support?

Formalizing industry practices such as certifications that reflect the increasing need for agile skill sets is one step toward bringing the traditional PMO and agile practitioners together. Industry standards have been raised to recognize the different and somewhat higher skill sets that an agile project requires. By the beginning of 2013, the Project Management Institute (PMI) had granted over 2,000 agile project-management certifications (PMI-ACP) in the 18 months since it had first introduced the program, illustrating the general recognition that some specialized form of training and certification was needed across all communities of the PMI.

Another PMI report, The High Cost of Low Performance, indicates that high-performing organizations provide well-structured, consistent training opportunities for project managers, which directly and positively impact project success. In terms of on-time, on-budget, within-scope project delivery, the success rate of those organizations surveyed that offered professional development for its project-management professionals far exceeded that of counterparts who did not provide such learning opportunities. As the agile PMP becomes more prevalent, we predict PMOs will become more involved in ensuring project managers receive the training they need in this area too.

The PMO must also show a willingness to forfeit some level of control by stepping out of the way of the agile team's path. Lean project management requires a more hands-off approach than more traditional PMOs are accustomed to. A give and take needs to take place in order for both agile teams and the PMOs that support them to work together.

Summary

In the end, agile teams and PMOs need one another more than ever. Although they may work at a different operational focus from one another, they can learn to collaborate if they keep the successful delivery of working products by means of the project-delivery teams in mind. When both parties adopt innovative communication styles, they can overcome enormous roadblocks. Recognizing the immense value of transparency and access to outside resources can contribute greatly to the agile team's acceptance of the PMO itself. In addition, the PMO must recognize its critical place as a change agent and strategic enabler in the overall scheme of things. It cannot be all things to all people, and most every enterprise will need to develop a unique definition of how the PMO will add the greatest value to the successful delivery of products and projects. Finally, the PMO must remain current on evolving best practices in the industry to stay on top of what is needed to deliver successful projects today and into the future.

Agile and the PMO can indeed coexist. With the right mix, they can enrich each other's existence to maximize their respective value to the enterprise without pulling each other down in the mire of conflicting priorities.

ABOUT THE AUTHOR

Peter Schmidt, PMP, ACP, CPL, services director at ESI International, has over 20 years of project and program-management experience in commercial, government, and international projects. His high-level expertise centers on agile project management, project portfolio management, planning, development and financing. He brings operational and global consulting expertise that spans a broad range of functional and technology areas in such verticals as IT, energy, finance, housing, transportation, and communication.

READ THIS ARTICLE ONLINE ON InfoQ

Q&A with Robert Pankowecki on His Book Developers Oriented Project Management

by *Ben Linders*

Self-organized teams lie at the heart of agile software development. To self-organize, team members have to manage their work, the processes that they use, and the way that they work together as a team and with their stakeholders.

Robert Pankowecki is writing the book on Developers Oriented Project Management, which aims to help programmers, product owners, project managers, and agile company owners to improve their project management practices and move towards more flat organizations.

You can download a sample of this book from Leanpub.

InfoQ interviewed Robert about planning activities, differences between developers and project managers, building relationships with customers, and improving communication in teams and between the teams and their stakeholders.

InfoQ: What made you decide to write a book on developer-oriented project management?

Robert: Initially, we started writing the book to help remote teams. But over time, we realized that many of the practices could be beneficial to any kind of programmer team, including a stationary one. Many of our friends from other agencies were interested in how we work; our potential customers also wanted to know it. Even in our own team, some of the strategies were born and used in one project, but the knowledge had not moved into the other projects yet. So the book is about strategies that are working for us, that we experimented with and found beneficial.

I hear from other programmers that they go to work, get a big task, and spend two weeks implementing it. We wanted to show and document that there is other way to work. And by showing and emphasizing the benefits, we want to give them something that they can propose in their current work environment – small changes to push from bottom up. But this book is not only for programmers. It is also for product owners, team leaders, and even project managers. They can push for the same changes but for their own benefits. After all, in the end, we all want the same thing. Working software, delivered continuously, that can react on daily basis to ongoing changes in the world.

InfoQ: Where does project management by developers differ from that done by project managers?

Robert: I believe that when programmers collectively deal with project-management activities, the result is much better transparency. As programmers, we are accustomed to looking at code that is stored in code repository like Git. The code is created and modified incrementally, and by looking into history of commits,

we can often see the process of creating a workable solution.

Project-management activities when done by project manager are however often sealed off from the rest of the team. Calls and emails are flying between the project manager and the customer but the rest of the team does not have access to it. This communication, when done well, builds trust between the agency and its client. And to know how to talk with the client, how the relationship formed, the fears of the customer that we were able to overcome, that is the knowledge that should stay with the team, even when the project manager leaves or is on holiday.

However, the communication cannot be sealed when multiple people do it, in the same way that code cannot be sealed if we want multiple people to work on it. When the team receives an email from the customer, everyone can see it. When one of the developers answers it, everyone can see the answer. Sometimes the answer is just okay. Sometimes, however, the answer is a spectacular, well-thought-out email that dispels all customer doubts. And everyone learns from such experience. When we see well-structured code, we try to learn and use it throughout our codebase in similar cases. In a similar way, we can learn from others how to communicate properly with respect to customer knowledge and opinions. But the most important aspect is that the communication is transparent so everyone gets a better feel of customer needs, expectations, and priorities. We learn how to work with such a person. It's not a hidden knowledge. It's exposed. We know how to talk to our client. And every client that we work for is unique.

Our superiors like to think that the business value of our product should not be limited by the technical challenges that we face. And that is what we always strive for. However, it is usually fiction. Many projects' codebases are not so clean that we can easily adapt them to every new customer requirement. And many teams lack skills to do that. But there is usually a wide range of possible things to implement and of possible solutions to any problem. When programmers talk with the client about a problem in the code, they almost immediately start to imagine solutions. And they know what is going to be hard and what is going to be easy. Because of their knowledge of the current state of the code, they can propose multiple ways of solving the problem and many times guess more or less correctly which solution is going to be easier and which one harder. So they talk with the customer, for example saying, "I hear what you say. Sure, we can do it. I think it will be about three days of work. But you know what is hard? That part about supporting this use case for unlogged users. If we drop this requirement, and by looking at the server I see that only 2% of our users are not logged, I can have this feature for you by tomorrow. What do you think about that?"

If your programmers can talk with the business and provide valuable feedback like that, that is a huge win. The other agency might be 50% more effective, but your programmer just convinced the customer to drop a useless requirement (by providing actual data) and speed up the development from three days to one day. The best money for your customer is the money not spent. Direct collaboration between programmers and business allows us to avoid confusion in understanding the requirements. And it provides opportunity to negotiate a scope based on developers' knowledge and business needs.

Many agencies fear letting the programmers, especially junior programmers, talk with the customer. They are not confident in their programmers' communication skills. However, there is no other way for programmers to learn those skills but by actively and constantly talking to the client. Engage in the communication to understand the domain of the problem and the real business cases that are the reasons for the software to be built. After all, that's what domain-driven development encourages us to do: to talk to the customer and get to know their domain very well.

InfoQ: One of the essays in your book talks about the developer's attitude to project management. It mentions that developers often expect that project managers have arranged everything so that they can do their work. Is this also the case with agile teams?

Robert: I would say that it depends on the company that you are working for. Some of them claim that they do agile because they've established some of the agile practices like iterative process, continuous delivery, pair programming, or others. But the entire process of talking to the customer and gathering requirements is delegated to one person. Be it a project manager or senior developer, does not matter much. It's still only one person doing the activities.

Sometimes this situation is an effect of the arrangements in the company, and such work style is pushed from top management because of lack of trust in the programmers to do it properly. But sometimes, even if management would like to lean towards more agile and direct communication with the customers, the obstacles are the programmers themselves. They feel insecure and uncertain when doing project-management activities. There is a small chapter dedicated to changing that attitude because it is a critical factor when you want more self-managing teams. Without a proper change of mindset, they will always prefer working with code over working with a customer.

But a great number of companies have adopted agile more deeply and their developers truly collaborate with the customer on daily basis. They do it eagerly and with passion, because they already know how valuable and critical these activities are for the success of the project. I hope that after reading the book, more programmers will also happily adopt this positive mindset.

InfoQ: Another essay talks about how developers can build a relationship with their customers. Can you explain why this is important and how developers can do it?

Robert: It is just easier to peacefully collaborate when we know each other and trust each other. I think the best way to explain it is to compare a situation of the customer only talking to the project manager with a situation in which the customer can discuss the issues with every team member.

In first case, the relationship is built between the project manager and customer only. They get know each other very well and find ways to cooperate. They learn each other. What's important for your agency is that the project manager learns the customer, how to propose features, how to talk about deadlines, what the customer fears, and more. The quality of communication depends entirely on the nature of this relationship. When the manager goes for vacation (or leaves the project or company), everyone else is left with nothing. What used to work smoothly is now at great risk. The whole trust gained by one manager is now gone.

Compare this to a team of five people all talking directly with the customer. When one person leaves, there are still four left and the trust stays at the same level. A new person joining the project to compensate for the one that left must, of course, earn his or her own trust and prove to be valuable team member. But knowledge of how to talk to the customer, how to negotiate, how to discuss scope change, and all of that stays in the team. In the same way that you can have collective ownership of your code, you can have collective management.

The trust and relationship built directly between client and programmers proves useful in negative situations – say a risk of missing the deadline or a personal situation that requires developer absence for few days, anything. It's great when you can just talk to the client directly and explain it from your perspective as human being instead of pushing it to the project manager, who must then explain it to the client, and worrying and waiting for the result of such conversation.

But such a relationship also gives us a space for better direct collaboration and a tighter feedback loop. As a developer, you can directly ask customer for the requirements, talk about your progress, and ask for feature verification. Listen to the praise and deal with customer doubts. This makes the entire work smoother and more pleasant.

So now that we know why, the question is how to build such relationship. I think the answer is by helping to handle classic project-manager activities. Talk to the customer. Help prioritize tickets. Help split features into smaller stories that can be deployed earlier. Challenge tickets that bring no business value. Implement business metrics and propose valuable things to measure that help in making future decisions. Sometimes the best thing to do is to honestly tell your customer that this feature/project/solution is not going to work, and that we should not waste money on that and instead try something else.

InfoQ: There is a whole chapter with essays about communication in your book. What makes it so important?

Robert: As I mentioned earlier, we started the book to help people work remotely. Even though the tools and the possibilities are already here, remote work is still not a popular option. We have few model companies that prove that it is a viable way of working and that they can deliver the software. So it is a possible option and desired by many people

but most of us working remotely were never trained for working in this kind of environment. We spent years going to stationary schools. Our parents, teachers, and mentors could not work that way and they usually had to go to some kind of office. There is no TV show that will show you how people work remotely so you can't learn it from media.

And when you are working remotely, communication does not happen the same way it happens when meeting face-to-face. Remote teams must find their own ways of working or communication just won't happen. And without working communication, you don't have a working team or working company.

It seems that every remote company that we talked to had to invent its own system. So we wanted to share what's working for us so that you can kick-start the beginning and iterate from that. Obviously, what's working for us might not work for you, but let's at least share a list of the practices that may be worth trying so that it does not take so much time for everyone to invent them on their own again and again.

To master the communication aspect is especially important for companies that are half-remote, half-office. I think this is the hardest setup. For everyone-in-the-office companies, communication will happen spontaneously. The fully remote teams sooner or later usually find their own way to keep everyone informed and in the loop. But if you are half-remote, half-office, you can't rely on spontaneous communication because that never reaches the remote workers. I've heard stories about companies with office workers that tried hiring remote workers and later found out that the deal didn't work very well. Usually, it fails because they haven't found a way to reorganize communication so that it is fully, conveniently accessible for remote workers. The whole thing fails not because of a lack of technical skills of the remote workers but because of communication problems.

InfoQ: I'm assuming that there are also similarities, activities where it doesn't really matter if developers or project managers do them. Are they also covered in your book?

Robert: Indeed, there are some activities that we cover in the book for which the end results might be the same regardless of the executioner, such as "split the ticket" (two stories instead of one) or "challenge the ticket" (story removed from backlog). But the problem is that such techniques, while maybe obvious to project managers, are not that obvious to developers. You need to tell developers that they have the authority to do such things, that it is part of their toolbox and should be used depending on situation to make their lives easier.

InfoQ: In your opinion, is there still a need for project managers in software development?

Robert: I always perceive a project manager as someone who plays a similar role in the team to a senior developer or architect, but in the area of soft skills. The senior developer is not the only person coding in the project; there is whole team for that. But it is his or her role to inspire and provide leadership in the technical area so that everyone can grow while working on the project and so the code remains clean. A project manager, on the other hand, should not be the only person doing project-management activities. Entire teams can work on that. But it is her or his role to coach and mentor everyone in areas like how to talk with the client, negotiate scope, manage deadlines, and keep quality high. If everyone in your team is great at that, you can live without a project manager because every dev at any moment can play that role.

But what if your team has not yet reached that stage? You need a coach or a mentor. The project manager can be such a person. But I imagine it to be more like "Come with me and we will talk to the customer and try to gather the requirements," instead of "Let me talk with customer and I will come to you later with my findings." I imagine the project manager to be a navigator through the business part of the project for the developers, not a buffer protecting them from ever going onto such land.

In one project, our project manager had experience in the customer-support team of that project. It was interesting to see this non-technical person be able to judge parts of our code based on the number of problems related to it that were submitted through customer-help channels. This person was able to provide feedback on what's important to the users and on the priority of next tasks because she knew what affected and irritated users. I would have never thought of giving that person some of the responsibilities of project managers. Our customer made a great choice.

InfoQ: Can you name some estimation, planning, and tracking practices that developers can use?

Robert: It's funny because we don't use much of these. We try to avoid estimates. Our preferred method is to split tickets into very small stories until we see the path from current status to desired state of the product. For planning and tracking, we simply use whatever our customer likes. Redmine, Pivotal, Trello – the tool does not matter as much as your process. We don't believe in iterations because we don't see much value in putting in artificial barriers that separate days, which does not bring any value. We try to see the project as constant stream of time and people working on the most important task that they can when they finish working on their current task. The customer can reprioritize tasks on a daily basis according to the always-changing business knowledge.

InfoQ: Can you suggest how communication can be improved in teams, and between the teams and their stakeholders?

Robert: Sure, although I am not sure if that's gonna be revolutionary in any way. :)

First, make sure everything important is written and people can link to it. Surprisingly, many companies still fail at this. They have telephone calls and forgot to make notes. When they do take notes, they send email, but you can't easily link to email. We went with Hackpad recently, which was one of the best things for us.

Remember to link to the discussions and decisions in as many places as possible – especially in commits. You might be wondering whose communication that is going to improve: developers with other developers. They are stakeholders, too. It doesn't really matter what you choose to go with. Pivotal, Redmine, Trello, Hackpad, GitHub... every good tool allows you to link to the ticket/issue you are discussing. If you communicate with stakeholders (clients, subcontractors) via email, just copy the most important decisions into your ticket tracker to the end of the conversation, especially if your email communication is not transparent to the entire team because you are exchanging emails one to one with the other side.

Don't assume anything. You might be thinking that the subcontractors have the same understanding of requirements and the same knowledge about the system as your own but that is often not the case. So please try to be explicit, state your knowledge, your understanding of the problem, rephrase your college point of view and maybe you will find the missing points between your knowledge and the other person's.

I was once writing an API for a few-week-long project. It was obvious to me that it was for a fat client. The developer working on the iPhone part, the consumer of the API, found it obvious that that was for thin client. We were pushing the design of the API in different directions. We realized our different visions few days before deadline (and just in time). We had both assumed and never verified our assumptions. For each of us, our points of views were obvious, and that's how we always dealt with things in that situation – until we didn't. So don't assume, and if you do, go ahead and verify.

Overcommunicate. When something important happens, like a decision is made, feel free to announce it on many channels. Mark the decision in a related ticket. Mention it at the stand-up, write on IRC, Campfire, or Flowdock. Note it in the Git commit message. Don't be afraid of repeating yourself, of communicating the same thing multiple times. It doesn't take much time and you will make sure everyone involved is notified. This is especially important in remote and asynchronous environments when we can't rely on daily visual clues and spontaneous, ad hoc communication between coworkers. We even write on an IRC channel that we just started working or that we are no longer working and away from keyboard. It's great to know whether you can just shoot a question and expect a quick answer from your coworker or not wait for an answer because nobody else is working at the same time as you.

Consider going fully transparent. I am sure the ticket tracker, project documentation, and code are all open for your developers and others to read and edit. How about email or phone calls? Does everyone in your team have access to them? This might sound crazy but opening those channels can greatly improve communication in your project. If we work for a company called FooBar, we will

establish a foobar@arkency.com mailing list that sends the email to everyone involved in the project. We ask our customer to send emails to foobar@arkency.com instead of single developer or project manager. Whenever we communicate about the FooBar project, we CC foobar@arkency.com. It's almost magical to be able to track every email and refer to it in our conversations and to be able to find the decisions even if everyone involved in the communication is currently on vacation. ;) We don't yet record phone calls with our customers (rather, we create short documents that summarize them) but we do record meetings with the customer in a conference room, not as a primary source of documenting decisions (that should always be written to make it easy to find and digest) but as a wonderful backup in case we missed writing something in a self-explanatory way.

Split communication with all stakeholders across your entire team. Instead of having one person (the project manager?) dedicated to communicating with the rest of the stakeholders (clients, subcontractors), make it the responsibility of entire team so they are more engaged. We do it by treating communication efforts the same as developing new features. Whenever there is something to discuss, we create a ticket in our ticket tracker and one of the programmers will complete the task the same way as they implement code features.

InfoQ: Do you have additional suggestions for developers who want to learn more about managing projects but who don't want to become project managers?

Robert: Don't be afraid of pushing for change in your company towards a more friendly environment for you. Do you want to work remotely at least sometimes, to stay more with your family, to avoid the traffic, to spend time in an environment you like? Talk to your colleagues and check if they have the same needs. Push together for the changes that will let you have that. That project that you are working on can't happen without programmers. And you have the right to remain happy in your job.

Managers often try to make developers work after hours (this is even sometimes glorified in the media) and we don't always have the power or the skills to avoid it. Go for the skills and use them to your advantage. Work smarter instead of more. You don't have to become a project manager for that. By acquiring proper management skills, you can help the customers that you cooperate with, choose tasks more wisely, and spend the budget better. That can result in better rates and in more time for you. I wish you that.

ABOUT THE BOOK AUTHOR

Robert Pankowecki is Ruby on Rails developer, working remotely for more than two years. At Arkency, he's worked on number of Web projects in collaboration with small startups as well as large corporations. He created the active_reload library, which made your Rails apps faster in development mode. He's a founder of the wroc_love.rb conference and one of the leading speakers at the Lower Silesian Ruby User Group.

READ THIS ARTICLE ONLINE ON InfoQ

Risk Management Is Project Management for Grown-Ups

Presentation summary by *Shane Hastie*

Tim Lister gave a talk at QCon London 2014 titled "Risk Management Is Project Management for Grown Ups". In it, he presented the advantages – and the dangers – of practicing risk management in an adult fashion while offering a process for tailoring an organization and discussing how an organization can grow up.

He discussed what risk management is, what it isn't, and how project-management approaches need to incorporate risk management at their very core.

The first time I led a project, the team had four members. My manager, Nancy, said, "I want you to drop a project plan and a timeline for me, and I'll sit down with you and talk about it."

So I chewed it up and I looked at this thing and I thought this was a crock. I took it to Nancy. Sitting in her office, I said to her, "This is my plan. I don't know what's going to happen but I do know it's not this." And she laughed and we talked. I said, "This plan is all sunshine. This is 'everything goes right'." I said, "I'm not an old guy but in software projects, it never goes right."

Somebody does win the lotto but, if you notice, it's never you, right? Somewhere there's a project that is on time, on budget, sunshine and roses, but it's never you. It's never your project.

All my career, I've wondered why organizations try desperately not to discuss the risks, the uncertainties, the unknowns that are there at the beginning of a project. To me, a project is a discovery. You start with a lot of unknowns and by the time you actually deliver, most of them have become known. And some of them turn out not to be what you thought they were.

You must deal with unknowns, with uncertainty, and with probability. Is it a 10% chance this is going to happen or is it a 90% chance this is going to happen? – that kind of stuff. It seems to me that how you do so is the difference between childish behavior and adult behavior.

When you're a kid, you're immortal. I have two sons and if you have children, you know there is an age where they just think "Nothing bad could happen to me". We were driving up to the mountains of New Hampshire to visit their grandfather, my wife's dad, up in the mountains in the summer. These guys do amazing cliff-climbing and rock-climbing stuff. We're driving up and a voice in the backseat says, "We're going to climb Cathedral Ledge."

Now, Cathedral Ledge is like several hundred feet – and it wasn't a question, like "Dad, can we climb?" It is a statement, a declaration.

And my wife says, "Oh, no, you're not."

I said, "Wait, wait, you want to climb Cathedral Ledge? I think we can compromise." And my wife is looking at me with the bullet look.

I said, "I tell you what. You give me a week's allowance each. I'll pay for the difference and I'll give you a morning of climbing lessons together with a teacher for climbing. I'm not going to teach you this. I have no interest in climbing Cathedral Ledge." My wife was looking at me.

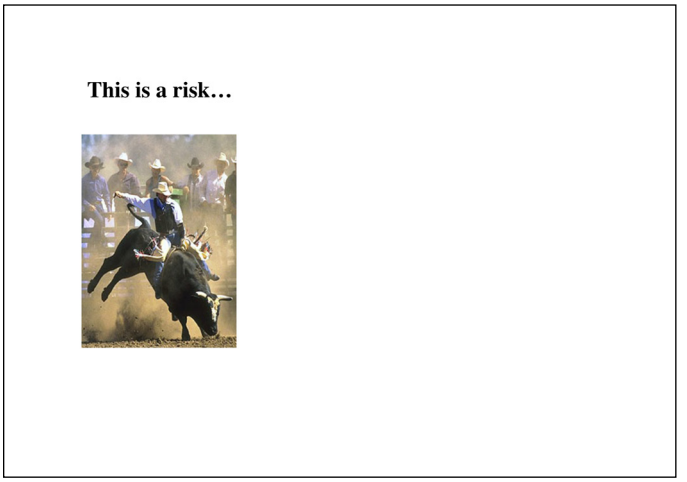
Later in private, I said to my wife, "If you tell them no, they're going to climb something, right?"

You can't say no because they're going to climb a building or something. They're crazy. Right now, these people are insane. On the other hand, in this presentation room, I am sure there are people who have gone skydiving.

When I was a young guy, I went scuba diving all the time. That's risk-taking, but you work really, really hard to minimize the risk, right? You learn. You have to get certified. You never let anybody touch your gear. You never dive alone. There are all sorts of things you do to minimize the probability that you are going to die and to ensure that you are going to have a great time exploring the underwater world for an hour or so in nice, warm waters somewhere.

So it's not the behavior. It is the approach. My entire career, I've been trying to talk to people about big, hairy projects, saying all this project planning. You've all got to understand. There's a giant amount of unknown in there. Don't tell me that you can look today and see everything that's going to happen two years into the future. Bull, nonsense. And the way we deal with this is to say, "Here's what I know now and here's what I am going to have to figure out. We, as a team, we're going to have to figure it out as we go, what makes sense."

I am not going to talk about it directly but let me just say at the start that I think the agile movement is deeply about risk management for many typical risks in software projects. It's really a reasonable strategy, right? It's a highly iterative, steering kind of mechanism, a "we don't know what we don't know" kind of thing. So we're going to go out and then we're going to make another decision. We're not going to plan everything and then go do it. We're going to say, "Here's a little piece we want to do. Everybody cool with that? Can we do this? Okay, let's do it. Let's do it. Let's show it. Let's add on or change course and we'll increment our way into the future where at any given time we don't have a vast investment full of assumptions and the way I think about it."



So here's a risk, right? I am going to try to define risk by example here. Here's a risk. This is a very American risk. This is a bull riding. Think about it. The best outcome for this bull rider is he stays on the bull for eight seconds. You have to ride for the full eight seconds. You can only use one hand. The other hand is out. They time you and after eight seconds they have like an air gun that goes, "Boo!" And you then try to gracefully exit the bull, right? If the bull has just gone crazy and you stayed on it, you get a high score.

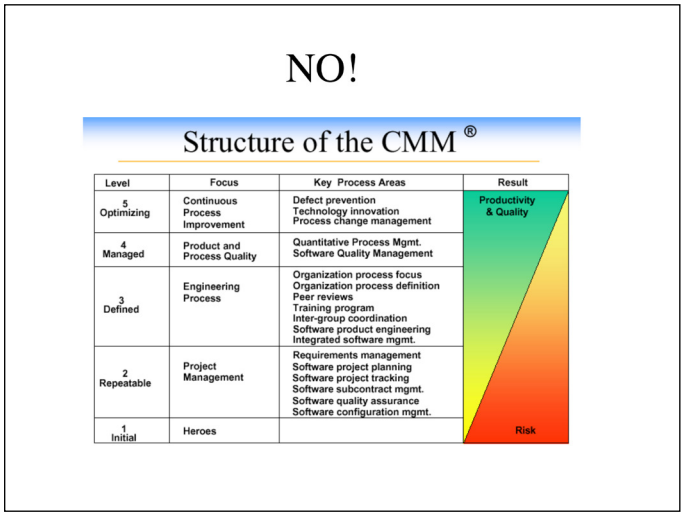
So if you are the highest scoring bull rider who successfully rides the bull, you win some money and a big belt buckle. I know this sounds strange, but you get a championship belt. That's the upside. And the downside is death, right? And you're in there somewhere. I mean people get seriously injured on bulls.

It's a crazy thing, but it's just a risk, right? It's a potential problem. There is in the outcome, you ride the bull and jump off, you win money, they give you a buckle and you feel great. That could happen. It's going to happen to one rider at the rodeo, I guess, right?

A couple of years ago in this Mexico City bullring, a bull managed to get over the wall and into the crowd, the people who actually paid money to go watch. As it turned out, no one was fatally injured or anything but this is a problem, right? This risk is there. This is not chance. When the bull is in the seat next to you, you've got a problem, right? I've done some work in Mexico City and I've become very good friends with one guy down there, and he was telling me about this. All the bulls have names and the best irony is that this bull's name is "el Pajarito". How ironic, the little bird flew into the audience.

So, a risk is a potential problem. A problem is a risk that is upon us. The probability of this risk becoming a problem is 1 – it is here. The bull is next to me.

We can't avoid risk. This is one of my favorite slides and the people at the old CMM always say, "Oh, we should never have made the slide, Tim. You're killing us." Remember the old CMM with the five levels and the CCMMI of software development maturity? I'm not going to talk about that at all, but what I love is this part right here.



At level one, you're heroes, smart people working really hard to build software, right? And the risk is everywhere down here at the bottom. Productivity and quality is a mere glimmer, a single data point or whatever. But if you can climb to Repeatable, to Defined, Managed, Optimized… all of a sudden risk goes away and productivity and quality is everywhere. Level five is nirvana. It is all productivity and quality and there is no risk, none at all, right?

This is the biggest, stupidest graph I've ever seen. It's all about risk in our business, right? Say we're a company. Everybody in this room is a company and I'm the big cheese. I am the managing director. And I believe that the CMM scale comes from God's lips. I believe this is true. So I call up the people at Carnegie Mellon University, whatever, certified assessors to come assess our software company. And they assess. Now, in comes the software assessor and he says, "Ladies and gentlemen, I've got great news. This is unheard of. I had to call back to CMM Global Headquarters before they let me tell you this. Simply put, ladies and gentlemen, you are a six. We didn't even know there were sixes. This is like finding a new element for the periodic table. You are simply the best software development group we have ever seen bar none. Thank you very much." And he leaves.

I, the CEO, go, "Okay, we are the best on the planet. We have evidence of this or opinion of this. What should we do? We have got the greatest group of software developers bar none. How do we use it?" The answer clearly kind of has to be that we don't do any projects that any bozo at level two or three could do. We'd be wasting our skills, right? We'd be underinvesting in ourselves. We have to do stuff that people will have a hard time following. With our expertise, we're going to go take it to our competitors in the market and ram it right at them. We are better than you are.

How would we know that we are doing the best, the most complex, the most wonderful things we could do? I would argue that one easy metric would be that some projects are failing because we're right at the edge of our capability. These failures would be a good sign some of the time, right? If everything is all happiness and sunshine, I don't know if I am stretching my people and I am getting full use of their experience and their abilities.

I know what CMM is trying to say here. A certain kind of estimating risk goes away. You become more predictable. I've seen companies who are really predictable and build really boring software. If you are really predictable, then you are doing the same old things over and over again.

And if there's anything in our industry we've seen, it's that. Haven't you ever noticed this? As soon as you get really good at something, you never do it again. It's part of why I love this industry. Just when you think you've nailed it, your people are good, you're really talented… – oh, the world changes on us. Oh, we don't do that anymore. But you've got to surf the technology or you'll drown. There are always going to be unknowns.

I am not trying to avoid risk and I am not trying to be a daredevil. I am not trying to run toward risk for no reward, but I am willing to embrace risk, to say, yes, there are risks on this project, there are uncertainties on this project. But if we can conquer them, we will be rewarded for conquering them.

What Does it Cost to Build a Swimming Pool?

Here's another example, from a bunch of years ago now. Richard is my brother-in-law's brother and is married to Kathy. They've got a bunch of money and they live north of New York City in kind of this hill 'n' dale horsey area of rich people who have horses and that kind of thing. They decide they are going to build an in-ground swimming pool in their backyard. They want that. They got a couple of kids and Richard loves to swim so they're going to have their own pool.

And so they build a spec. This is a true story. Richard told me this story. They specify not only the pool but also the area around the pool and what is going to be – have slate, all the stuff. And Richard is even into what diving board he wants. They get these little metal flags and go in the back and place the metal flags so that, I forget, yellow flags are the edge of the pool and the larger blue flags are the terrace area that is going to be around the pool. They write a spec, literally, a written document, and make copies. And they start to call up people who build pools.

Whenever a person who represents a pool company comes by, they hand him the spec. They walk out in the backyard and they talk and there are measurements of how deep the pool is going to be and all this sort of stuff. And they say to the person basically, "If you have any questions, ask away. If something comes up, call us. But in the next couple of weeks we'd like your company to give us a bid of how much it's going to cost you to do what's in the spec."

The first guy comes through and walks off with the spec. The second person does the same. The third person, an old, kind of rough guy, looks at the spec and doesn't even look at it too much. They go in the backyard and talk, and Richard says toward the end, "We'd like you to bid."

The guy says, "You wonder how much it will cost to build the pool?"

"Yeah, yeah, I'd like a bid."

"We're not bidding."

Richard asks, "What do you mean you are not bidding?"

And now, I want you to remember this for the rest of your professional life. Here comes a sentence that is going to be in my brain until my brain no longer works. The guy looks at Richard and Kathy and says, "Do you know what's under the grass?" I love that.

And Richard kind of goes, "Uh, what?"

And the pool guy goes, "Buddy, buddy, buddy, over there. Look at that. That's rock. It's not far away. If this is just dirt, topsoil, I'd bring in a backhoe. I can excavate in less than a day. But if we scrape off maybe six inches of topsoil and we hit Mother Earth, we are talking about blasting. We are now talking about having to get permits from the town, notifications of your neighbors, all sorts of safety controls. Until I know what's under the grass, I can't bid."

And Richard told me, "I was stunned but I totally got it."

And so there is silence there and the guy goes, "Look, here's what I am going to do. I'll bring you the backhoe, but not with the hoe. We can drill probes down, just drill head. And I am going drill in different places where you think you've got the pool and I am going to tell you what I find. And it is going to cost you $250. And I will give you the report whether we win the bid or not. Once we know what's down there, we will bid. And if you accept our bid, $250 was on us." I forget if the number really was $250.

Of course, you know what happened. Richard and Kathy, being reasonable people, said, "This guy is a straight shooter. He's honest. What was with the other two people? What were they thinking?"

And I started laughing when I heard this. Richard is not in our business and I was telling him I know those two other people, who say things like "This project will be done in a year. Oh, my goodness!" and "Oh, I am so upset. It's 10 months into the project but we've got at least six more months to go. Wow, I am as stunned as anybody else."

I mean, at some point, people in our business are not going to get that "Oh, I'm shocked and dismayed" routine. What a bunch of nonsense. Maybe we can say that we hope to be done at 12 months, but there's all sorts of things we don't know about that make that only a guess. It's a guess. We got to go a certain ways down before we can say, "You know what? Here it is."

In our business – I will not name this company – I know of a woman is running a project whose company is like, "We've got to get it done by the end of the year."

She's going along and she says, "Yes, I understand. I understand why. I understand what it costs the business if we bleed over into the next year. I understand that. I want it, you want it, okay. But I am not committing."

In month eight with three months to go or something, she goes into a meeting with her boss and says, "On my reputation, we're going to make it. I can see the end. My people see the end. We're going to make your end."

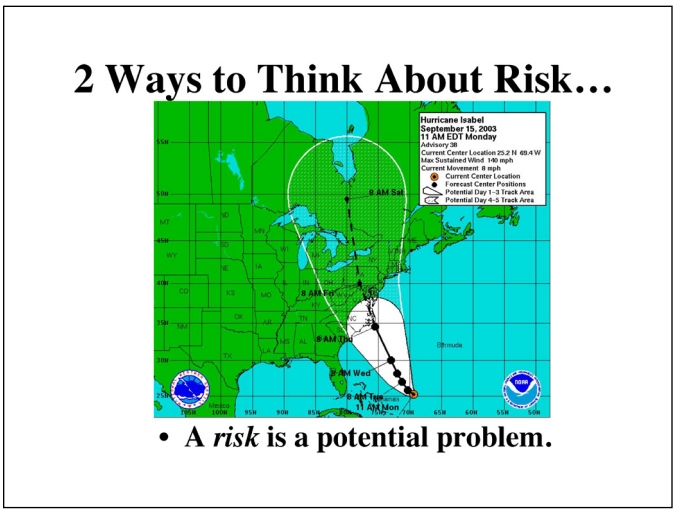
And the boss says, "Are you sure?"

She says, "Basically, yes, I am sure. Unless we get the plague going through the office or something, yes, we are going to make this."

Three hours later, she gets called back in and the boss asks, "How about November?" That's what we are dealing with, right?

The whole point is the world is full of uncertainty and we have to proclaim it in our own work.

This is a picture from the National Weather Service, their National Hurricane Center. In the United States, we get hurricanes in the late summer and into the fall. And this one is from all the way back in 2003. This is Hurricane Isabel. I didn't want to pick any other ones that had been brutal.



But look at this graph. I don't know if you've ever seen one before, but it's a wonderful graph. It says, "This is advisory number 38." So I am assuming there had been numbers 1 to 37. And it says the current center location, the eye of the hurricane, is right there at 25.2 north, 69.4 west. Maximum sustained winds are 140 miles per hour. That's way over 200 kilometers an hour. Current movement is only 8 miles an hour. The storm is moving very slow, right?

The map indicates that the brown is the current center. The black is center positions of the forecast. It's forecasting the hurricane to go here. Right now, it's 11:00 a.m. on Monday and they're saying by 8:00 a.m. on Thursday, Isabel's eye will be right off the coast of North Carolina. But the forecasters also say they're not sure of that. There are so many variables that steer hurricanes. There are all sorts of uncertainty.

So the map gives us a potential one-day to three-day tracking area, this teardrop that says, "Hey, in the next three days, we think the storm will be contained in that teardrop somewhere. But we don't know." And the forecast four to five days ahead blooms all the way out to that outline. But by Thursday, this map tells us that Isabel could be as northerly as the state of Delaware. The most southern projection is all the way down by Charleston, S.C. We're talking about a width here of what, 600 miles? Something like that. A long way.

In our business, some manager will say he needs to know where it's going to hit. And you reply, "Well, the mostly likely probability is right off North Carolina." No, no, no – that's the most likely single point but that doesn't tell you anything. The teardrop range tells you everything.

Say, for instance, you own a boat in the water in Virginia. What do you do? Do you run to the marina and get them pull up your boat and put it up in blocks safely on dry land? Maybe the marina is going to get wiped out. Do you try to sail your boat out of the teardrop? Sail north to New Jersey? What do you do? The answer is there is no right answer until you tell me about your boat.

If your boat is a small fishing boat that you can pull on a trailer behind your car, you don't do anything on Monday. You wait for more information because when advisories 39 and 40 come in (they come in on a regular basis), you can better gauge your odds of getting hit. All you need is an hour and a half to go down, put your boat on your trailer, and pull it to your backyard or wherever you think it's going to be reasonably safe.

On the other hand, if you're some rich dude with a 40-foot sloop, you have a different decision to make. It costs a ton of money to take a 40-footer out of the water. And if you are going to try to sail, you'd better be sailing now. You better go as soon as possible, right?

So sometimes, risk management is really about when you make decisions: based on what information you have and baseline information, when is the time to take action? If you've been around, you know you never have complete information, or by the time you have complete information, you've lost all sorts of options. So you are always making decisions on partial information, right?

But if I wait for better or more information, do I lose options? I've watched many projects wait until the problem is there, and then they have problem management, not risk management. And so I always complain to them. Your position may happen to be project manager but the real project manager is time. Time is managing your project, not you. You're just reacting. You're not managing.

A risk is any variable on your project that within its normal distribution of possible values could take on a value that is detrimental or even fatal to your project. That's the way I think about it, a potential problem. Interestingly, the risks take on this kind of curve called a Rayleigh distribution. I am not going to talk too much about the Rayleigh distribution but it has a steep rise, peaks, and then has a long tail.

So let's say we're trying to decide when we are we going to deliver. We say there is no chance of delivering in the next four months. The first chance is just after four months but the most likely delivery time is about month six-and-a-half. We could take as long as 14 months all the way out there in the tail. So we're looking at 4 to 14 months is what we're saying. Tom DeMarco calls the most optimistic projection the nano-percent date and says most projects are run with the expectation of delivery on that date. It explains why there's such skew.

The boss asks a team when they think they could possibly be done. And young, naïve people who want to please go, "If we catch every break, we are really, really good, and the wind stays at our backs, we could get it done in four months."

And the boss smiles and goes, "Let's go for it." It's at the nano-percent date. It's winning the lotto, right?

Tom and I years ago collected data on real projects. We asked people done at certain places to estimate the amount of work and a completion date, and to estimate it every month until they actually finished by our definition of done. We looked at over 100 projects.

My dad is a mathematician so I took our data to him before we sent in a refereed paper. I didn't want to be made a fool. My dad is a mathematician. He's an algebraist. He's not in the software world. And I was worried that he was going to go, "You moron! You should have used a correlation coefficient," or something like that.

So I show him the data. He's looking at this. He's not trying to be funny. He's looking at the data and he says, "These people originally estimated 14.25 calendar months to complete and they were done in 19." And he's looking at the data. Again, I will never forget this. He goes, "These people have no right to estimate anything less than the season." Given the data and the precision, he concluded they should only estimate in quartiles of years. They had no right to say 14.25.

My dad said, "In math, this is a joke. Oh, you silly people." And he basically said that we could legitimately estimate that we might be done next year when there are tulips. Next spring, when it's sunny and hot, we could be done then.

My dad also said, "Everybody is late."

I said, "Dad, it's not really that bad. Look, there's a couple here really close. Dad, look, people are on time or they're late in our business."

And he goes, "There's another possible outcome." At first, I didn't get it. "Early," he said.

I said, "Dad, I've never heard of anybody really early. Or if they're early, they deliver much less functionality than they estimated they were going to deliver."

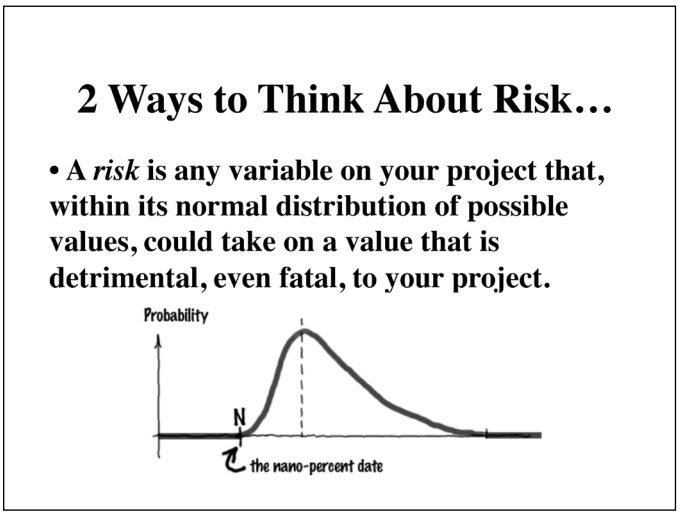
He looked at it and said, "Tim, skew, skew. Remember, this is skew. Everything is late or on time and there is nothing in that other domain. Something is going on that's skewing this data." In other words, it's not an estimate at all, the way other people are talking about estimates. I've always thought that's kind of interesting.

We can't avoid risk, right? All projects benefit but none or few can take little risk anymore – as I like to say, those were all done when I had a different hair color. All the low-lying fruit is gone; the easy things that benefit your organization or make a swell product are basically gone. And you can't control many of the variables that could be risks, right?

I am an arbitrator. I had a case where a software vendor convinced a company to use a product even though both parties admitted that the current product couldn't do what the company wanted. The company accepted the vendor's promise that version 8.0 would be ready by the time they needed it and that version would have all the required features. It got even worse that that. The vendor said, "Buy version 7.2 now and we'll give you version 8 for free."

I don't know why these idiots reached for their wallet. They bought version 7.2 and guess what? Version 8 did not come out as announced, and it didn't come out and it didn't come out and it didn't come out.

Finally, after having the project all done except for what they needed version 8 for, nine months later, the arbitration began. There still was no scheduled delivery date for version 8 at that point. They couldn't control it but, you know, if you say, "Hey, has software ever been late?" Ding, ding, ding. When some salesman says it's going to be there in plenty of time, why are you buying that? What are you smoking? Come on, guys.



I love this picture. The problem is that avoiding a risk usually lowers the value of the product. You can't be risk-averse. If you run away, you devalue what you are doing. Usually, the risks, for whatever reasons, tie to the value. Say you have a feature that is really hard and raising all sorts of problems. Yet if you don't have this feature, a lot of customers won't be that interested. Yeah, there are risks there. That is the way we usually work.

From my point of view, in this case, I will tell you to run toward that risk. Let's get the team to go right at that feature because if we can't build it, the rest of the project is probably irrelevant. Let's go find out. And it's a totally honorable outcome, I would argue, if the team tries to do this but discovers massive problems and difficulties and we cancel the project. I'd buy everybody lunch and say, "Great job, folks! Do you realize how much we save by going after this and finding out that unfortunately this is not going to work out for us? Look at all the other things we didn't do and waste our time on. Now let's go do something reasonable."

Again, from my point of view, agility that's best always ties to risk management to say here's what we got to do because until we have this as a known, we are just dancing in the dark.

A Risk Ritual

- Identify risks.
- Keep the tribes separate.
- Assess risk exposure.
- Determine which risks to manage.
- Form action plans for direct risks.
- Form mitigation plans for indirect risks.

- Determine a contingency fund.
- Build tripwires into project plan.
- Keep the process going....

A risk ritual…. I am not much of a process person, to be honest with you. People over process is the name of this track. But I often believe in rituals, and rituals exist to protect the people in the ritual. Think about weddings. The ritual of a wedding often involves two families getting together beyond just the bride and groom. But the ritual is down to the bride and groom having to say "I do." That's all they've got to do, and they are central to the ritual but all the other things that go on around them, that whirl around them, the reception and all the service and the rest of it protects them. The bride and groom basically play a tiny role and are protected by the rest.

As you get older, you get to go to funerals. They protect you because they allow you to mourn. A man crying on the street is very upsetting. If you ever see a grown man crying on the street, you wonder what the hell is going on. But men crying at a funeral are fine. It's a ritual that lets you get that out of your system. It's saying goodbye. It protects people and lets them emote in ways that most of the time would be odd or strange or unacceptable.

Unfortunately, in many organizations, the enumeration of risk is very upsetting.

A lot of organizations I see say they do risk management. And if you look at what they do, at some point early in the project, you see them make a list of risks, file it, and then move on. That's not risk management. That's naming of the risks and chances are good they only name ones that are palpable threats to survival or that kind of thing.

So I want to risk ritual in organizations where naming of risks is considered whining, is considered naysaying. As a consultant, I say things that no employee can say because I'm like the court jester. I can say things to the king that the citizens can't say. Things can go badly for the project manager, team leader, lead designer, or whatever who says, "I don't know. We'll give it our best shot. I can't guarantee anything." Bosses can start to look at you as if you're not a team player when you don't have that happy, smiley face on all the time.

A happy project manager can say, "I can do that for you. I can get it done by the end of the year. No problem." But at the end, they say, "Oh, I am shocked and dismayed."

No one remembers the other person who warned, "I don't know. This is really dicey. There's a good chance we can't make it. No, no." They look for the person who has the happy, smiley face who begins, "Ooh, boy, sunshine!"

So I like the idea of starting with a ritual of a bunch of steps. We identify the risks. We assess the risks. For risk exposure, we determine which risks to manage and which to accept. We form action plans for direct risks, risks that we can attack, to minimize the probability that they will happen or to minimize the cost should they occur. We have mitigation plans for indirect risks, risks we can't deal with now but for which we're going to have to have some sort of risk mitigation once we see that they are happening.

A contingency fund is a slush fund of time, money, or whatever else you want to think about. Stuff that's going to happen, although I don't know what it is. For example, I need six months to complete the stuff that I know I have to do, but I add three months for stuff I don't yet know about. This becomes a nine-month project. And if we get lucky, I'll give those three months back. We don't need them. I am not saying that I am going to burn them up no matter what, but I am saying that I will not tell the customer that this is a six-month project. I am saying it is a nine-month project. Six months I can see and three months are underwater, I think.

Build tripwires into the project plan to trigger a warning. How do you know a risk is turning into a problem? You smell smoke, so to speak.

And repeat this process. There's no reason to do this only once. Again, it's discovery. You've got to build a rhythm of revisiting your project risks while asking if you see things differently now. Do you know more? Can you retire a risk? You passed this particular risk successfully so retire it. Guess what? Every other risk just moves up the list.

Tom and I have done a lot of consulting. I have never seen a project that has completely passed through its risk list with more work to do. It's really interesting. I have never seen one. I am not quite sure why that's true, but I think it is. Many risks you don't even know you've passed until you deliver, until you hit the finish line. It's not a matter of going, "Oh, the risky

stuff is done. Now let's just work." Just turn the crank and away we go. So I want to keep that assessment process going.

Identify risks. There's no reason to start with a blank piece of paper. One of the few things that SEI (www.sei.cmu.edu) did really well is some software risk management, and they've got a software risk hierarchy, so to speak. It's basically 130 questions that you can say apply to you or not. Do you have contractors? Nope? Okay, ignore the next 30 risks and move on. But it's a beginning point, not an ending point.

Steve McConnell in his book Rapid Development has a good list of typical risks in software projects. But let me say this and say it clearly: risk management must be idiosyncratic to your project. If you just say, "Here are standard risks. Which ones do we have?" you are not really playing the game. What those people who make those risk lists never know is your customer. What is your team makeup? What other pressures do you face? What I'm getting at is that you want to ask "What's different about my project from the project down the hall?" not "What's the same?"

Keep the tribes separate. What I meant here is that you want to keep the technical people talking about risk. Anybody can talk about any risk. You do want to talk to the customers about risk as well, but you don't want them at the table when you are trying to enumerate the risks. You'll get so much friction and heat. A customer will think X is a risk and a technical guy will counter, "That's not a risk. I'll tell you what a risk is here." What we do is we'll merge all tribes' lists together. Everybody gets to see the official list when we're done. But we do the identification by tribe.

For risk exposure, determine the probability of risk becoming a problem and the cost of effort if it does become a problem.

Now, some people roll their eyes here. A lot of books on risk management tell you to first of all look at past risk lists. Doing so, you see that 20 company or group projects have listed a certain risk and that four times it turned into a problem. So the probability is 0.2. But what if your database is empty?

Here's good news and this is going to sound weird. If you don't know what the probability is, guess. Just flat out guess. Put together people who understand that risk and have them guess the probability. The very fact you're talking about it shows you're 90% of the way home. It's the way I look at it. At least that's a start. No one says, "Oh, our project blew it because it was a 40% chance, not the 30% chance we estimated." Either it's going to hit you or it's not. That's the deal.

Sometimes I just do small, medium, and large on risk probabilities, and I try to buffer it. I say a low probability falls under 25%, medium under 50%, and large above 50% – large probability, large chance. And for cost, just give me tiers. I have people vote and the distribution of votes usually gives a pretty good view.

Determine which risks to manage. Is there a profitable tradeoff here? It's not always that managing risk is good and not managing risk is bad. It's what I get for my investment in managing that risk? Say somebody says, "There's a problem that's going to cost you 1,000 hours of work and it's got a probability of 50%."

Somebody else says, "I can make that risk go away at a cost of 700 hours. Give me 700 hours and I can make that zero probability." Do you take that? It's a 50% chance that you are going to spend 1,000 hours and a 50% chance you're going to spend nothing versus committing to spend 700 hours. There may be extenuating circumstances there but I think that's a bad bet. It's not a great bet.

Are there any actions I can take now that can lower the probability or the cost? Should I try to contain this risk by building some contingency? Think about this as a participant in my slush fund of time, effort, whatever. Now, the word for the day – I love this word – is "abulia". Abulia is the loss or impairment of the ability to act or make decisions. And abulia is rampant in many corporate cultures. And I am not saying the managers make the decisions. Sometimes the technical people, who know better than the manager about a technical high risk, must make these decisions.

And again it comes back to can I act now? If I don't act now, am I going to lose options? What's it going to cost me to act now? Can I wait another month for more information before I act? Do I move the boat out of the water today or do I wait till I see tomorrow's new hurricane map? Can I afford to wait or must I move because if I wait and wait and wait until the hurricane is on top of me, I lose my yacht?

Form action plans for direct risks. Some risks you can mitigate immediately but the mitigation is going to force you to change something: the project plan, the product definition, something. There's no such thing as free risk management that will let you vaporize a risk to zero for nothing. It doesn't happen.

Some risks you can't mitigate, so you try to build. Let me show you this one.

Risk 3: All functionality may not be ready to go at start of new fiscal year.

Mitigation: Build bridge code between old system and new, using subsystems 3 and 4 of old system until all is ready.

Probability: 50%

Tripwire: If all DDRs are not passed by 12/21/1999, we build bridge.

Cost: Al + two contractors = six work months = $170,000.

This happened a few years ago. I was consulting in New Jersey on a financial system, and this company was trying to go public. I walk in and the accounting people and the underwriter basically are saying, "If we can't get this financial system in this year, we're going to have to wait a whole year to go public. Inserting this financial system in the middle of the fiscal year causes so much extra work in terms of getting your financial in order so you can go public." So it's like if we can't get this in 2014, forget it. We're going to go to 2015. We're going to miss a day, then lose a year. That's their dilemma.

So they bring me in and ask, "What's a chance we're going to miss this? Is there anything we can do, Tim?"

We look at this and I talk to people, one of whom was this guy named Al. You've met Al. He's been at this company forever. He's a real tech guy. He knows the system, and I mean that with respect. He knows how it works. He sees its DNA – that kind of thing.

Al says to me, "You know, what they are doing is they are bringing in a package, mostly. And they're going to customize the package that will do all this extra accounting that they don't do. Our current system does a lot of this. It's not like this new package is doing everything 100% different."

And he starts talking about pieces of it. I don't want to go into detail, but he says, "What we could do is we could write some bridge code to work between the old and new systems using two pieces or two subsystems, and we can leave them as old and running. Therefore, we only have to work on the other piece and we could go live at the end of the year. And then behind the screen, we could clip it eventually so that the new package is completely working."

And so I go, "Ooh! Al, grab a couple of people. Go check this."

They come back and say, "We can do this. It's not big. We're going to have to write some bridge code between the old and the new but we could do this."

We estimate that there's about a 50% chance that we're not going to make this in a year. We come up with a tripwire: if all detailed design reviews do not pass by December 12, 1999, we build a bridge. If that tripwire triggers, if it's December 21 and all DDRs have not passed, we're going to build the bridge.

Al, the human spec, and two contractors, doing six person-months of work to build the bridge would have cost about $170,000. We propose this to the company and it's hilarious. They're like "50%?" and asking what happens if all but one of the DDRs has passed by December 21.

I'm from New York and I am old so I tell them, "One is greater than zero. We build a bridge."

"We couldn't wait?"

"Every time you wait, you increase the probability of the nightmare and the nightmare is the mitigation isn't ready."

Oops, the tripwire went off too late.

You know why I picked December 21. I don't know about the UK or wherever you are from. But here basically no work gets done unless it absolutely has to around Christmas and New Year's.

It got really funny. We're coming along and by early November all the DDRs passed. And so I said, "We're not building the bridge."

A couple of managers go, "It's only $170 grand, Tim."

I say, "You're sissies. We are going to make it. You're just going to throw $170 grand away. You guys are rich. I should raise my rates." – that kind of thing.

But they're like, "Wait a minute. Wait a minute."

The great thing was that they avoided the tripwire and never had to pay extra for mitigation. We made it, but it was fascinating to watch their reaction.

Keep the process going:

- There is no reason to believe that you can identify all risks in one go.
- Review risks for changes in likelihood and opportunities for new actions.
- When you retire a risk, all others move up the list.

So, good luck on your project. Just don't count on it when you are racing your boat.

Anyway, we're at break time. Do you have questions?

Participant: Does this mean don't bid on government contracts?

Tim: Well, it depends on what the bid process is. One of the biggest problems is fixed-cost bids with unknowns. Good luck, right? I mean I don't want to talk about any other country but in the United States one of the biggest problems with government contracts is that the people who win contracts are really good at writing contract proposals. I'll leave it at that.

Participant: Along a similar line, how do you budget for something when risk is unknown?

Tim: You budget just the way the weatherman budgets. You say this is where my boundary is. And as time passes, you will be narrowing in on a more precise measurement. If your organization wants precision more than it wants accuracy, you've got a problem. You've got to talk to them about the difference between precision and accuracy. You got to be able to say, "Look, I don't know exactly how much this will cost. I think it's $350,000. It could be as much as $450,000 or as little as $250,000." And there's no person on the planet that can show me they know any better. But I am not saying that this unknown is going to be there till the end of the project. We will be narrowing in as decisions are made, as truth reveals itself.

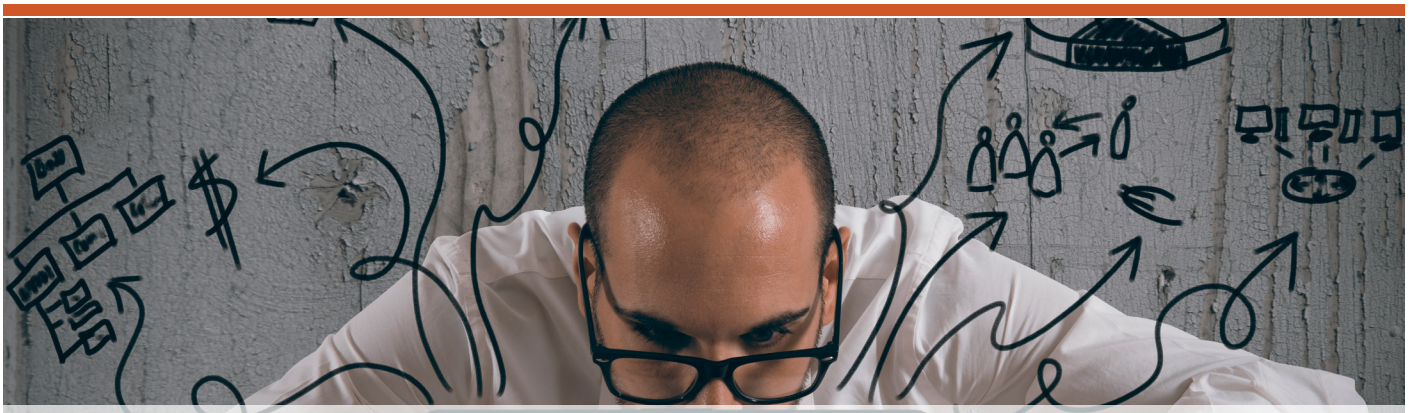
Participant: Do methodologies like PRINCE2 give enough background?

Tim: I want to say no, but the dilemma here is that the difference between okay risk management and excellent risk management has to do with the particulars of your project. And PRINCE2 never knows about your project. PRINCE2 or any generalized process is not the same as this anti-process, in which you consider what's different about your project, not how it resembles other projects. These processes usually look for what's the same between your project and other projects. You should ask yourselves what unique risks you are facing that could cause delay, loss, cost, or whatever you want to call it. Then you are doing an excellent risk management.

ABOUT THE SPEAKER

Tim Lister is a principal of the Atlantic Systems Guild, based in the New York office. He divides his time between consulting, teaching, and writing. He has over 30 years of professional software development experience. He holds an A.B. from Brown University, and is a member of the IEEE and the ACM.

WATCH THIS PRESENTATION ONLINE ON InfoQ



Solving the Gordian Knot of Chronic Overcommittment in Development Organizations

by *Rolf Häsänen and Morgan Ahlström*

The difference between successful people and very successful people is that very successful people say no to almost everything. – Warren Buffet

Why do we as humans and organizations have such a strong tendency to promise more than we can deliver? One simple explanation is that we want to please those around us. We want to say yes when someone asks for our help. Saying no could mean that people see us as rude or less capable. While being of service to others is a positive thing, when taken to excess it can be devastating - leading to stress, poor productivity, and congestion (both physically as well as work-wise).

Another less flattering and perhaps more dangerous reason for overcommitment is external pressure to promise what we know that we can't deliver. The forces behind these external pressures can include the following:

- Implicit or explicit threats to our job security if we don't accept work that is being pushed upon us.
- A view that the role of the IT organization is to support the business and that it cannot stand in the way of business development.
- An erroneous understanding of how work works, based on efficiency being a virtue in itself without understanding the damages of high levels of work in process.
- Pressure on sales organizations that overpromise on behalf of the delivery organization in order to close more deals. This behavior is often reinforced through dysfunctional incentive systems.

This behavior is often displayed in a fractal way throughout the organization: people overcommitting on a personal level; project managers committing their teams to fantasy project plans; sales selling features that the organization lacks the capacity to implement; and top management making promises that won't be delivered. This cultural aspect becomes self-reinforcing; when everyone else promises to deliver whatever is being asked for, it's easier for individuals to become part of this ever-growing mass than to be the troublemaker who says no. After all, by the time the problem becomes impossible to ignore, it might have already landed on someone else's table.

When the organization fails to analyze its mistakes, or makes too shallow an analysis, it's easy to come to the conclusion that the plan was flawless and the problem lay entirely with the delivery. And this is where most organizations focus their efforts to come

to terms with the problem: how to deliver according to corrupt plans.

Organizational behaviors driving overcommitment

The main behaviors for chronic overcommitment in organizations are:

1. Delivery capacity is unknown or hard to measure so it is easy to accept more and more work. It is common at several levels in the organization. Investment and schedule decisions are taken without knowing what delivery capacity exists. Projects are started and shoehorned into the unknowingly overloaded organization.

2. Projects and work are not prioritized. "We already have 234 projects ongoing and suddenly Marketing/Finance comes with a high-priority project that is a **must**. We do not know how it stacks up against the 234 other projects since most of them are also high-priority projects and a majority are **musts**".

3. There is no single point of entry for projects. Projects are started on many different levels and places in the organization.

4. A political cause is that there is no structured and holistic approach to stop workers from starting or killing projects for the benefit of more important projects. This results in zombie projects that run long past their expiration date.

All of these causes enforce each other in a vicious cycle that companies have a hard time recovering from. To examine this vicious cycle more closely, let's take a look at a hypothetical case study.

Case study: Claes Claesson at MegaRetailer AB

Claes Claesson is an IT manager at MegaRetailer AB, a company that sells thousands of products in its own retail shops and through third-party retailers as well as through their own popular Internet shop where some of the product families are available.

Claes attends a management meeting where the marketing director presents a two-pronged approach to increase sales and revenue for MegaRetailer. The first required change is to provide real-time inventory data for company retail shops so that salespeople can direct customers to another location if their own shop does not have the item the customer wants. The real-time inventory data must be made available both in the retail shops and the Internet shop (showing closest retail shop that carries the item). It needs to be implemented in time for the Christmas shopping season. The business case for this change is good, there is no denying that. A projected 12% increase in sales for the Christmas period would provide a substantial revenue increase and the implementation would reduce inventory costs.

The second required change is to provide the Internet sales channel on mobile and tablet formats and enhance it with state-of-the-art visual product recognition. Customers can take a picture of an item they want and the app will identify the product and recommend MegaRetailer's own brands or similar products. The app will also identify the closest location that carries the item and allow a purchase from the online store.

The overall plan is to implement these changes over the coming 12-24 months in multiple projects. Sourcing and Logistics have already started to look at scanning systems to provide product-matching data when new products are brought into the MegaRetailer line.

There is no debate that the end result of these changes will be good for MegaRetailer, but Claes already has hundreds of projects in various states of progress and isn't sure whether there are enough people to staff these additional projects. There is no understanding or acceptance in the meeting that the IT department teams are already busy. In addition, the marketing director somewhat heavy-handedly reminds Claes that business drives the company forward and that IT is to be a supporting function and not a roadblock. Here Claes's boss, the CIO, steps in and says, "That's right. IT supports the business and Claes here is our go-to guy for solving difficult problems and delivering the most challenging projects. But in all fairness Claes and his people should be given some time to analyze how we can deliver these work packages in the best way."

The supply manager cannot refrain from sniping. "Well, maybe Claes could take a look at another challenge, our delivery control system, since it is already more than eight months delayed." Luckily, the meeting ends before that discussion escalates any further.

The decision is made. Claes and his organization are to analyze the marketing requirements and come up with a plan for development and implementation. His boss tells him, "Claes, I know that we are stretched thin but these two items are crucial and have the backing of the CEO and the board. Look at hiring consultants if you need but no hiring of employees beyond the current plan. We do not want to add to our cost mass. I have all faith in you and your people."

Understanding the dynamics at play in the case study

In this section, we will work to understand the dynamics at play for Claes and his team. We will map the situation using causal-loop diagrams. If you are not familiar with causal-loop diagrams, you can read through the short how-to section.

Claes is starting projects to deliver what the marketing director requested. One of the reasons it is so hard to say no is that the perceived benefits are seen as great and no one at the table understands the consequences of starting one or more new projects. Even if Claes, or anyone else on the team, has an instinctive feeling that they are already overloaded, it is hard to debate or discuss it without facts and numbers for consequences. The harder it is to understand the consequences, the harder it is to say no to new work, and we cannot understand the consequences without understanding our current capacity. You can see this mapped in figure 2 *(next page)*.

If you continue to add projects in an uncontrolled manner for long enough, the IT organization will become overcommitted. It may take some time but it will happen.

When existing projects get delayed, trust between IT and the business deteriorates and business stakeholders will look to take any opportunity to push their needs on the agenda. This can show up in the form of requirements forced into projects that have already gotten clearance to start or, perhaps, as departments doing skunkworks projects without IT participation, causing additional work for the IT department in the form of integration, maintenance, and security issues.

All of this can be summarized as pressure to accept more work. Eventually, this will become a reinforcing loop that keeps heaping pressure on the team to accept additional work, as figure 3 shows *(next page)*.

One of the actions Claes and his boss can take is to kill projects that are less important to create space for projects that are more important. This would lessen the strain on the IT organization, provide a more even flow of deliveries from the projects, and allow Claes to come out on top of the situation. Let's add that in figure 4 *(page 38)*.

Unfortunately other dynamics come into play when Claes wants to kill some existing projects. When the organization is overcommitted, it usually has hundreds of projects with a multitude of stakeholders, making it very difficult to kill a project. Project champions and stakeholders will want to keep their projects running in order to get the anticipated benefits. They will point to other projects and say, "Find another project to kill because my project is important."

Basically, Claes is trying to sell the message "Your baby is ugly and needs to be closed so someone else's baby can be started." He will not find many sympathetic listeners. This gives us the "Do not kill my project" reinforcing loop in figure 5 *(page 38)*. The more overcommitted our organization is, the harder it is to pinpoint the right projects to kill in order to bring us back to a stable execution mode.

Unfortunately, killing a running project will further tear the trust between IT and business and increase the pressure on IT to accept new work. So once in this catch-22 of overcommitment, it is hard for an organization to get out of the hole.

In summary, the IT organization can be viewed as a bathtub. When this bathtub overflows, it creates extra costs and problems and delays delivery of existing projects. The basic rule is never add more work than flows out of the organization once you have reached your capacity.

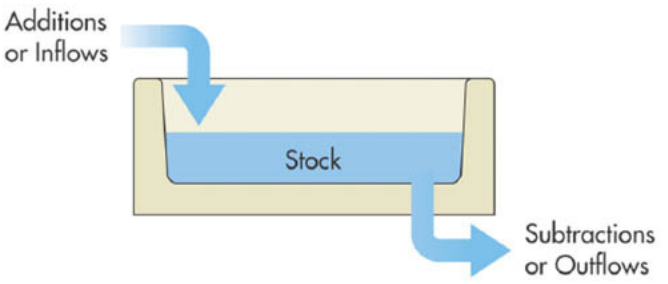


Figure 6 - IT department as a bathtub.

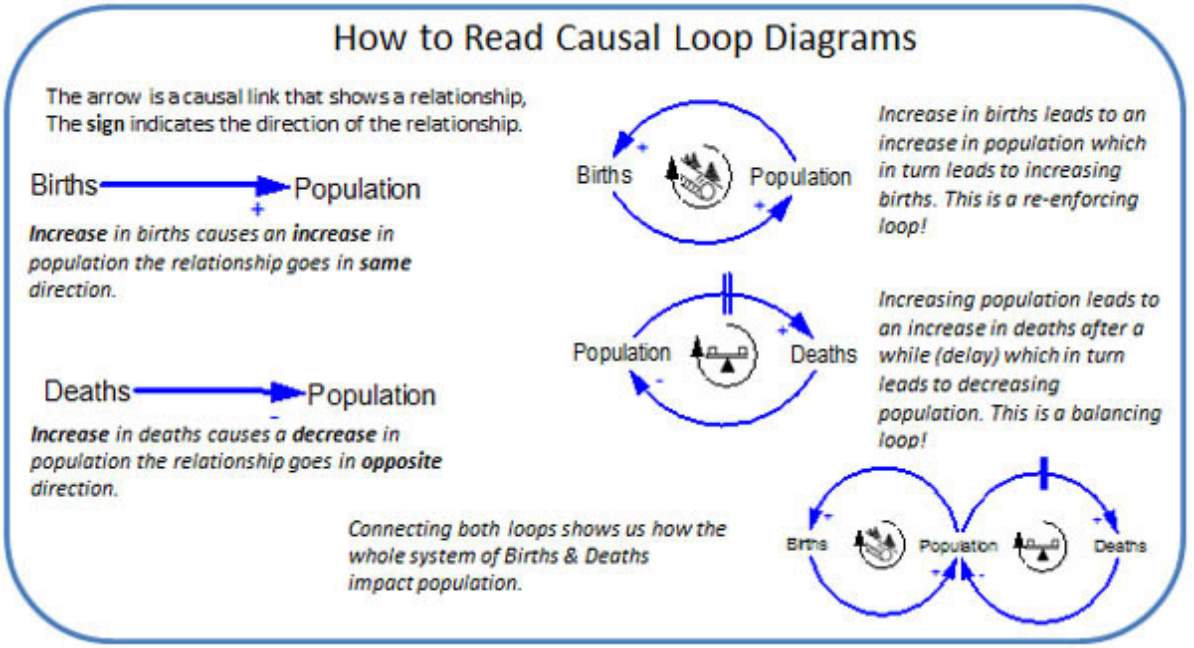


Figure 1 - How to read causal-loop diagrams.

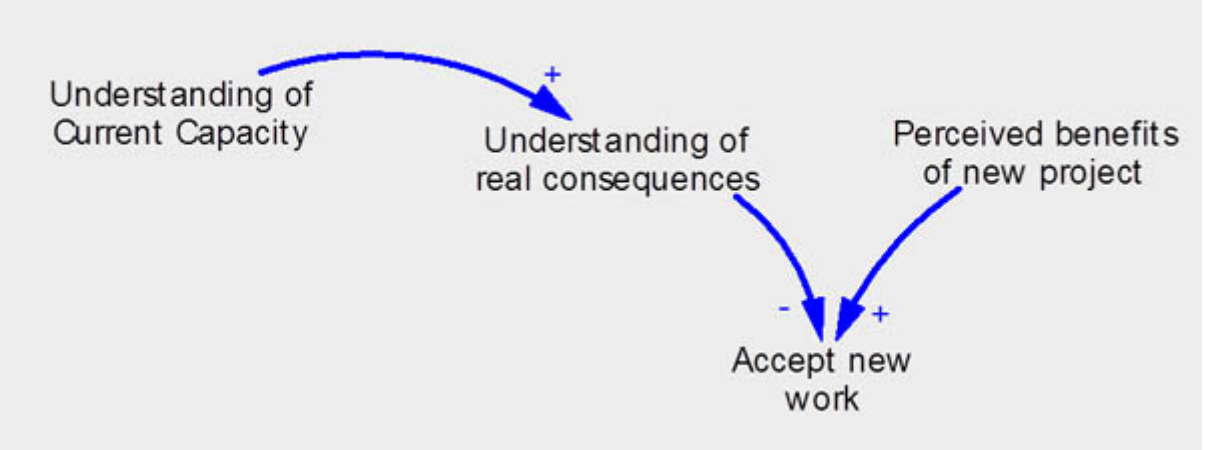


Figure 2 - Acceptance of new work.

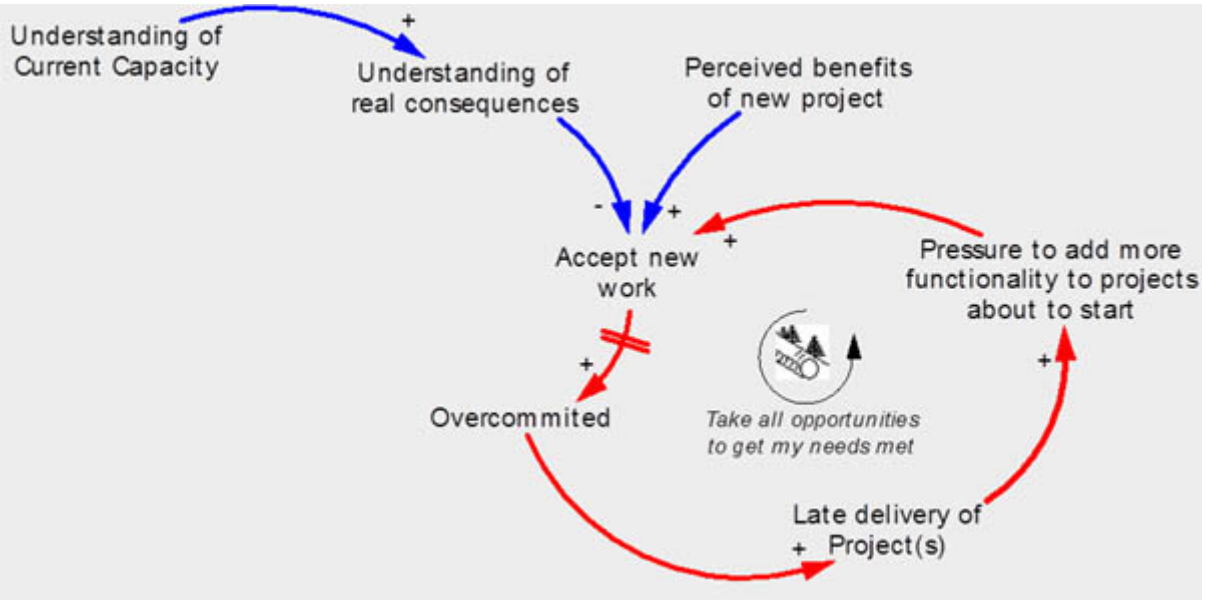


Figure 3 - Increasing pressure to accept more work keeps the organization in a perpetual state of overcommitment.

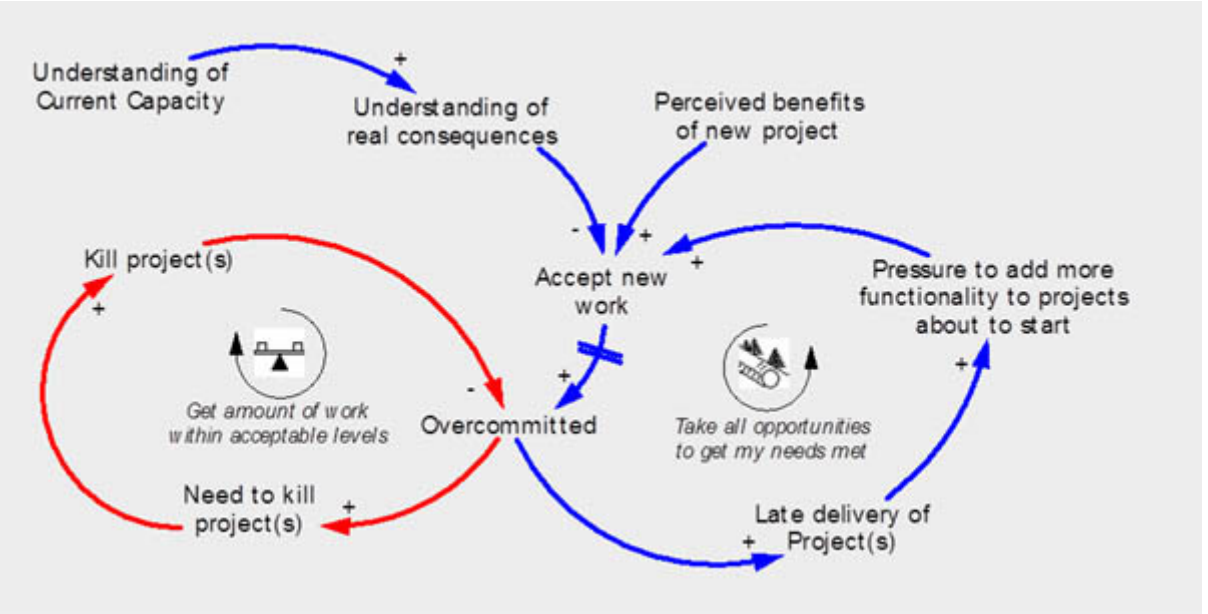


Figure 4 - Killing projects to lessen overcommitment problems.

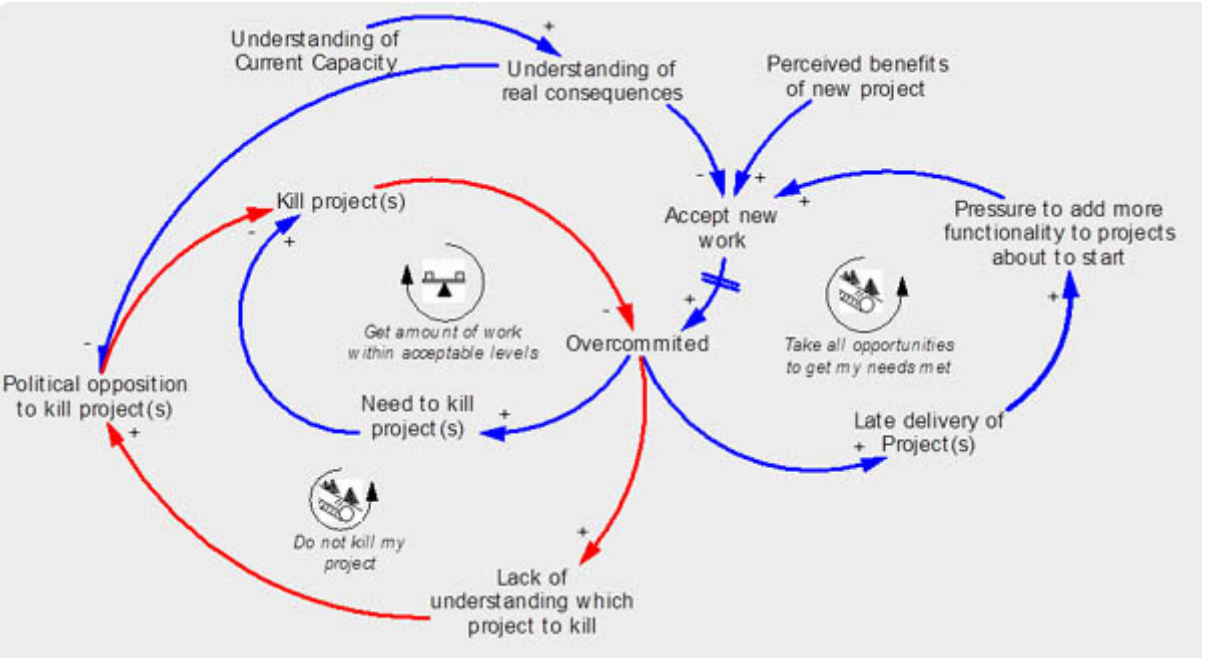


Figure 5 - "Do not kill my project" reinforcing loop.

Resolving the Gordian Knot between business and IT

A Gordian knot represents a difficult, intractable, and often insolvable problem, which is exactly what it feels like in an organization with chronic overcommitment. But there is hope. Let's see what Claes and his boss could do to improve things.

Claes could try to solve his resource issues by using external suppliers and various workarounds to provide functionality faster, but none of these methods addresses the main reasons for overcommitment. While workarounds may help, they will only serve to keep the organization in the catch-22 they find themselves in. Even more worrisome is that over the long run, performance will deteriorate since the IT department's capacity gets worse and worse over time.

Instead, Claes should apply the following four steps:

Learn to deliver on a small scale.

Isolate some value stream or function where it's possible for a team to deliver value without dependencies on other parts of the organization. Have the team deliver with quality – accept no shortcuts or deliveries where the quality is unknown. Feed the learning of how to organize around a value stream back to the organization and then slowly begin to scale up, being careful to avoid unnecessary dependencies between the value streams. Use agile and lean practices and principles to improve your delivery capacity.

Have a clear strategy and live by it.

You cannot kill project #265 without a clear strategy. By articulating the business vision and priorities in a concrete manner, with clear, real examples, people can make the right decisions. Remember that everyone is prone to overcommitment and that the job of management is to make sure that the delivery engine is not overloaded. Create hard limits for the system and enforce escalation procedures like requiring CEO-level acceptance before exceeding those limits for work in progress.

Companies like Volvo and Harley-Davidson have learned this lesson and created a strategy to deal with it. At Volvo, the strategy was codified in the quote "666 is the highway to Hell," meaning that they should never do three major projects in parallel (major projects are called category 6 projects at Volvo). Similarly, Harley-Davidson made huge progress by understanding that they could not do more than one big project per year.

Change your habits around problems.

Normally when a problem occurs, management rushes in to help solve that problem but often makes things worse by enforcing more control, status reporting, and other work that is disconnected from reality.

By doing what should be the work of the front-line people, managers lose their strongest card since their focus shrinks. Once you have ensured that your people have the needed expertise and any other resources necessary to solve the problem, make sure you have a structure through which to share information about the issue.

Finally, think about the problem's potential side effects on the organization, both today and in the future, and try to answer the following questions. How could this problem have been detected earlier? Is there anything in the current system and strategy that creates this problem? How do we share new knowledge gained from this problem? Innovation requires space, so how can we create this space for our teams so that this problem does not impact us again?

Create common knowledge.

If you are responsible for the delivery engine in your organization, you will need to create a common understanding between all parties so that everyone understands the strategies and rules of the game. It is important to create this common ground so that everyone understands that the decisions and behavior are optimal for the organization and not done for local silo optimization.

In conclusion

We have hopefully shown that the problem of chronic overcommitment is the result of many patterns and people interacting, and that any solution to this problem must be applied across several different layers. As a rule of thumb, though, until you have a clear understanding of your organization's capacity, don't start any new work until you've closed something old. Our business is not to start as much as possible, it's to finish as much as possible.

I'm as proud of what we don't do as I am of what we do. – Steve Jobs

ABOUT THE AUTHORS

Rolf Häsänen is the founder of Value at Work and a systems thinker who is passionate about improvement and innovation. Combining principles and methods of systems thinking, lean product development, design thinking, and the agile world, Rolf helps people solve complex business problems and improve their organizations. Currently, Rolf is working with challenges involved with large-scale product development and how to capture and share knowledge across organizational boundaries and unlock the full potential of team learning.

Morgan Ahlström combines a broad experience from several different roles in IT organizations in many different industries such as banking, media/television, automotive, telecom, travel, fashion, and facilities management. This has given him a great ability to help his clients find new solutions to their challenges. For more than 10 years, he has continuously nurtured a strong interest in agile software development and change management in organizations and uses these skills to help his clients develop more effective and efficient organizations. Today, Morgan is mainly working as a lean/agile coach, helping individuals, teams, and organizations implement agile processes.

READ THIS ARTICLE ONLINE ON InfoQ

Management of Agile Projects

Presentation summary by *Shane Hastie*

At Agile Cambridge 2012, Tony Willoughby discussed the project manager's role in an agile team, focusing on resourcing, cost control, high-level scope management, risk management, and wider communication with business stakeholders.

His talk started with an introduction of himself and his topic. An important constraint that Tony mentioned is the domain in which he presents his ideas: a commercial organisation that builds software products for sale rather than a company building software for internal use. He also stated that the agile approach the company used was Scrum.

He stated that his reason for preparing the talk is the lack of literature on the role of project management in Scrum and in agile in general. He challenged the idea that the development team can perform project-management activities and practices and negate the need for the role of project manager. He said that the focus of his talk is the role of the project manager in an agile project and how one can add value.

He maintains that agile is now a proven approach. It:

- Results in better software (fewer defects)
- Brings faster results
- Meets requirements better
- Produces happier teams
- Produces happier customers

In his organisation, the adoption of agile was driven from the bottom up – developers and technical people who said, "We're fed up with these waterfall-type projects where we have to try and do everything up front and we can't. Agile is a better way. Let's try it."

Initially, management resisted.

> *There was quite a lot of resistance from management at the time because in the commercial world you're mostly driven by what your customers want and customers weren't at that time ready for it. So there was quite a lot of work to be done to get…to a way we could adopt agile and use it.*
>
> *Now, it's more the other way, actually. Most customers at least have heard of agile and I think it's a good thing. They don't necessarily know how they're going to use it and what effect it will have on them but most of them are brought into it in some form or other. So it's less of a selling job to be done now.*

He summed up the reasons that agile is better.

> *First of all, agile encourages better development techniques. We don't have to be agile to use things like continuous integration, test-driven development, pair programs, and those kinds of things. We don't have to be agile to do those but agile certainly*

emphasizes them, and they go hand in hand with it and they're certainly good techniques.

So most projects are subject to a lot of uncertainty. Now, there's a lot of debate about where this uncertainty comes from. Some people say, "Well, software requirements keep changing and therefore we have to accommodate that." I think it's not quite as straightforward as that. I don't think that if you speak to most users during the course of a six-month project…, they will say the requirements have changed.

What has changed is the understanding between you as a development team and your customers as a set of business users who want something out of the system but who can't explain it to you. And, of course, we all know that the waterfall method, which expects everyone to understand at the start what all the requirements are, how the system should behave, and so on. We all know now that that is virtually impossible, and agile kind of gives you an ability to have that journey of discovery of what the requirements really are.

And it's partly down to a question of language, because we all use the same words but I think the business people will have a different understanding what those words mean from the developers. So we can agree on the set of words and the specification but our understanding of that specification can be quite different in some very crucial ways, which has a big effect on the outcome.

Agile gives us a way of getting around that problem by having developers say, "I'll just develop a little bit and show you." And together we'll go on that journey to understanding. And to my mind, I think that's really why it works better.

Tony said that not all projects should be agile and not all activities in projects should be agile. At times, the customer will refuse to accept an agile approach, and we may not be able to persuade them otherwise. Some naturally sequential activities take time and don't need adaptation, such as selection of underlying architecture, infrastructure setup, and some aspects of UX/creative design.

In many cases, those decisions are made for you by whom you've got in your team or what the customer expects you to do. But in other cases, you have a choice about whether to build a piece of software or to buy it as a package. And those are the difficult decisions which will have a far-reaching effect on the rest of the projects and are quite difficult to undo so you need to think about those in advance…. Buying hardware and commissioning data centres and all those kinds of things are pretty conventional types of waterfall project, especially when you've got a long lead time for certain types of hardware.

He summarised the key roles in a Scrum team.

- Business/product owner – the requirements
- Scrum master – organisation and tracking
- Developers – coding and unit-testing
- QA staff – testing
- This team should be largely self-regulating

The project manager is conspicuously missing.

Tony listed project-management responsibilities, incorporating many tasks not recognised in Scrum.

- Resourcing – numbers and correct levels of skill
- General administration – arranging meetings, travel, project-management tool setup
- High-level project planning and tracking – meeting key customer milestones
- Tracking expenditure and customer billing
- Commercial/legal. Do we have contract cover?
- Tracking of risks, issues and dependencies, and escalation where required
- Quality-control of documentation
- Communication – with customer, internally and with third parties
- Especially outside the immediate agile team
- Reporting – internal and external
- None of these is covered by agile processes
- Some customers claim to be agile but not many really are
- The project manager provides the interface between our processes and theirs

He explained some of these points.

So we have our self-regulating team, most likely in an organization that is not agile and that is much

more used to conventional project and management roles. The project manager has to bridge the gap between the team and organization.

Most projects have third parties who are delivering key parts of the solution. Now, managing them is a pain. It's often one of the key things that can cause a project to fail, and somebody has to manage them and you usually need a project manager to do that. But on top of that, there are all sorts of things in the commercial software environment that have to be done. Have we got the resources or the team that we need? Have we got the right mix of skills? What about team holidays and that kind of thing? Do we have contract cover and commercial cover for this? Are we being paid? Our job processes do not normally cover these. And, as I say, it's made more difficult by some customers who claim to be agile but do not actually understand what it means.

About the Scrum master role, Tony explained that in their organisation they have two roles that provide leadership and guidance: the technical manager/ architect and the project manager, who share responsibilities.

Development projects usually have both a project manager (PM) and a technical manager/technical architect (TM).

- The TM is usually dedicated full-time to one project.
- The PM is often running several projects.
- Who acts as the Scrum master?
    - Usually, it makes more sense for this to be the TM, because to be an effective Scrum master:
        - You need to attend every stand-up (not possible for a part-time PM).
        - You should understand all the technical details of the work.
    - Scrum master is a natural fit with the TM, if the TM is not too controlling.
        - But the PM may need to assist, especially with keeping storyboards up to date, people management, and customer communication.

Tony elaborated.

We tried various things, and the solution for us generally was to say the technical managers, the technical architect ends up as the Scrum master, for several reasons. Firstly, that person is fully in the team and so is there all the time. Sometimes the project manager is part-time. Secondly, I think to be an effective Scrum master, it helps to be fully in the technology and so to understand exactly what everyone in the team is doing, and the technical manager usually has a better understanding than the project manager.

But the project manager certainly doesn't need to assist in this process. Certainly, if the technical manager needs help like making sure the team keeps their stories up to date, updating the backlog and the burn-down charts and so on, then the project manager can help with all that.

Expanding on project-manager responsibilities, he stated that requirements definition is a project-manager responsibility.

First of all, there are business requirements. What does the business actually want out of this software, out of this system? What are their constraints? They may require that the system must go live by Christmas. Why must it go live by Christmas? What are the business drivers behind that? This must be captured and made clear because the reasons are important. Maybe there's a big marketing drive happening and all the marketing materials have already been ordered. So there again we have another clear date.

Another business requirement or constraint might be that you must work with this specific partner. Why? Well, we have all sorts of reasons why that partner needs to be working even though they may not be ideal. Or we need to use this payment gateway. Whatever.

Then we come to the user requirements. Now, the requirements of the users of the system are often uncovered by a combination of a technical team, doing stories, and a UX (user experience) team,

the creative designers. And sometimes the UX team can really muddy the waters, I'm afraid. They think they're doing a good job, but there is a tendency on the UX side of things to inflate the scope beyond what was agreed to, and that needs to be managed. Sometimes it's good to inflate the scope because they discover things that were not yet discovered, but sometimes it gets too focused on something, and that needs to be managed because in the end we have to deliver working software. We don't want to be driven down sidetracks.

So they need to be managed quite carefully. And indeed we need to do this as one team, really. So when the UX team drives out software requirements from the customer in their own particular ways, we need to have technical representation there to make sure that it's all actually deliverable and makes sense from another point of view.

And then, of course, we've got the usual software requirements. They are usually captured in user stories and so on. And sometimes the team will need help in formulating their stories and making sure they're the right size and so on.

Unless you're going to do everything in a very software-as-a-service kind of way or platform-as-a-service, or maybe more, then you're probably going to need to define the requirements for and the constraints of the infrastructure up front and then put in place a plan to manage those.

The project manager also supports the product owner.

- The role of product owner is crucial.
    - Define the business requirements
    - Consult with other business stakeholders
    - Prioritise the product and sprint backlogs
    - Make trade-offs when things deviate from the plan
- Not all product owners are equally effective.
    - The project manager can help the product owner to stay on track.
    - In extreme cases, the project manager may become the product owner.

Another role I think that's often forgotten is the role of the product owner. The product owner is actually quite a lonely job, or can be a lonely job in an agile team. The buck really almost ends up stopping with the product owner in terms of where they are with delivering the business requirements. Obviously, the team behind them will work to do everything. But they have to decide the prioritization. They can get it wrong. They can have all sorts of pressures on them from the business.

So I think the project manager can help the product owner to explain why things are what they are to the rest of the business, and help to explain to the team why the product owner has these particular constraints and so on, basically supporting them, helping them with this prioritization, explaining the role of these trade-offs, and so on. I have worked in projects where the product owner was very ineffective. And in that case, the project manager can sometimes take on this part of that role. I don't know if it's ideal.

He said that scope management should be easier on an agile project.

We don't fix the scope. We decide it as we go along. Some customers take that as a sort of carte blanche to then change the scope whenever they feel like it. It's not a good idea because change still costs. So if there's going to be a lot of that, the project manager can step in there and explain the consequences of some of these changes, say, "We can accommodate that change but do you realize what else they're going to have to change in the project or accommodate that?" And if it's a major change, I still think it's probably best to go through some conventional sort of change-request process just so the customer knows that there has been a major change and there are consequences of that in terms of what they might see.

Tony spoke on project planning and milestones.

So, we have our release planning, sprint planning and so on. That's pretty detailed to your progress. It's not really what the business is looking for. The

business probably doesn't want to know that at the end of this sprint you're going to get a certain increment in features. They're going to want to know when they will have to prove the UX design. When will they have to mobilize people to do the UAT? When are we going to go live? (They want to know) those kinds of big dates.

These are all conventional types of dates of the software project in a conventional approach, but the business still expects to see them and so project management can help you with that. Sprints and a two or three-week… is interesting to the team because that's the case to which they work but it's not so interesting to business sponsors.

The project manager is also responsible for tracking and managing issues, risks, and dependencies.

Some people make a big thing about the difference between an issue and a risk. Okay, there is a difference. But actually, they're more similar than they are different. They can all derail a project. They're all time-critical. So someone needs to manage them. Someone has to be assigned as the one who is responsible for them. They should need to be reviewed regularly.

So the project manager can be surfacing these things to the rest of the team and to the business and explaining what these issues, these risks are and how they're going to be managed, and also dependencies on third parties – very important. But I would say that you have to be aware of what I call issue fatigue here, which is that some project managers decide that the best way to communicate is to document every possible risk and issue – always dozens, if not hundreds – and present them all to the customer every week. Very quickly, the customer will get issue fatigue and say, "Well, I can't handle all those if this needs to be going. I'll just ignore them." So the key is to prioritize them and just pretend the two or three that are key need immediate attention.

Then there is the project reporting and tracking.

Project reporting is another boring job that the project manager has to do, usually writing these weekly reports. The good thing about working in our job team is that usually the reports you provide for a customer are pretty similar to the reports you would provide to a commercial software company, that you provide to your own people because there should really not be many, if any, secrets. You can be open and share. So if you have problem with resourcing, as it is a common problem in commercial software environment, you say, "Yes, we have a problem with the resourcing. We are trying to work around that. And when we get this actual person on board, then we'll start to catch up." Make those things clear rather than pretend there isn't a problem.

Another area where the project manager is crucial is when a customer wants a fixed price and scope but still wants to run the project along agile lines.

The big bugbear is how do we handle a fixed price project? Quite often, you know – less often now but it still happens – the customer asks us to run a fixed-price, fixed-scope, and sometimes fixed-time project but expects it to be agile. There's a pretty big mismatch between those two fundamentally contradictory approaches. So we found some ways to resolve this. Ideally, first of all, persuade the customer that they're wrongheaded and they need to be more agile and less prescriptive.

We've tried that and sometimes the people you're working with say, "Yes, we understand all that."

And then the procurement department come in and say, "Sign here in blood. We're not having any of that." So you can't always get around that alternative. You could pretend and they'll pretend to be agile. Just go back to the old method, which again is not ideal.

What we found works sometimes is this sort of hybrid agile approach. We know we expect to deliver X during this period, but we're not going to admit to X because that's far too risky. So we'll sign up to 60% of X. We'll give you these 10 features. We'll guarantee to give you these six but we expect to have enough to deliver all 10. So we'll agree to fixed scope and fixed price. And if we deliver the six features, we'll get paid. But we know that actually in order to keep the customer on board as a satisfied customer, we actually should endeavour to deliver all 10. We're not going to rest on our laurels when we get to six.

That's worked pretty well because often while we're delivering those first six features, we find that other priorities creep in, or the customer realizes that maybe some other things are more important to them and they come to that negotiation and agree. And then they can see the benefit of working in agile

way because we actually can accommodate that change while we're still working within a fixed-price type of approach. So that works. That's worked quite well.

Tony presented some ideas on tools and techniques the project manager can use.

We share information through the project wiki, those kinds of things, with the customers and the stakeholders and try to show all of these issues and risks on a continuous basis if they want to see them. We generally have a couple of issues logs, with one for what the project team is working on at the moment. That would be perhaps a set of tasks in JIRA or some similar tool and showing the development logs, as I called it there, showing all the stories, the tasks, the bugs, etc., arranged on a sprint basis. So, for the next two weeks we're going to be addressing these ones.

And then the project manager's log, which is the higher level, maintains the kind of things that the business people really want to see, which are the risks, the issues, the milestones, dependencies, any CLs, and so on…. We've used JIRA and looked at Trello as well as a lightweight way of looking at some of those. But there are other tools I'm sure. There are even better tools available.

He summarised his talk.

I hope I've managed to convince you that the project managers are not redundant. In an agile project, they should be less involved with the day-to-day task management, because the agile team should hopefully run itself with a bit of guidance, but more concerned really with this interface between the team and the business. Remember, the team – we're hoping – is agile, doing things in agile, whereas the business is not generally agile. The business really needs to know milestones, what's going to be delivered, how much it's going to cost, all those kind of things. So the role of the project manager is to try to span those two different paradigms and make sure they can work together.

ABOUT THE SPEAKER

Tony Willoughby has been working in IT for over 20 years in a variety of roles, including requirements analyst, technical architect, project and program manager, and account director. He is employed by KIT digital, an international company specializing in digital media, where he has delivered a number of large-scale video-on-demand projects in the UK and the Middle East.

WATCH THIS PRESENTATION ONLINE ON InfoQ