Programmers are from Mars, Customers are from Venus: A practical guide for customers on XP Projects

Angela Martin
Computer Science
Victoria University of Wellington
New Zealand
+64 4 463 5666
angela@mcs.vuw.ac.nz

James Noble
Computer Science
Victoria University of Wellington
New Zealand
+64 4 463 6736
kjx@mcs.vuw.ac.nz

Robert Biddle
Human-Oriented Technology Lab
Carleton University
Canada
+1 613 520 2600 x6317
robert_biddle@carleton.ca

ABSTRACT
Extreme Programming and other Agile methods have a dedicated *customer role* that acts as the interface between development teams and their clients, sponsors, and end-users. The customer is critical to agile projects, but there is little research, experience, or advice about effective practices required to fill that role. We present a set of patterns describing the key roles on a customer team, and the practices that enable customers to fill those roles. By adopting these roles and practices, customers and development teams can increase the velocity and reliability of their projects, and ensure all participants in a project, not just the developers, can work at a sustainable pace.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management

General Terms
Management, Design, Economics, Human Factors.

Keywords
Agile Methods, XP, Extreme Programming, Customer.

1. INTRODUCTION
This paper introduces patterns for roles and practices that can increase the effectiveness of the customer on an Extreme Programming (XP) project. Customers have one of the most complex and difficult roles on a project, yet XP includes very few practices that support the customer in their role — the aim of this paper is to change that. Over the last three years, we have investigated many projects around the world to identify how customers succeed in this complex and difficult task — discovering not what people think should have happened, but what really happened and what actually worked.

This paper distils this research, grounded in practical experience, into a number of patterns:

- Covering **key roles** required on a customer team, both what they are and why they matter.
- Covering **practices** that enable customers to sustainably drive XP projects to successful completion – similar to "XP practices" but for *customers*.

We use a number of quotes from the interviews we have conducted during the last three years to illustrate our findings in this paper; names have been avoided or invented to preserve anonymity.

2. CUSTOMER TEAM ROLES
The XP customer is typically a team of people; as there is too much work for one person:

> *"We probably needed about three of me … it's been my life for about a year … look at these grey hairs"*
> *— Customer, KiwiCorp*

> *"I've always worked at least 70 [or] 80 [hrs a week] I don't even mind it, its like what I do"*
> *— Customer, RavenCorp*

Lest these quotes be taken out of context, we should introduce the important finding that practitioners are excited about the changes introduced by XP to the dynamic between the customers and programmers:

> *"Overall – I love this approach to development and I'd certainly like to use it again in any future projects I am involved in"*
> *— Customer, KiwiCorp*

In the successful project teams that we have studied [8, 9], each one had a customer team, and although each of the teams had an identified "customer", there were nine key roles being performed on these successful customer teams:

- Geek Interpreter
- Technical Liaison
- Political Advisor
- Acceptance Tester
- User Interface Designer
- Technical Writer
- Diplomat

- Super-Secretary
- Negotiator

Each of these roles directly supports the success of the “customer role” in some aspect, but which roles are needed will depend on the nature of the project. More than one role can be played by one person, and more than one person might combine to play a role. How these roles are established is also a matter for context: someone in the “customer role” may informally create the roles to provide the support they need, or the roles may be created as part of a more formal management process.

2.1 Pattern: Geek Interpreter

Problem

“H3Y D00DZ, L3TZ C0D3 UP SUM ST0R1Z”
— Geek, 2005

Programmers and customers do not always speak quite the same language, even when the both speak English (or French, or …). It is easy for a customer to become baffled by a programmer; did they actually answer my question, what did the answer actually mean to me, will it fix my problem?

Forces

- Programmers say “it will take fifteen days as we will need to introduce [xxx]”
- Customers think “but I just want to do this simple thing, do I really need [xxx]”
- Programmers explain why [xxx] is needed, but Customer does not fully understand, but after a few attempts is unable to find a way to communicate further with the programmer, and is worried that they have been misunderstood.

Solution

We have found that customers who are not themselves (ex-) programmers often lack expertise in programmer jargon, and need a geek interpreter, a person who helps the customer understand and talk to the programmers.

Forces Resolved

The Geek Interpreter generally does not talk to these programmers directly but instead provides a sounding board and coaches the customer in G33k 5p34k (Geek Speak). The Geek Interpreter role is never an official team role / position on the team. Often the Geek Interpreter has the official role of a Business Analyst, Tester or project manager and is a recent ex-programmer. However, the Geek Interpreter can also be a programmer (on either the same or different project) whom the customer trusts.

It is also interesting to notice that in all cases we have studied, the customer has been very careful with the use of the Geek Interpreter role and never plays the Geek Interpreter against another programmer directly. If a customer were to play the Geek Interpreter off against the programmers then the customer could (a) damage their relationship with the programmers who may feel that the customer no longer trusts them and/or (b) the customer could damage their relationship with the Geek Interpreter, who might no longer feel comfortable providing assistance to the customer. Additionally the Geek Interpreter’s relationship with the programmers could be damaged, and this could be significant depending on their role on, or relationship with, the programmer team.

2.2 Pattern: Technical Liaison

Problem

Most projects don’t exist in isolation; they have to deal with existing organisation technical infrastructures. Customers who attempt to deal with the entire technical liaison quickly become overloaded. Programmers will tend to focus on getting their stories completed, and will also become overloaded if attempting to undertake this task. Additionally although programmers may have the technical expertise, they do not always have the same perspective as technical infrastructure specialists. For example, technical support people can find it difficult to deal with programmers, who sometimes don’t understand why they cannot rebuild the organisation’s entire database so that the simplest thing might possibly work.

Forces

- We need someone who is able to interact with the existing technical infrastructure and the organisation/departments that are in place to support it.

Solution

We have found that a person on the team picks up the Technical Liaison role, removing this communication “overhead” from both the programmers and the rest of the customer team.

Forces Resolved

This role can sometimes be a formal role on the team, particularly in organisations with large IT departments or in organisations where their operational facilities have been outsourced. In situations where the IT departments are smaller and operational facilities are in-house, then this may be a part-time role that is picked up by a member of the project team.

We have also found that this person may be able to support the programmers find answers to the legacy system queries, and at times even code, although typically this person is post-technical but with a strong technical knowledge base.

It is important to ensure that this person remains a part of the team and does not go native (get captured by the larger IT organisation). They need to obtain support from the team because they will be facing a lot of pressure externally and can easily become battered and bruised.

2.3 Pattern: Political Advisor

Problem

Every organisation of more than two people has politics. Customers are – by their nature – involved intimately and continuously in a development project. This means they **cannot** – by themselves – keep up with organisation’s politics and power structure.

Forces

- Customers need help to identify the players and the rules:
    - Who needs to say “yes!”?

    - Who needs to say "no!"?
    - Which rules to follow?
    - Which rules to break?
- Note, this is different from XP's Goal Owner & Gold Donor, as organisations are much more complex than this, with many stakeholders at different levels.
- We may need multiple political advisors, depending on the size and complexity of the organisation and its political by-plays.

Solution

We have found that a wise customer recruits one or more political advisors whom they can (hopefully) trust.

Forces Resolved

The political advisor will help the customer work out ***who the political players actually are.*** There are always both official and the unofficial players, and both need to be identified and understood. ***A strategy is then needed for how to work with them,*** both in the short term and long term. A wise customer is aware that they need to not only see this project succeed but will also set up their political network for the next project to succeed as well.

It is important to ensure that one is constantly assessing the advice and guidance of your political advisors, that one is really plugged into all of the political dimensions necessary. For example, in one project, it became painfully aware to the project team that one political player had been overlooked, in this case, operations. The project was delayed and portrayed within the larger IT organisation as a failure due to the delay, despite the fact that the project team delivered working software that added business value, and all much quicker than originally expected; all that was remembered was that the team was a month late. The customer had missed a key political player within the organisation and the long-term perceived success of the project was compromised.

2.4 Pattern: Acceptance Tester

Problem

Classic, or first edition, XP [2] had programmers seconded to the customer to help the customer test the application that can leave programmers politically conflicted and customers without the specific skill set to thoroughly test the application.

So, how do you ensure that there is a focus on quality, from the customer's perspective rather than the programmers?

Forces

- We want to ensure the customer's perspective is represented when the application is tested: a focus on how the customer and/or end-users will use the software to achieve their business goals, rather than how they get a story signed-off.
- We want to ensure that the testing is structured and prioritised so that the most important tests (from the customers perspective) are undertaken and not simply the easiest tests. A constant cost/benefit analysis of when to test, always considering the impact to the quality of the application versus the ability to add new functionality.

Solution

We need to ensure that someone on the customer team has testing experience and is prepared to take on the role of acceptance tester on the project.

Forces Resolved

We found acceptance testers as assistants to customers on almost all large standard contemporary XP projects. Real testers understand testing, are good at it, and take the customer's side!

The role of acceptance tester may or may not be a full-time role on the project, but we have found that it tends to be a full-time and recognised role on the project team. It is essential the acceptance tester role is perceived both by themselves and by the rest of the team as belonging to the customer team, otherwise we can end up with technology-focused testing occurring, and no business-focused testing.

2.5 Pattern: User Interface Designer

Problem

Programmers are famous for not being able to design User Interfaces, at least as far as UI designers are concerned – not to mention users!

Forces

- We want an application that end-users will be able to use effectively to perform their tasks, UI design is more than just "looking good", it is all about interactions.
- Requirements for the system can be introduced from the UI Designer, which is another reason for the UI Designer to be in the customer team.

Solution

In the situations where the User Interface of the project is considered critical to the application, then it is essential that we hire a user interface designer. In other situations (i.e. the User Interface is not seen as critical) then it is still recommended that someone with user interface development experience is assigned to the customer team, as designing an application that meets the essential interactions for the end-user is an important part of any application.

Forces Resolved

We found UI Designers end up on the customer team, providing UI designs to the programmers. In the first case (UI is seen as critical), this role is likely to be a full-time recognised role on the project team. In the second case (all other situations), this role is likely to be performed in conjunction with another role on the project team, although it is highly recommended this role is not performed in conjunction with the role of programmer. One especially important reason that UI work be aligned with the customer is that UI design, in order to provide usability, may lead to new requirements.

Often the User Interface Design appears to be Big Design Up-Front (BDUF), because the UI designers do their own iterative design and evaluation for usability. We are finding that UI designers. in conjunction with the programmers, are learning together how to make this work in the incremental fashion of agile iterations.

2.6 Pattern: Technical Writer

Problem

XP downplays technical documentation but *user* documentation is still important. Programmers do not tend to have the technical writing skill-set required to write effective user guides.

Forces

- We want to deliver support documentation (e.g. user guides) to end-users so that they can use the application effectively.
- We do not want to interrupt delivery, or pay the cost of expensive programmers to write the documentation, poorly.

Solution

In the situations where end-user documentation is to be produced, it is essential that someone on the customer team has technical writing skills.

Forces Resolved

We have found that real technical writers often end up on customer teams, particularly when the application is a software product. Typically technical writers are assigned to the customer team on a part-time basis, and will often be assigned to multiple projects as a technical writer. The good news is that technical writers love agile development:

> *"At least I've got **something** to write about from the start"*
> *— Technical Writer, EagleCorp*

In traditional software development the technical writers often start writing the user guides from the requirements specification, only to find when the software product is delivered a few days before shipping, that the requirements specification does not reflect the software product's functionality, many long and intense hours are then spent re-writing the user guides to match the software delivered. the technical writer to spread their load, as the software changes

So, while agile software development allows and evolves during the process the technical writer will be required to evolve their technical documentation regularly. We must beware that this situation could quickly become frustrating for the technical writer and may result in them pushing for a more defined specification up-front, in order to better balance the amount of work and re-writes over time.

2.7 Pattern: Diplomat

Problem

Customer teams require organisational representative(s), including:

- Subject matter experts
- End users
- Senior stakeholders and decision makers
- Architects

They are responsible for representing their organisational area or perspective on the project. A project typically will involve many departments within an organisation, and will require multiple perspectives, which will not always see "eye-to-eye" as each area will have competing goals priorities and requirements. It is important to ensure each organisational area is represented, and one person cannot do that alone.

Forces

- We want to ensure that all of the organisation's requirements are brought into the project, not just the end-user requirements, or the senior stakeholders, or the architects, or the sales department or the production department and so on; but all of these different perspectives.

Solution

We have found that projects must ensure diplomats are identified from each organisational area and represent the views of this area for the project.

Forces Resolved

We have found that the people fulfilling this role may be full-time or part-time. To work successfully the Diplomats must have the time to participate in the project. Their role is significant and includes not only representing their requirements into the project team and working directly with programmers, but also finding out and representing the diverse views of the people they have been chosen to represent. Finally the people in this role must be open to negotiation and be able to understand other department's/area's perspectives and needs.

2.8 Pattern: Super-Secretary

Problem

Within the customer team there are many administration and organisational tasks that need to occur in order for the customer team to be effective in their interactions with both the business and the programmers. Overloaded customer team members find it easy to either let these tasks "slip" or become a burden that results in them either not being as effective (e.g. stories get lost) or working even more hours in a day.

Forces

- We want to ensure that customer team members are not distracted from their core roles by administration and organisational tasks.
- We want to ensure that the organisational tasks occur or stories might get lost, cards run out, and so on.

Solution

We have found that typically one person on the team will surface to pick up the administrivia load from the rest of the team; we have called that role the super-secretary.

We have recently identified that the name of this pattern causes some perception issues, and are considering renaming this pattern to "Steward" (based on the analogy of a King's Steward), which perhaps better illustrates the importance and significance of this pattern. Additionally it helps illustrate the role this person plays when the customer lead is unavailable.

Forces Resolved

We have found that the super-secretary always has another formal role on the customer team, so this role is always "part-time", despite the sometimes very large amount of work in the role. The super-secretary will typically always write down the stories, and keep them organised as well as track them through their lifecycle, often with a sticker system with different colours representing each stage. The super-secretary also undertakes other tasks such as:

- Following up the story status with the programmers
- Ordering stationery, including cards and marker pens
- Printing cards or tracking cards on the wiki, as required by the programmers or business
- Organising meeting rooms for iteration kick-offs or planning meetings

One thing to be aware of with this role, particularly given that it is always a secondary role, and often unrecognised role, is that the person performing this role can become very overloaded, and while we have named the role "super-secretary", it is fair to say that there is often a limit to this person's "super" powers. In one case we investigated, the super-secretary had become too overloaded and had recently left the project. The team was feeling the ramifications of her departure and perhaps becoming aware of the true load she had been shouldering for the best part of the year-long project. It is important to keep an eye on this person's load and consider ways to mitigate the overload they will experience.

2.9 Pattern: Negotiator

Problem

While the roles above are helpful, we need **an** on-site customer, someone who decides what to build when and talks to the programmers. None of the other roles do this, so where has the on-site customer gone?

Forces

- Programmers need to know who to talk to concerning their story, and they need to be confident that it is the "right" person, someone who has the confidence of both the person paying the bills and the end-user of the system; and who can talk to the programmer in a way the programmer can understand.
- Business people need someone to help them clarify their vision and ensure that an application gets built that will meet their competing needs, and will be accepted by both stakeholders and end-users as achieving the business goals/vision.

Solution

DeMarco [7, p5] suggested that negotiating "with a whole community of heterogeneous and conflicting users is a gargantuan task"; he goes on to liken the diplomatic skills required to "the skills of a Kissinger negotiating for peace in the Middle East." Like DeMarco we also noticed this negotiation role as essential, one of the key functions of this role is to negotiate or facilitate communication and agreement amongst all of the Diplomats, in order to provide a "single voice" of requirements to the programmers.

We also noticed that an onsite-customer is clearly identified on the team and is the single-point of contact for all initial programmer queries and decisions. Finally, like with DeMarco's initial writing, this role facilitates direct communication between the Diplomats and Programmers.

Forces Resolved

On every project we studied, everyone in the team (and typically outside of the team) could clearly identify the on-site customer. We have found that to be effective, customers must be able to be:

- Good (active) listeners
- Confident & decisive
- Comfortable working at the "big picture" and detailed levels
- Know their limitations and work with a customer team
- Handle intense pressure … workaholics should apply!
- Recognise multiple perspectives exist … and help them see each other's world

Despite the fact that there is a customer team, on most projects we have studied the person performing this role has clearly been overloaded, often leading to burn-out, or the person performing this role leaving the organisation after the project completes. The organisation looses this person's valuable knowledge and the application may suffer once the strong identified vision-holder is changed.

Interestingly, by clearly separating the responsibilities of the customer (team) from the programmer (team) we have also noticed a tendency for programmers to simply say "it's not my problem: that is for the customer to solve". We very much doubt that was Beck's intent with XP, to create a division, instead we believe his intent was very much to improve the communication between the customer and the programmers and have a whole project team approach. We believe that some of the practices we outline next directly help to remove some of the divisions created between the customer and the programmers.

3. CUSTOMER PRACTICES

This section outlines nine practices that enable customers to sustainably drive XP projects to successful completion – similar to "XP practices" but for *customers*:

- Programmer On-Site
- Customer's Apprentice
- Programmer Holiday
- Story Standards
- Show & Tell
- Customer Pairing
- Customer Counselor
- Look Before You Leap
- Three-month Calibration

While this list is not complete, it provides the initial core patterns that we have seen working on real projects that allow customers to do their jobs effectively. Later papers will extend and explore these practices further. Like the roles described above, these practices exist to support the customer role, but may be established with lesser or greater formality, as a situation requires.

3.1 Pattern: Programmer On-Site

Problem

The onsite customer can create a number of problems:

- If the customer needs to move physical location to become the "onsite" customer then there is a risk that they will become isolated from the business

organisation, and can also become prone to the Stockholm Syndrome.
- Programmers do not get to understand and respect the end-users of their application, as they have no knowledge of their world.

Forces

- The customer representative needs to remain grounded in their organisation and connected to all of the end-users, business stakeholders and political advisors.
- Programmers need to better understand and respect the end-users and other stakeholders.

Solution

We should re-tune our advice, and as well as an "on-site customer" and "programmer" roles, we should have "customer" and "on-site programmer" roles.

Forces Resolved

This advice of getting programmers into the field is not new. The pattern resembles Constantine's advice for *office visits* [6], and the old story that aircraft manufacturers offer their avionics programmers seats on early test flights.

Programmers need to understand the rhythm and flow of users jobs and experiences – who they are, what they do, why they do it, why they will ignore the software. This practice is not about making decisions but instead it is about understanding the end user and context of use, and gaining enough information to making helpful suggestions.

> *"I worked with a social worker, doing a death review. This is what she does every day, it helps put the importance of the system we are developing in perspective, while it might be the most important thing for me as I am 100% assigned to it and have deadlines, is it more important for her to help us or do her day job?"*
> *— Business Analyst, 2005*

Consider a comparison and contrast of the two versions of the practice that results in customers and programmers being co-located:

Customer On-Site	Programmer On-Site
Good for programmers	Good for users … if a nuisance!
Makes customer aware of programmer's jobs and issues	Makes programmers aware of the user's job and local issues
Incorporates customer into programmers culture	Incorporates the programmers into the user's culture
Customer capture is a "**bad**" thing.	But programmer capture is a **"good"** thing

However, it is important to emphasise that a little knowledge can be dangerous: programmers may end up believing that they "know best" based on their limited knowledge. Programmers need to understand that while they are gaining an appreciation and understanding of the end-users world they will never know it to the extent that the customer does, they will never have the customer's overall view.

3.2 Pattern: Customer's Apprentice

Problem

The previous practice helps bring the programmers into the world of the end-users, but they still lack an understanding of the world of their projects customer. We have found many situations where programmers see the customer as the bottle-neck and have no understanding why the customer cannot pump out stories to keep up with them.

Forces

- Programmers need to understand and respect their customer.
- The customer needs to manage their excessive work-load.

Solution

> *"To understand someone, walk a mile in their shoes"*
> *– Old Saying*

So, rotate programmers to act as the Customer's Apprentice:

- Attending meetings with users and stakeholders
- Writing stories and being secretary

Forces Resolved

We have found in all cases where this practice has been used that the programmers quickly change their tune, their complaints rapidly diminish and they become aware of the true load the customer carries on the project; they quickly gain a deep respect for the person playing the customer role. This practice tends to work best when the programmer acts as the Customers Apprentice for at least one iteration, otherwise the programmers do not truly get to see all of the demands on a customer's time. Programmers who have played the role of the Customer's Apprentice are more likely to see the team as a whole team and will step in to help the customer out when the customer becomes overloaded, and will also "defend" the customer within the programmer team, helping other programmers become aware of the true demands on the customer.

However, convincing programmers can sometimes be problematic. In our experience, this practice works best when the programmers suggest it themselves. But it is possible to head-hunt a good programmer candidate for this role when the customer becomes aware of the programmer "grumbles" at delays, or, more helpfully, realises the customer is overloaded and needs extra help.

3.3 Pattern: Programmer Holiday

Problem

XP is intense; thinking about the requirements of the new system is hard and sometimes Customers just need more time to get ahead of programmers, as the "stay ahead" dynamic is really important.

Forces

- A balance is needed between the need to deliver working software with ensuring the customer does not burn out and that the project delivers what the business truly needs.

Solution

Customers need to "send the programmers on holiday" when they need time to focus on communication with stakeholders, and cannot commit to new stories or priorities.

Forces Resolved

A "programmer holiday" is not often an actual holiday (although there maybe times when that is indeed appropriate), but mostly the customer will choose to prioritise technical debt, lower priority bugs, technical system upgrades and such above stories. Other wise and long-term effective strategies include implementing the "Programmer's On-Site Day" and "Customer's Apprentice".

3.4 Pattern: Story Standards

Problem

How do you break a problem down into stories that are both meaningful to the business and at the right level of detail for the programmers: how big should a story really be?

Forces

- We need to provide a consistent way of writing stories
- We need to find a way to break problems down, to ensure that we understand the context of a story so that it can be effectively prioritised. For example, in one project we are aware of, the business prioritised their stories, and these stories were built and released. The problem is that no end-user could perform any task with the software, because sometimes lower priority stories need to be implemented in order to deliver software that provides value to an end-user.

Solution

We need to provide a common template for every story. The most effective template we have seen in use is the form "as a *persona* I want *something* so that *goal is achieved*" [4].

However, that is not enough. We also need to provide a larger "container" in order to effectively prioritise development work so that it provides value to the business, one method of for this is use case identification [3] and yet another is user-centered design task analysis [5].

Forces Resolved

Customers need to take time to get stories right; story decomposition and prioritisation is difficult. Story standards and ways to organise the stories so that the business can prioritise at a higher level of granularity are essential (e.g. use cases), but we need to be very careful so that stories do not simply become specifications and the importance of a conversation between programmers and customers is lost!

3.5 Pattern: Show & Tell

Problem

Middle level bosses need to be convinced the software is making progress, programmers need hard milestones, and in the case of product development, Sales and Marketing need software to demonstrate to clients in order to solicit their feedback.

Forces

- We rely on demonstrating progress with working software rather than Gantt charts, so we need to actually demonstrate working software to the people interested in our progress!

Solution

Schedule regular demonstrations of the working software to those internal or external parties that are interested in (or need to provide feedback into) the project/application being developed.

Forces Resolved

We have found that demonstrations are one of the most effective ways to:

- Gain the trust of senior and middle management, once they see progress and are assured it is not "smoke and mirrors" they become more confident that the project will meet its deadlines. It is often worthwhile retaining status quo reporting and demonstrations until such time as management become comfortable enough with the demonstrations to remove the need for the overhead of MS Project and all that it entails.
- Obtain regular feedback from the larger external population who will be end-users of the system [8], either internal users in the case of in-house business applications or external clients in the case of product development. In many cases, sales and marketing departments are able to leverage this opportunity to not only inform the direction of the application but also to provide confidence to this community that this project will deliver value, that it is not simply vapour-ware.

Demonstrations can also be useful internally within the project team, however one essential for internal demonstrations is that they need to add value. Often, with the programmers and customers working closely together, the demonstration of functionality is not the thing that will add value, rather it is more likely about the environment, and issues of integration or stability. On one project we investigated weekly internal demonstrations were the norm. Programmers could not see the value to themselves but thought they added value to the customer, and customers could not see the value to themselves but thought they added value to the programmers. No-one commented on the valueless practice during retrospectives as each group believed the practice added value to the other group. In the end, we discovered demonstrations were instigated at the start of the project when there were a number of environment problems and no-one had re-questioned their use since then. So, whether it is an internal or external demonstration, always confirm that a demonstration adds true value to the participants.

3.6 Pattern: Customer Pairing

Problem

The on-site customer is overloaded and many are suffering burn-out despite being in a customer team. One of the significant parts of the problem is "being alone", making hard decisions alone.

Forces

- We need to find a way to provide effective support to On-site customers so they do not feel alone
- Customers need to discuss issues that do not relate directly to their stakeholders nor the developers, but to the decision-making process itself
- Stakeholders and developers cannot be expected to discuss the process itself dispassionately with the customer, as they are affected, and have their own concerns.

Solution

If Pair programming is good – pair customering must be good!

Forces Resolved

It works for the customer, in every case we've found. It is important to consider the most effective technique for managing customer pairs for your project; the solutions we have seen include:

- Divide by functional area
  - "Along the grain of the domain" (Brian Foote)
- Divide geographically
  - Distributed projects
- Inward/outward division
  - One customer works with the programmers
  - Another works with stakeholders & users
- Visionary & Detail
  - One has the Visions! Goals! Plans! Dreams!
  - Another does the work
- Most extreme: like pair programming – don't divide.

- Close working relationships are key to any division of the customer role

However, programmers can find it difficult to work with a customer pair, as at times the different pair members will provide different directions. Additionally in some situations we have seen the programmers have played the customers like divorced parents, to obtain the decision they preferred or thought was the "right decision". From the perspective of this paper, that's not nearly as large a problem as the customer overload leading to potential burn-out which has been identified as part of this research, because a burnt-out customer will take down the whole project!

3.7 Pattern: Customer Counsellor

Problem

The customer role is a lonely and intense role that we know has "caused" burn-out. Programmers get a coach, and customers need someone too.

Forces

- The customer needs someone to talk to, to help them resolve their issues, ensure they realise they are "not alone" and to mitigate the risk of customer burn-out. To be effective this person should:
  - Not be on the project
  - Not be a manager
  - Have enough IT & business experience to provide effective and pragmatic support
  - Will not try to solve their problems
  - Is someone the customer can trust

Solution

We need to provide professional support to customers, a Customer Counsellor (think Deanna Troi from Star Trek: The Next Generation). This practice combines the patterns *Mentor* and *Shoulder to Cry On* [9] outlined by Mary Lynn Manns and Linda Rising.

Forces Resolved

We have found that the Customer Counsellor practice makes a difference to the well-being and effectiveness of the customer. To be effective the Customer Counsellor meets the customer regularly in a private place that is completely confidential. If the customer gets stuck they can call the Counsellor straight away

The Coach and Customer Counsellor could be the same person as there is no intrinsic conflict of interest, but more realistically they will be separate people as the skill sets are different.

3.8 Pattern: Look Before You Leap

Problem

Software projects, even small ones, cost money (e.g. 10 programmers for 6 months can cost upwards of $1M) and someone needs to decide if that investment is worthwhile before the project begins.

Forces

- We will need to prepare a business case, a requirements document and some kind of scope before organisations authorise projects.

Solution

The customer should lead initial analysis and design workshops for a short period, typically 2-4 weeks, before coding begins. These sessions are release planning and scoping sessions, working out at a high level what to include in each release. We need to do some "pre-thinking" about what we are going to build and why, and ensure that this project adds value to the business. In many cases we need to provide the information that will allow senior executives to decide which projects to invest in (i.e. prioritise the projects within an organisation).

A number of case studies [1, 12] show aspects of this pattern in their day-to-day use of agile.

Forces Resolved

To be effective the (small) up-front analysis should be:

- Led by the customers,
- Involve end-users, stakeholders (across multiple departments where appropriate), to ensure that a shared understanding of the problem and solution is developed that takes into account the multiple perspectives of the project within the organisation.

- Sympathetic programmers should observe so that they can gain an understanding of the business and be able to more effectively estimate during release planning sessions.

It is important to put a reasonable time-box on this "research" activity. Analysis (or problem definition and solution clarification) will not be complete at the end of this process, only enough to make a decision as to whether the project is worth taking further. Too large a time-box could put us back into the same place we have been with traditional Big Up-Front Analysis, too small a time-box will mean we could attempt to start a project that is not the most important to the business, or just as easily the reverse, miss a project that would have added significant value to the business. Our research tends to indicate that the 2-4 week time-box is about right for most projects.

3.9 Pattern: Three-month Calibration

Problem

After three months, many teams realise that "their eyes were bigger than their stomachs", and they aren't going to deliver everything they promised. At this point in the project the customer typically feels *absolutely betrayed!*

Programmers have insisted on ruthless prioritisation, so the spec is the *absolute minimum* the business can accept, or perhaps the customer believed they would get the medium-priority stuff. Even if customer has been on a project before, they believe XP will deliver (that's why they picked it). Interestingly enough, the customer's sense of betrayal is larger than on traditional projects, perhaps because XP and Agile emphasise prioritisation or perhaps because the customer believed XP/Agile was a silver bullet. Whatever the cause, we should be aware that the backlash is strong, and a pat answer of "well XP/Agile let you know this sooner than traditional software development" is often not well received.

During the crisis period, XP projects

- Typically stop doing iterations and development
- Have lots of meetings with stakeholders & bosses
- Re-plan the release
- Redo budget and scope
- The customer has to do lots of selling to the organisation; programmers don't often realise just how serious the situation is.
- The customer finds it hard to come up with stories as they are even more overloaded, and is unsure what to prioritise given the project may be doomed.
- Morale low throughout whole team

Forces

- We need to find a way to ensure that the business is prepared for this event, as it has happened on all of the projects we have studied.
- We also need to find a way to ensure that the programming team are also prepared for this event and they realise the seriousness of the situation.

Solution

We recommend customers understand this event is a possibility, be ready to recognise it when it occurs, and be prepared to address it on its own terms, rather than with outrage or denial.

Forces Resolved

We recommend that we consider doing this every *season* (about every 3 months). In some ways this is nothing new, it is just XP release planning: but it is essential to do it and set customer expectations that it will need to occur.

We must manage customer expectations concerning this event; we must be more upfront that customer won't get everything. They will get something but there is no guarantee that it will be enough!

4. REFERENCES

[1] Armitage, A., Wisniewski, P., de-Ste-Croix, A. Proceedings of Agile 2007. Greater Successes by Using Agile Techniques Closer to the Light Bulb Moment. Pages 181 – 186. Washington D.C. IEEE Computer Society.

[2] Beck, K. eXtreme Programming Explained: Embrace Change, Addison Wesley, 2000.

[3] Cockburn, A. Writing Effective Use Cases. Addison-Wesley. 2000.

[4] Cohn, M. User Stories Applied: For agile software development. Addison-Wesley. 2004.

[5] Constantine, L., Lockwood, L. Software For Use: A Practical Guide to the Models and Methods of Usage-Centered Design. Addison-Wesley. 1999.

[6] Constantine, L. The Peopleware Papers: Notes on the Human Side of Software. Yourdon Press. 2001.

[7] DeMarco, T. Structured Analysis and System Specification. Yourdon Press. 1979.

[8] Gatz, S., Benefield, G. Proceedings of Agile 2007. Less, Never More: Launching a Product with Critical Features and Nothing More. Pages 324 – 327. Washington D.C. IEEE Computer Society.

[9] Manns, M-L., Rising, L. Fearless Change: Patterns for Introducing New Ideas. Addison-Wesley. 2005.

[10] Martin, A., Biddle, R., and Noble, J., Proceedings of the Second Agile Development Conference, Sherman Alpert (Ed.), The XP Customer Role in Practice: Three Case Studies. Pages 42-54, Salt Lake City, USA, 2004. ACM SIGSOFT.

[11] Martin, A., Biddle, R., and Noble, J., Proceedings of the Fifth International Conference on eXtreme Programming and Agile Processes in Software Engineering, Jutta Eckstein & Hubert Baumeister (Ed.), When XP Met Outsourcing, Lecture notes in Computer Science 3092, Springer-Verlag, 2004.

[12] Takats, A, Brewer, N. Proceedings of Agile 2005. Improving communication between Customers and Developers. Pages 243 – 252. Denver, Colorado. IEEE Computer Society.