COMP33711 Agile Software Engineering: Case Study

A Monday at ThoughtWorks Manchester

Suzanne M. Embury
School of Computer Science
The University of Manchester
Oxford Road
Manchester M13 9PL, UK

25th June 2012

In June 2012, the ThoughtWorks Manchester team kindly invited me to spend some time in their office, observing them during a typical morning's work. The notes and photos below show the team in action, and will help you see how the concepts covered in COMP33711 are employed within an organisation that is committed to the practice of agile software engineering.

ThoughtWorks is a global software consultancy that has been influential in the development and promotion of agile methods. It has offices all around the world, including Chicago, London, Singapore, Brisbane, Bangalore and (of course) Manchester. The company writes bespoke software, and also provides agile consultancy and coaching services. Some major projects they have been involved in include: the new Guardian website, thetrainline.com website, The Lonely Planet website and a new mobile phone comparison site for Which.

As well as undertaking many large projects using agile methods, it has a product division (ThoughtWorks Studios), which develops tools for managing agile products (currently the Go™ release management platform, the Mingle™ project management tool and the Twist™ agile testing tool). The company also encourages its employees to contribute to open source projects, and to make use of open source tools in their development. For example, ThoughtWorks was the brains behind Cruise Control, the first continuous integration server, and has also contributed to projects such as Selenium, JBehave and NUnit.

The Manchester team's primary role is development of software for ThoughtWorks' own corporate web site. Since one of the main goals of the corporate web site is to attract new clients, the work of the team is highly visible, giving prospective clients a flavour of the organisation's capabilities.

At the time of writing, the Manchester team is engaged on a project to develop the parts of the website aimed at supporting recruitment: join.thoughtworks.com.

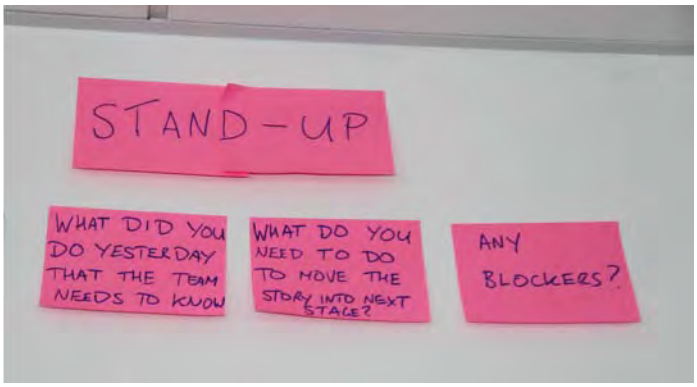
The goal of the recruitment site is to provide a series of web-accessible functions aimed at people interested in applying for positions at ThoughtWorks. This includes information about open positions and the recruitment process, as well as some more unusual features. For example, the team is implementing functionality to allow any current employee of ThoughtWorks to create his/her own profile, that is immediately available to members of the public interested in finding out about the experience of working for ThoughtWorks.

The team works on a 1 week iteration, starting on Wednesdays, and releases to the live site every Tuesday. The full team is distributed between Manchester, London and Pune (India). The Wednesday start date for iterations was chosen partly to avoid releasing new functionality immediately before the weekend (as would have been the case with a Monday-Friday iteration), and partly because Wednesday happens to be a good

day for all the different teams to be able to engage in the conversations needed for iteration planning without distractions. My visit was on a Monday, so the team was focussed on finishing off the stories for the upcoming release, as well as grooming the stories and tasks to be ready for the next iteration planning meeting.

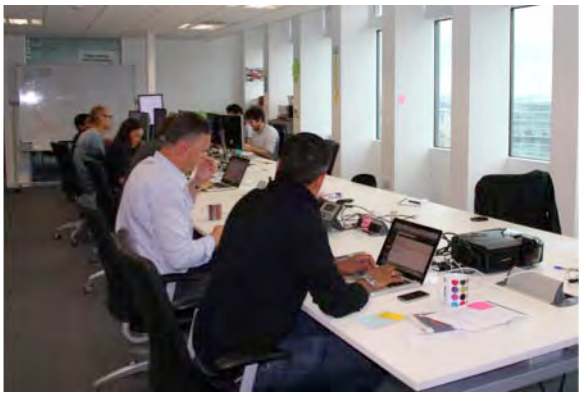
9.00am Daily Stand-Up

The day starts at 9.00am with the daily stand-up meeting. Five members of the team are present: Kate (the business analyst), Torsten (the quality analyst), Andy (the technical lead) and Ignacio and Ashok (developers). As you can see from the pictures, the team stands at the story wall for the meeting. They pass a small ball around the team, to indicate whose turn it is to speak. Each team member gives a brief 1-2 minute summary of what he or she has been working on, and how it has been going. They also mention any "blockers". A blocker is anything that is preventing progress on one of their allocated stories. For example, one team member mentions a knotty technical problem he has been working on for some time, and another team member suggests "pair swapping" to generate some new ideas for how to fix it. The daily stand-up is also used to let team members know about other team members who are on leave, and remind them about upcoming events for the week.



At the end of the stand-up, the team chooses new pairs for the day. Each pair decides which stories/tasks to work on, and uses their avatars to indicate who is working on what. Then, everyone heads for the desks to get on with the tasks they've selected.

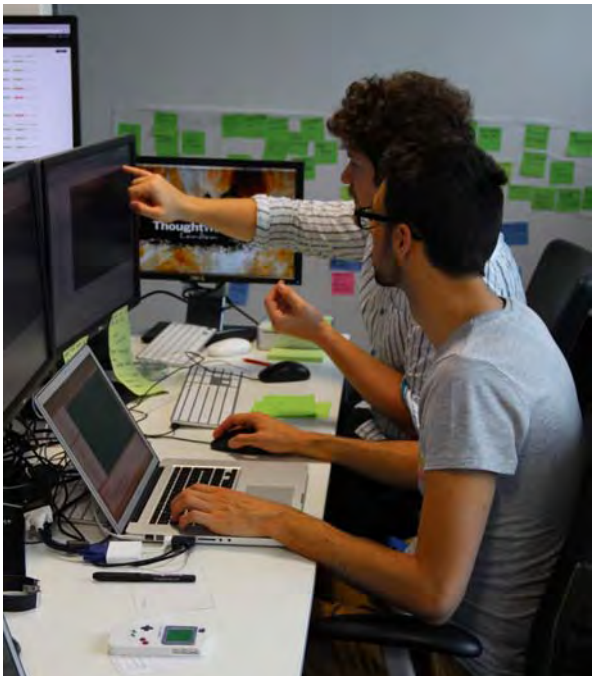
The whole meeting takes around 7 minutes.



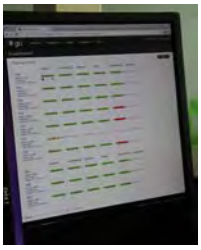
9.10am Pair Programming

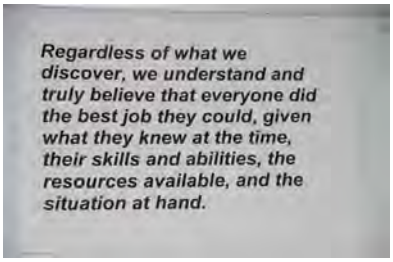
After the stand-up, the developers sit down in pairs to get to work. Tom, another developer, arrives, so that there are two developer pairs working at the same time. In each pair, one person (the driver) codes at the keyboard while talking aloud about what they are doing and what they hope to achieve. The other person (the navigator) asks questions and provides suggestions. The roles are swapped regularly, at convenient points in the development.

What I find noticeable about the coding activity going on in the room is that the pairs seem to do a lot more talking and thinking than typing. The pairs are focussed on understanding what the specific task in hand really requires, so that time is not wasted writing code that is not actually needed. When a member of the pair takes on the driver role, they begin by explaining to the navigator what (test or production) functionality they wish to implement, and the other member, the navigator, asks questions to clarify their meaning. Together, they consider different implementation options, and continually question and discuss as each line of code is being written.



Information on the current status of the build of the system, and the various test suites, is displayed on a large screen positioned at the end of the worktable, where it is easily visible to the developer pairs. A green bar indicates that a particular part of the continuous build and test process has passed. A red bar indicates some kind of failure. The column headings are: Unit Tests, CreatePackage (a build process), Integration Tests, Smoke Tests and Regression Tests. The pairs use this screen to get rapid feedback on the code they check in to the version control system, and the effect it has on the quality of the build.

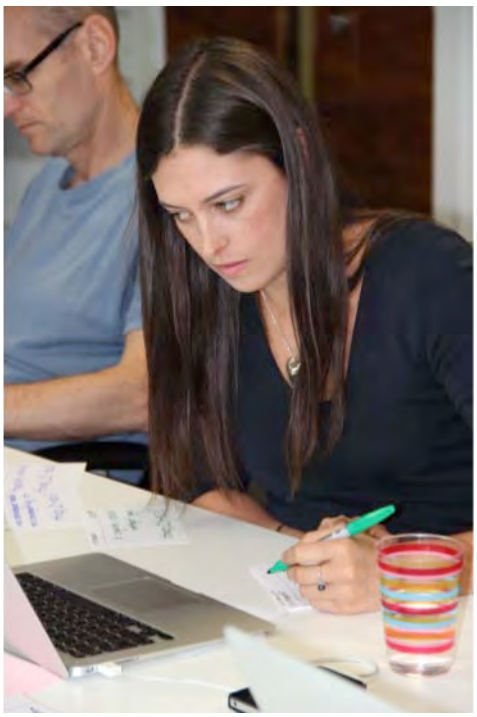




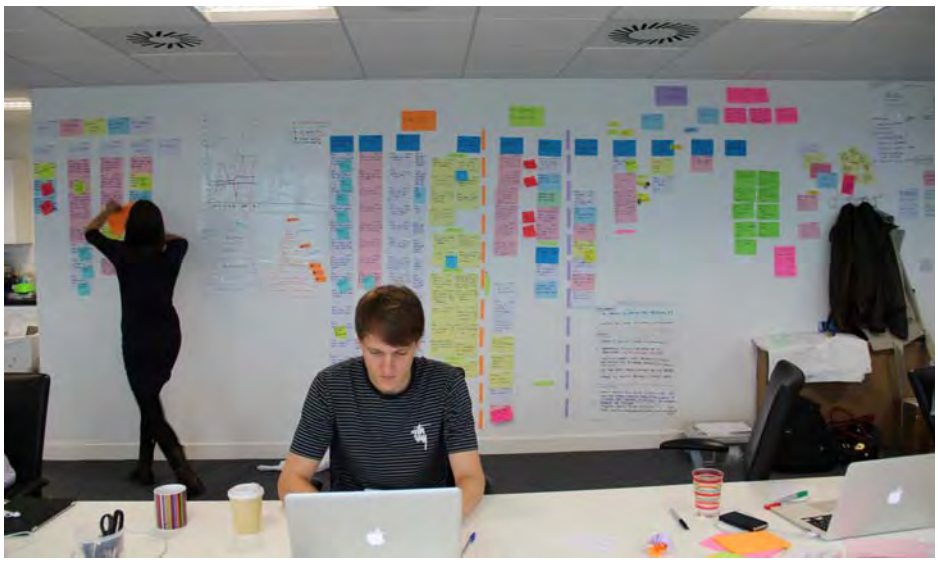
9.40am Updating the Story Wall

After the stand-up meeting, Kate (the business analyst) works on creating new cards and updating the old cards on the story wall. The story wall contains a mixture of requirements information, project tracking information and general team information. Everything is presented in a low cost way (hand-written index cards) but which has visual impact and is easily absorbed by the members of the team.

The team keeps its stories on a wall, and also in Mingle (the agile project management tool developed by ThoughtWorks). Kate has responsibility for managing the story collections on both the wall and in Mingle, but the whole team is encouraged to own the stories. Any member of the team can make changes to the wall, and has the responsibility of keeping the Mingle instance consistent with the wall when they do.



The Mingle instance has the full text of stories, plus acceptance tests, whereas the cards on the story wall contain only the headline details. Despite this, Kate considers the story wall to be the primary representation of the current state of the project and the requirements, not the Mingle instance. The story wall is visible to all the team all the time, and all the key discussions about what the team should deliver to the customer, and how it should be done, take place in front of it. The Mingle instance, on the other hand, is useful as a place to store additional details about stories that will not fit conveniently onto the cards. It also provides useful statistics on the progress of the team, and it acts as a historical record of the activities of the team (something the rapidly changing story wall cannot do).



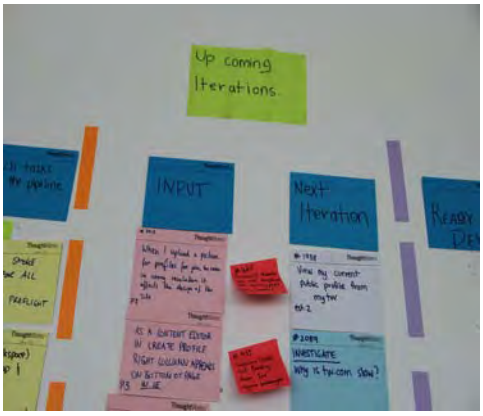
Kate and Torsten also report that, as analysts, they see the benefits of the limited space on the cards. Index cards contain so little information that developers have to come and speak to the analysts about what functionality is actually required before they start work on it. It isn't possible to just take a story, then go and sit in a corner and implement your own version of it, in isolation from the rest of the team, without checking first that you have really understood what the customer requires.

A major part of Kate's role as an agile business analyst is to work with the customer (in this case, a senior recruitment manager at the London ThoughtWorks office), to gain an understanding of new requirements, to create the story wall cards needed to describe the new requirements and to understand how they relate to the rest of the project. She also works with the customer to decide which stories to "play" in each up-coming iteration. She speaks with the customer at least once a week, using Skype, and expects to be travelling down to the customer every couple of weeks or so, in the future.

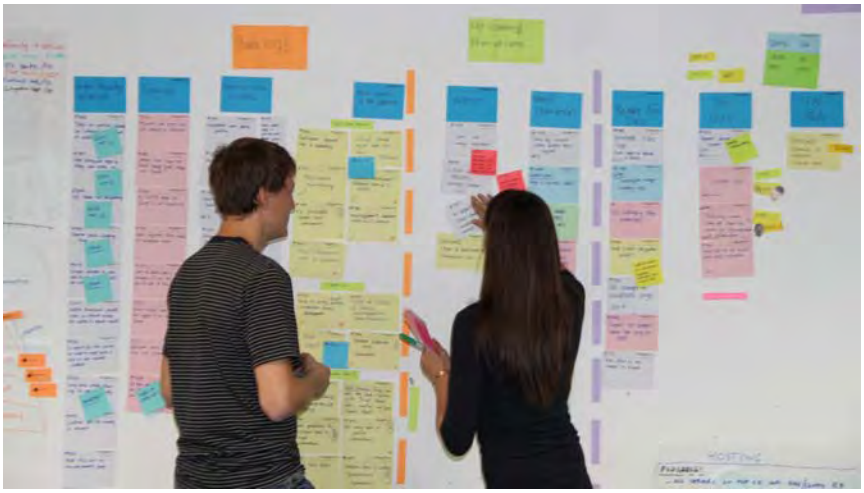
Another aspect of Kate's role is to pair with Torsten, the quality analyst, to make sure that suitable acceptance tests are added to each story, and that the stories on the board represent a meaningful and useful description of the customers' requirements, as they are understood at each stage of the project.

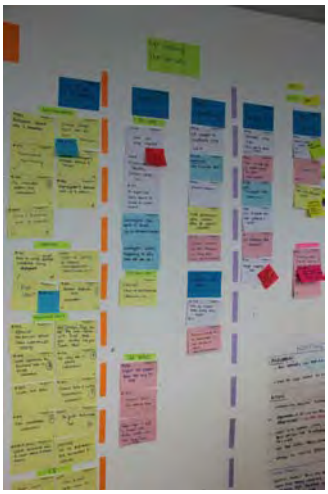
10.45am Discussion of Story Wall Organisation

In line with the agile philosophy of continual feedback and improvement, Kate is always thinking about how the story wall can be made to work better for the team. Today, she is reorganising the board to bring in several new ways of working that she wishes to try out. One recent change, for example, was to introduce the new column called "Next Iteration", which contains the stories that the customer (with help from Kate and the team) has chosen to be queued for implementation in the upcoming iteration. This allows Kate to capture the outcomes of her discussions with the customer, and to make them immediately available to the team, for use in making short-term planning decisions at any point in the current iteration.



Torsten and Kate discuss some post-it notes that Torsten has recently added to the story wall, concerning a number of bugs that have been reported. Together, they come up with a new way of organising the "Input" column. This is the column that contains the stories that are high priority for the customer, and from which stories for the next couple of iterations will be selected. They decide to split the stories in this column into groups labelled "BA Input" (business analyst input), "Tech Dept Input" (technical department input) and "QA Input" (quality analyst input). Together, they convert the post-it notes into cards, and organise them on the story wall.



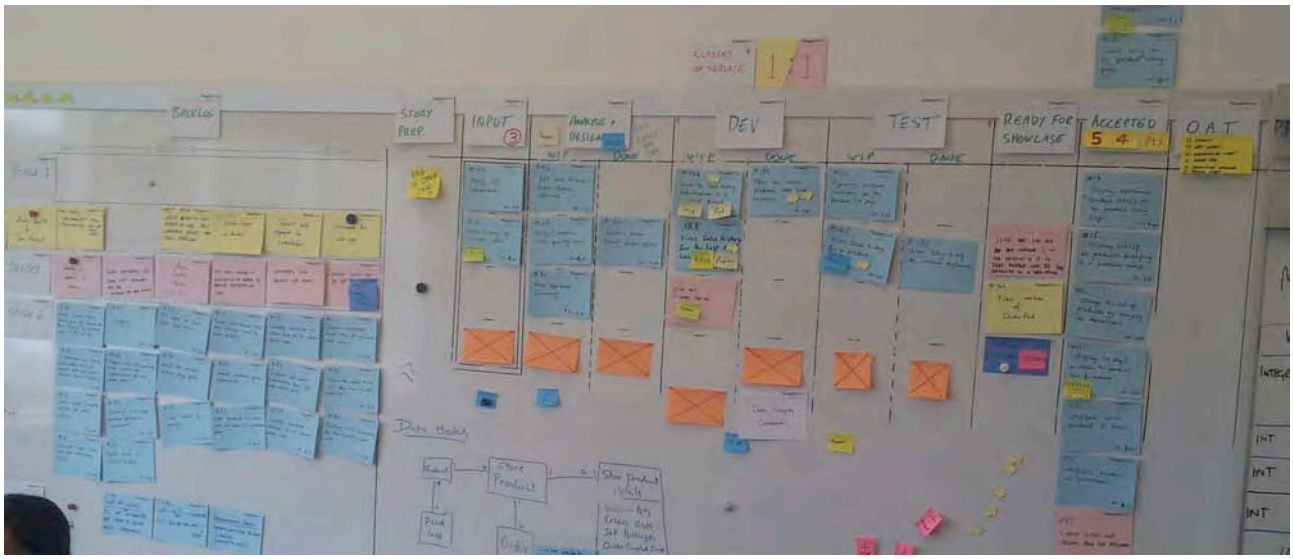


11.00am Remote Stand-Up Meeting

As well as the team members we have met so far, the ThoughtWorks Manchester office includes other staff with a variety of technical and administrative roles. One of these is Ian, a delivery principal working with other ThoughtWorks teams around the UK. During the morning, he uses Skype to hold a remote stand-up with a team working at a client's site in Bradford. The Bradford team position their webcam so that their story wall can be viewed by Ian, and members of the team talk him through the progress so far. Even from the small webcam image on Ian's compact laptop, it is easy to gain an impression of the status of the project from the story wall.

The client is also present at the stand-up, and takes part in the discussion. This is important, as it means that progress on the project is kept visible to the client. There is no need for uncomfortable meetings where major problems and delays with the project have to be confessed to the client, after the team has made several failed attempts to deal with them. Instead, the client sees the detailed progress of the project at first hand, and is part of the discussions about how to respond to problems when they are first discovered. This means that the team is always following the line of action that is most productive to the client, even when problems and delays are encountered.

The Bradford team uses a Kanban board to manage their stories, as shown in the figure below. On a Kanban board, the columns are called “queues”, and they represent tasks as they pass through particular phases of the project. Notice the orange post-it notes at the bottom of each queue. These limit the number of stories that can be placed in each queue, and thus help to prevent bottlenecks. In the case of the example board below, there are only a few staff available to work on QA tasks, so a tight limit is set on the number of QA tasks that can be added to the board. Similarly, the number of “In development” tasks is limited, so that a bottleneck does not form around QA tasks. If the developers have filled all the QA task spaces, they cannot work on any more development tasks until a space opens up for one in the QA task queue. Instead, they have to find another, more immediately valuable way to use their time, rather than just adding more tasks to the slow QA queue. For example, they might put some effort into helping clear the QA queue, or work on some other slow-moving part of the process.



After everyone has reported back in the remote stand-up, Ian briefly checks that the team is aware of some relevant features of the project that have come to light. A team social event is mentioned and then the meeting ends. It has taken barely 10 minutes.

12.30pm Discussion Regarding Technical Debt

The recruitment website team participates in a discussion of which cards to play in the up-coming iteration. One of the cards under discussion relates to a technical debt issue, while another describes a bug to be fixed.

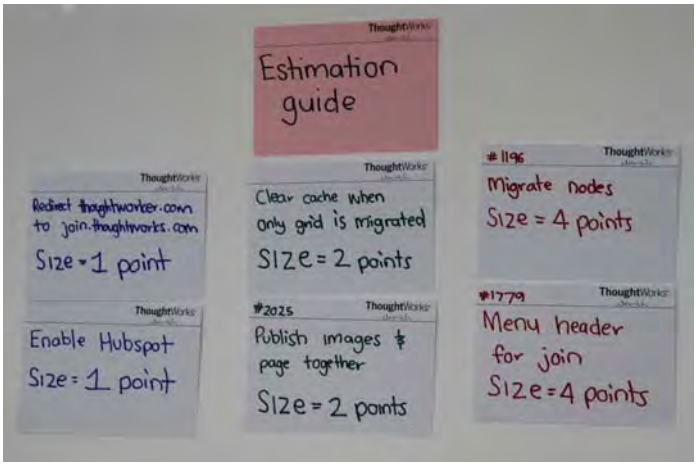
In addition to the developers, both Kate (the business analyst) and Torsten (the quality analyst) are involved, so that the customer focus and quality focus can both be taken into account when deciding how to proceed. The discussion covers both technical and customer requirement issues. "How often do they change the events?" wonders Andy (technical lead), thinking about the dynamics of the data that the application will manage. "I don't want to play the card if [the customer] won't get what they want quickly", says Kate, thinking about the need to deploy the team's efforts in a way that maximises business value for the customer.

After this, the developers brainstorm ideas for solutions on a whiteboard. When consensus is reached, the outcome is converted into a set of tasks on post-it notes, and one of the developer pairs sets about working through the tasks.

Reference Stories for Estimation

Although I did not get to see the team perform any story size estimation during my visit, I did talk briefly with some members of the team about how they do it. They use a standard agile estimating approach, that aims at producing "good enough" estimates quickly, rather spending a lot of time trying to get more precise estimates that may still be wrong. Instead of trying to estimate the number of person-hours required to complete a story, agile estimators work to produce estimates of story "size", in valueless units commonly called "story points". Story points are only meaningful when considered in relation to a story point estimate for another story by the same team. For example, if a team gives 1 story point to story A and 2 story points to story B, this means they judge that story B is estimated to require about twice the effort required to implement story A.

To help the team to reach good estimates quickly, Kate puts some reference story cards on the story wall, where they can be clearly seen during estimating. These are stories that were implemented in previous iterations, and whose story point estimates were confirmed by the effort needed to complete them, relative to one another.



To arrive at estimates, the team uses a popular approach based on the game of rock-paper-scissors. When a story is selected for estimating, each member of the team reads the story card and privately decides what their estimate for it is, in story points. Team members put one hand behind their backs, holding out the number of fingers that corresponds to their estimate. On a count, all team members reveal their hands, so that the different estimates can be seen and compared. Any differences between the estimates can be discussed, so that a consensus estimate can be reached while all team member viewpoints are taken into account.

Concluding Remarks

Two key points stood out from my observation of the ThoughtWorks Manchester team at work. The first of these is how comfortable the entire team was with change, and how change is embraced as a positive thing that moves the team towards its goal, rather than a barrier and a nuisance. The most striking example of this was the degree to which the contents and organisation of the story wall changed over the course of the day. Although no new stories arrived from the customer on the day of my visit, the way the stories were modelled as individual cards varied, along with their placement relative to each other, as the teams' understanding of how best to capture the state of their project evolved throughout the morning. Kate tells me that this degree of change is by no means unusual; the wall changes significantly almost every day.

However, there were many other examples of the team dealing gracefully and flexibly with change, taking it as a matter of course. The developers changed their pairs readily at the start of the day, without worrying about seniority, politics or personalities. Everyone was ready to work with every other member of the team, as the task in hand demanded. When an issue that needed the whole team's attention arose, the pairs broke off from their current tasks without complaint, and resumed them just as naturally once the issue had been resolved. When Torsten needed to add information about bugs to the story board, he added it quickly as post-it notes, rather than spending time thinking about how the board could be reorganised to accommodate it. Later, when the number of bug post-it notes increased to a level that made them a hindrance, he and Kate discussed how to handle the information better. They worked together to translate the post-it notes into story cards, and to rearrange the board. The developers were comfortable with this rearrangement occurring while they developed, knowing that Torsten and Kate were on hand to answer questions, should the meaning of the new story board be unclear.

The second key point was the way the work of the entire team was organised around conversations. When pairing, coding becomes a continuous conversation between the developer pairs. But this same principle applied across all members of the team. On a couple of occasions, a few people would head off to separate meeting rooms, but for the most part the entire Manchester office staff carried out their work around the single large table. This meant that it was easy to get information from another team member, simply by walking over to them and pulling up a free chair. But it also facilitated general information sharing throughout the team. For example, during one conversation, Amy (a recruitment manager) was describing to Ian how she was planning to use cards and a story board approach to keep track of the status of recruitment for various posts. Another member of staff who was working nearby had had experience of doing just that while working at another ThoughtWorks office, and was able to pass on some useful contacts to Amy. That interaction wouldn't have happened, had Amy and Ian's conversation been taking place in a separate meeting room.

Overall, my visit gave me a fascinating glimpse into the realities of agile development. I'd like to thank the ThoughtWorks Manchester team for their tolerance of my questions and my camera, during my stay.