Chapter 19
Roles and Responsibilities in Feature Teams

Jutta Eckstein

Abstract Agile development requires self-organizing teams. The set-up of a (feature) team has to enable self-organization. Special care has to be taken if the project is not only distributed, but also large and more than one feature team is involved. Every feature team needs in such a setting a product owner who ensures the continuous focus on business delivery. The product owners collaborate by working together in a virtual team. Each feature team is supported by a coach who ensures not only the agile process of the individual feature team but also across all feature teams. An architect (or if necessary a team of architects) takes care that the system is technically sound. Contrariwise to small co-located projects, large global projects require a project manager who deals with—among other things—internal and especially external politics.

19.1 Introduction

If the world would be ideal, then this chapter would be superfluous. Because then everyone working on a system would do the right thing. For small co-located agile teams software development often comes close to the ideal case. Yet, unfortunately this isn't the case for large global projects—projects that are spread over at least two sites and consist of more than one team or more than fifteen developers. In the latter setting various roles and responsibilities have to be implemented for successful collaboration.

The first section talks about the more classic roles, like database expert or user interface designer and what happens with those roles in an agile feature team. The next section elaborates on the role of the product owner. This person should steer a feature team business-wise. In a global project this means that the product owner needs to be close to both—the feature team and the customer. And moreover, in a

J. Eckstein (✉)
Gaussstr. 29, 38106 Braunschweig, Germany
e-mail: feedback@distributed-teams.com

D. Šmite et al. (eds.), *Agility Across Time and Space*,
DOI 10.1007/978-3-642-12442-6_19,

large global project many feature teams work on the same product. Thus the product owners of the diverse feature teams need to collaborate so that the overall delivery is in the mind of the customer.

The third section discusses the role of the coach. This role ensures adherence and necessary changes to the agile process. In a large global project every individual feature team requires a coach and all the coaches together need to take care that the overall agility isn't lost in the interworking of all feature teams.

The next section looks into the various possibilities of ensuring conceptual integrity, or technical soundness of the overall system. Depending on the complexity of the project and the knowledge and skills of the project members, each feature team might require one architect. In other circumstances a single architect for the entire project might be enough. If more than one architect is required, the people taking this role have to work together. Otherwise conceptual integrity is out of reach.

The fifth section explains the responsibilities of a project manager in a large, global, and agile project. And finally the last section elaborates on the location of the various roles. Especially key roles should be close to the team they are supporting. The product owner steering a specific feature team should for example be co-located with the respective feature team.

19.2 Context

The experiences described in this chapter refer to several projects I have been working on. Moreover, I verified some of my experiences through frequent exchanges with colleagues of mine. Please find below some ranges for the characteristics of these projects:

- Most of my projects are rather large in terms of people, between thirty and 300 project members.
- The smallest distribution degree is two sites within one country and the biggest one six sites spread over the globe.
- The following countries were involved: Austria, China, Czech Republic, Germany, Hungary, Poland, Singapore, Switzerland, UK, USA.
- The domain for the system varied very much, we built systems for embedded products, financial applications, mechanical engineering, multimedia, and telecommunication.
- Although I refer to all of them as "projects" this doesn't cover the truth, because some of them are actually product development. The biggest difference between the two is that product development has no ending—as soon as the first version is shipped this product is maintained and additionally the team works on the next version of the product.
- Some of these "projects" were greenfield applications whereas other ones were existing ones with a lot of legacy.

I hope this context description helps in order to classify and better understand the experiences described and the recommendations given.

19.3 Configuration of a Feature Team

A single feature team should always be able to deliver whole business functionality (features or stories) or in other words business functionality should never be split across several teams.[1] Therefore, for a feature team to perform well the members of the team should either already comprehend, or be capable of acquiring the required knowledge that is necessary to complete a unit of business functionality. In addition to the domain and technical know-how that has to be present in each individual feature team, the team also requires the knowledge to actually deliver the functionality. In a typical feature team its members will fulfill the roles of architects, database administrators, designers, technical writers, domain experts, infrastructure specialists, integration experts, programmers, testers, and user interface designers.

Typically in agile teams, feature team members take turns in fulfilling these roles. There is hardly a single person taking the responsibility for only one role. Head monopolies—a few people who are regarded as the only experts for a specific area—have to be avoided, because they create a high risk. The project will have difficulties making any further progress if these people are for example on vacation, sick, or change jobs.

Ideally a feature team consists of seven, plus or minus two member (the Miller rule) and stays together for the whole lifetime of the project. Yet, in many projects the individual feature team size depends on the complexity and size of the domain area this feature team is responsible for.

The development of some distinct features might require a specific knowledge for only a certain amount of time. For example, in one of my projects we rarely required the expertise of database specialists for data migration (but most of the time this knowledge wasn't needed). We had therefore only a few migration specialists for the whole project, but not for every feature team. In such a situation, I recommend that this migration specialist gets part of the feature team only while the corresponding features are developed within this team. Very often such a feature team membership lasts only for one iteration.

Yet, preferably the required know-how should exist within each team, or else the team should be supported to build it up. Therefore, it is a good idea to use a similar approach if a feature team requires support for acquiring some specific know-how. In that case the respective mentor works with this team for as long as it is necessary to transfer the knowledge.

It often simplifies the job of these transient team members if they travel to the site of the feature team they're currently supporting. But, this is not always required especially not if the feature team is dispersed because such a team has by definition no common site. Therefore, for the support of a dispersed feature team—a team that is distributed in itself—the transient team members stay either virtually in contact with the people they are supporting or they travel to the sites where these people reside.

[1] More on feature teams see Chapter 18: *Feature Teams—Distributed and Dispersed*.

19.4 Product Owner

Each feature team has to be made aware of feature prioritization and has to know whom to ask if there are problems in understanding the specifics of a feature. The agile manifesto requires in one of its principles that:

> Business people and developers must work together daily throughout the project.

In other words: a feature team needs to collaborate with somebody representing the customer. To simplify matters, this representative is often labeled as *the customer*, or as in XP the *on-site-customer*. Most often this term is misleading, because this representative is seldom a customer himself, but somebody providing the business perspective in terms of the customer. For example, some systems need to serve various (competitive) customers who can't agree on the requirements. The Scrum term *product owner* is widely used for differentiating the real customer from the representative who collaborates closely with the feature team.

The product owner's task is to clarify the different requirements of the various customers and to decide on priorities. Thus, the product owner needs to know the business domain of the customer in detail and has to have a good communication channel to the (different) customers. The product owner is caught between two stools—he needs to support the feature team regarding the business knowledge and to involve the (real) customers to decide on priorities. This makes fulfilling the product owner's responsibilities very exhausting.

Most often product owners are recruited from different areas or departments, such as: marketing, support, product management, sales, or business analysis. If the end users of the system under development are developers, of course a developer is also an excellent candidate for the product owner.

19.4.1 Team of Product Owners

Most often there is a one-to-one mapping between product owner and feature team. The exception is if the system is rather simple and/or the feature team has built a similar system in the past, then a product owner can support several feature teams. For the norm, large global projects have a product owner for each and every feature team. These product owners need to synchronize the prioritization of the features so that the resulting system provides a surplus with every iteration. Ignoring the synchronization leads most likely to a system in which the contained features do not fit together or even contradict each other.

Thus, in a large global project the product owners have to work together as a (virtual) team. This team is claimed to be virtual, because the respective feature team is the "home" for every product owner. Moreover, most often the collaboration of the product owners happens virtually, because the product owners are situated at different sites.

19.4.2 Lead Product Owner

Establishing a (virtual) team of product owners creates the risks that:

- Every product owner pushes his or her own features and ignores the overall business functionality of the system.
- Decisions on priorities are delayed, because the team can't come to an agreement.
- Different product owners contact the same customer with similar questions, which might upset that customer.

The core of these risks is that too many cooks spoil the broth. The more product owners belong to that team the more likely it is that the business perspectives differ. Therefore, the team of product owners has to be steered by a *lead product owner*, who mediates in case of discrepancy.

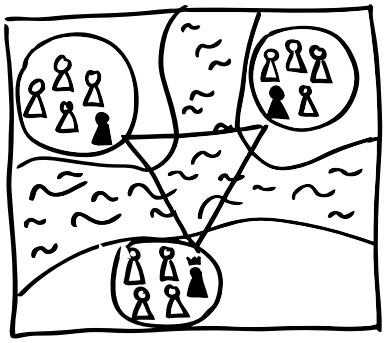

Fig. 19.1 Virtual team of product owners (in whole black) with lead product owner (symbolized with crown)

The lead product owner is the key contact to the (real) customers and collects their key ideas. The responsibility of the lead product owner is to spread these key ideas. Moreover, the lead product owner requires the input of the team of product owners in order to decide on business priorities.

Depending on the complexity of the system as well as of the project (based on size, degree of distribution and the like) the lead product owner might be able to additionally support one of the feature teams. That is next to the responsibility of leading the team of product owners, this person takes the role of an "ordinary" product owner. Yet, in most cases fulfilling the role of the lead product owner will be the unique (full-time) task of that person.

19.4.3 Collaborating with Both: Customers and Feature Team

The lead product owner is the key contact to the customers. Yet, for supporting their feature teams adequately—getting feedback and clarifying possible misunderstandings—also the product owners need to work together with the customers.

Preferably, every product owner is co-located with the feature team he's supporting. It's a rule of thumb that the more complex the business domain is the nearer the product owner has to be to the feature team. As a consequence, the product owner needs to work together with the customers virtually most of the time. At other times, he has to travel to the customers sites.

Obviously, if the feature team the product owner is supporting is not co-located but dispersed, he can't be co-located either but needs to travel to the various sites. Often it helps if the product owner is then at least situated at a site where a few members of the dispersed feature team reside. This way he will always be aware of the current situation. Whenever the product owner can't be close to his feature team the conversation between product owner and feature team has to be enriched by utilizing all kinds of communication media. The most important thing though is to try to reduce the times when the product owner is not close to "his" feature team.

However, it should be clear that there is no difference between an onshore and an offshore team. This is stressed, because sometimes it is assumed that offshore teams don't need a co-located product owner. A co-located product owner can ensure in the best way possible the growth of business value in the system. And delivery of business value is what agile is about.

If the assigned product owner can't be co-located with the feature team he is supporting—a different product owner is required. Global projects use often a shadowing concept to educate product owners at every site involved. This shadowing concept relies on experienced product owners at other sites who act as mentors for the inexperienced ones.

19.5 Coach—Also Known as Scrum-Master

The coach (Scrum term: scrum-master) ensures the agile process is supporting the feature team in the best way possible. If the process is hindering more than helping, the coach will work together with the team to improve it. Or whenever impediments occur that hinder the feature team on making progress, the coach will smooth this impediment out. An example would be escalating problems to the right people.

This doesn't necessarily mean that the coach is doing all this operatively himself, yet, he reminds the team that if something isn't working it needs to be changed for the better and asks every team member to responsibly do so. In this way, the coach acts more as a vivid reminder for the team. It should be the goal for every coach to become superfluous—and although I have never seen this happening in reality—this mindset helps the team to understand self-organization better.

In large global projects there is always a one-to-one mapping between coach and feature team. For every agile team there is only one exception to this rule: the team is perfectly self-organized and as a consequence the coach is superfluous. In all other circumstances every feature team is supported by one coach. If there is no experienced coach available, at least one of the team members has to grow into that role. The coaches of the different feature teams work together for ensuring the

overall agility of the project. They help feature teams to benefit from one another by transferring learnings, and good practices from one feature team to the others.

I find it important that the coach is actually a member of the team and not an outsider to the team for example by being one level higher up in the hierarchy than the team. To ensure this, the coach should support the team additionally as a developer or tester. However, the coach might not always be able to fulfill his responsibility as a regular team member—this depends on how well the team jells and how good it is in self-organization.

It should be obvious that the coach could fulfill his role best if he is co-located with the feature team he is supporting. Of course, for dispersed and thus not co-located feature teams the coach needs to communicate and collaborate with team members in different ways: by phone, e-mail and also by traveling. Moreover, similarly to the product owner, also the coach should be situated at one of the dispersed feature team's sites (and not a third site) in order to know what the team is struggling with.

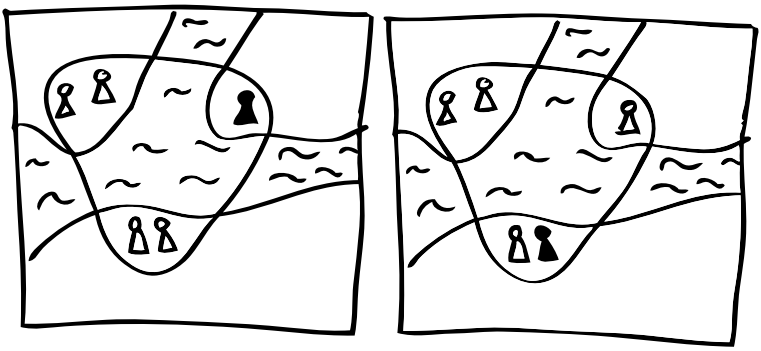

Fig. 19.2 Coach (in whole black) should be located at one of the dispersed team's site (right figure) and not at a third site (left figure)

19.6 Architect and Architecture

Small co-located agile teams typically take the responsibility for the architecture altogether. While concentrating on business features, close collaboration enables such a team to additionally ensure that the system is technically sound. This technical soundness is also called conceptual integrity and is defined by Fred Brooks as follows:

> It is better to have a system omit certain anomalous features and improvements, but to reflect one set of design ideas, than to have one that contains many good but independent and uncoordinated ideas. [1]

Conceptual integrity is according to Brooks the most important consideration in system design and the basis for simplicity and straightforwardness, which is in large global teams the main responsibility of an architect. For some projects it is sufficient for an experienced developer to additionally take the role of the architect. Yet in a large and global project you will need an experienced chief architect to ensure conceptual integrity across the entire project.

The need for architects depends on the technology, the technical know-how of the project members, the complexity and the size of the project. One architect advising in all technical decisions is enough for many global projects. This architect brings technical dependencies of features to the awareness of the business site which help the (lead) product owner to make the right prioritization decisions. In some cases the technical complexity of the system or the lack of technical know-how requires every feature team to get the support of one architect. Very often though, this support is only needed for the start of the project till the knowledge is built up. Then the responsibility of the architect morphs to regular development with only occasional pure architectural support. Finally, some projects benefit from a team of architects. In such a setting feature teams will get full-time support of an architect only for limited period of time, for example for the duration of one iteration before the architect moves on to support the next feature team.

Supporting a feature team doesn't mean that the architect provides concepts or documents yet it means he assists in implementing the features through actual coding.

19.6.1 Chief Architect

As soon as more than one architect supports the project it becomes crucial that the architects collaborate closely—otherwise conceptual integrity isn't guaranteed. Similarly to the virtual team of product owners, also the architects benefit from somebody—the chief architect—leading the group.

It is the chief architect who ensures that the big picture is communicated and understood well. He will act as the key contact for the business side and will keep the memory of key ideas alive [2]. Moreover, the chief architect spreads these key ideas and makes this way certain that more and more people gain the same understanding of the system. Without a chief architect feature teams (with the support of their architect) tend to suboptimize towards their own targets and lose sight of the total effect on the entire system [1]. An architect should never come up with concepts driven by self-fulfillment. Thus, it is the chief architect's task to convince all architects that they are providing a *service* for the feature teams that supports the development of business features.

Although the name of the role—chief architect—might imply that this person is dictating architectural decisions, this is far from being true. Yet, on the other hand using democracy like majority decision is also not a good advisor for making key decisions. Instead the guiding concept should be *nemawashi* defined by the Toyota Way as:

> Make decisions slowly by consensus, thoroughly considering all options; implement rapidly. [3]

Taking this concept into account requires the chief architect to ensure that everyone gets heard, different views are evaluated and everybody is this way involved in decision making with the consequence that the final decision is backed.

19.7 Project Manager

For small co-located agile projects the classical tasks of a project manager are mainly performed by the product owner (a few are left for the coach). Yet, in a large and global setting the burden for the product owner(s) and coach(s) is already so high that these persons can't additionally take care of the organizational stuff. A critical responsibility belonging to that area is politics. A project can be in optimal shape but still be killed by ignoring political issues (or lobbying)—both externally and internally.

If a feature team requires specific resources, has difficulties accessing its product owner or customer, or if a project member wants to change sites or move to another department—the project manager can support effectively because his network inside and outside the company is typically more powerful (than for example that of a coach).

Often the responsibility for the budget lies also in the hands of the project manager. However, this requires close collaboration with the (lead) product owner.

Basically, it is the project manager who enables the entire project team by removing all impediments that can't be removed by the feature team members themselves.

If the development of your global project is spread over several locations, it is unimportant at which location the project manager actually resides. He will have to travel to all the different locations anyway. However, if development is situated at one location and project management at a second one then that definition of project management clearly does not match the project manager role bed here. In such a situation I recommend project management to move to the development site for the duration of the project.

19.8 Key Roles Support Their Teams Directly

All key roles—especially coach, product owner and architect (if there is one per feature team)—should be co-located with their feature team. Actually, they belong to the team. Some organizations assign key roles to project members located at the site where the headquarters resides. However, this is by no means supportive. On the one hand, such a setting implies the key roles are more controlling than participatively serving the feature team and on the other hand, it feels like the headquarters mistrusts the other sites of being able to self-organize. Additionally, proximity makes collaboration more efficient.

Yet, if the respective feature team is dispersed there is no specific site where key roles should be located. The only rule to follow in such a setting is that key roles should reside at one of the dispersed feature team's site and not be solely located at yet another one. Still, the relation between efficient collaboration and proximity holds also true in a dispersed setting. Therefore, it is crucial for everyone playing a key role to travel frequently to all sites involved and to stay virtually in touch with the remote sites.

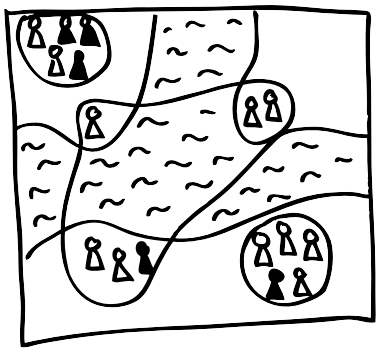

Fig. 19.3 Key roles (in whole black) should be co-located with their feature team no matter if the team is co-located or dispersed

19.9 Conclusions

Large global projects require special attention to roles and responsibilities. Each feature team should be structured in a way that all necessary knowledge and skills are either available within—or can be acquired by the team. Sometimes, for example during an educational period it might be necessary that a feature team requires the support from a mentor over a limited timeframe.

Every feature team is steered business-wise by a product owner. The product owner decides on the priorities of the features this team is working on and clarifies possible misunderstandings. In order to do so the product owner stays in contact to the (real) customer and to his peers. This team of product owners with the support of a lead product owner ensures that every iteration results in the delivery of a meaningful business value.

Every feature team is supported process-wise by the coach. Thus the coach ensures adherence to the agile process and if the process is inadequate he motivates the team to change the process for the better.

The focus on business value might lead into ignoring technical aspects. The architect brings these technical aspects to the awareness of the product owners. He explains technical dependencies between features which might influence their priorities. Depending on the project (in terms of size, content, technical complexity) every feature team might need the support of an architect or at the other extreme one architect might be sufficient for serving the whole project. If a project requires

more than one architect, the chief architect will ensure the synchronization of—and agreement on technical decision by all architects.

Contrariwise to small co-located agile teams, large global ones require a project manager making certain that the whole project runs smoothly organizational wise. The major concern of the project manager is to get the backing for the project both inside and outside the organization. This requires the project manager to deal a lot with politics.

Unless a feature team is dispersed, all key roles should be located at the same site as the feature team. In order to support a dispersed feature team the key roles need to be located at one of the team's sites, travel frequently to the various sites involved, and communicate virtually while away.

References

1. Brooks, F. P. Jr. (1995). *The mythical man-nonth: Essays on software engineering* (20th anniv. ed.). Reading: Addison-Wesley (Quoted from p. 42 and p. 44).
2. Cockburn, A. (2006). *Agile software development: the cooperative game* (2nd ed.). Reading: Addison-Wesley.
3. Liker, J. K. (2004). *The Toyota way. 14 management principles from the world's greatest manufacturer*. New York: McGraw-Hill (Quoted from p. 241).

Further Reading

4. http://www.agilemanifesto.org.
5. Eckstein, J. (2004). *Agile software development in the large*. Cambridge: Dorset House.
6. Eckstein, J. (2010). *Agile software development with distributed teams*. Cambridge: Dorset House.