# Agile infrastructure and operations: how infra-gile are you?

Patrick Debois
*Supporting Open Source*
*Patrick.Debois@sos.be*

## Abstract

*Some have described Agile and Infrastructure as an oxymoron: they just don't fit together. During one year we have focused on using agile techniques in three different infrastructure related projects. From a unique infrastructural point of view, we will show that the term 'agile infrastructure' consists of multiple layers. To become effective, each layer needs to be addressed.*

## 1. Case 1: Datacenter Migration

### 1.1. Application junk yard

Our first project was the construction of a new datacenter infrastructure. The production environment contained several unfinished, non-production ready applications. A lot of these applications had been forced onto the datacenter infrastructure: the development of a new application always exceeded the deadlines.

As usual in large enterprises, development, infrastructure and operations were separate groups. Development and infrastructure would work in isolation on a project and would integrate just before the political deadline to present the application to operations. Then there was no time left to fix things.

A new datacenter would allow to cleanup the situation by migrating the old applications to a new more controlled environment.

### 1.2. A new infrastructure, a new hope

A group of architects was assigned to describe the requirements of this new datacenter. They focused on the new design and came up with current state of the art improvements: new development frameworks, new application servers, scalable infrastructure would buy new and more powerful machines. For managing the datacenter, ITIL would be introduced as a process.

They would only release their datacenter to new applications after every system or process was completely detailed. Rolling out an application would be just a simple case of following the new guidelines. They felt no need to talk to the different projects, as their environment would be generic to all applications.

New applications were to go directly onto the new datacenter. Because the task of describing the new datacenter was taking longer then foreseen, new projects were delayed instead of advancing. The president of the company became nervous and expressed it like this: "I don't care if it is not finished but there has to be something and then you can continue to make things better afterwards." A small taskforce would get things going.

### 1.3. Change of mindset

While investigating application-testing criteria, the taskforce came across several agile inspired concepts like test-driven development and Scrum. Most of the literature involved the development process, but they could only control the infrastructure process.

Scrum as a project methodology was not necessarily related to development. There seemed to be a perfect match between the idea of iterative design and the demand of the president: every sprint you would have new working release and it would constantly improve. They would experiment to see that agile concepts would indeed work for infrastructure projects.

### 1.4. Applications as customers

Instead of building a generic datacenter, the taskforce contacted each project leader to get involved in the project meetings. Each application was seen as a customer for the datacenter. This way, they started to compile their backlog, with different priorities assigned. The interaction also allowed them to better understand the needs of their projects, which were their customers. Projects became aware of the shared nature of the infrastructure and better understood the problems of scheduling. Also the taskforce could point out several non-functional requirements like security, performance, logging, monitoring that had not been

taking into account. An initial list was compiled and the priorities for the next two weeks were discussed: the content of their first sprint.

### 1.5. A minimal working environment

The main focus of the taskforce was to start building a temporary environment that could host the new applications, until the new datacenter was ready. A minimal working environment would typically require several weeks as lot of different groups had to interact: servers, networking, monitoring, storage and new hardware was still on order.

One by one, they started to overcome the dependencies. Missing DNS servers were compensated by using host files. Servers instead of routers did routing. The use of VLAN tagging on interfaces allowed them to overcome the lack of network ports. Load balancing and SSL Termination were performed in software instead of hardware. Internal Disks were used for data to make up the missing storage arrays. These solutions were not final, and would be replaced once there definitive solution became available. But at least it was a working environment.

### 1.5. Deploy often

The delivery of the first sprint would be tested by a successful deployment of the first application. This was again a disaster: several configuration files were missing, the developers were working on another version of the database, and there was no monitoring. These were typical discussions in the past. The difference was that now because they still had time before the actual deadline, they could try the deployment again. In the mean while the infrastructure would also be improved in parallel. After three test deployments the benefits were clear: a lot less integration problems. The application went live and even during production this improvement process continued. Every release they would improve both the software and the infrastructure.

### 1.6. Service Levels Agreements

The incremental approach resulted in an interesting side effect: In the past when the application would go live, a Service Level Agreement would be negotiated between the customer and the external partner. Because this was the document that described when penalties should be paid, this document often provoked huge discussions. With the new incremental approach, when was the application now finished? The SLA managers had not been in the discussion loop. Even with significant improvements the operations would not sign of the acceptance, as it was unclear when the application or infrastructure was finished.

## 2. Case 2: Disaster Recovery

### 2.1. Technical debt

Our next case was focused on disaster recovery: a company had experienced several outages and money was lost. Infrastructural updates had been postponed because the impact was not predictable on the uncontrolled environment, resulting in a large technical debt. Again the idea was that a migration to a new hardware platform would solve these problems. The manager already had good experiences with Scrum used by a development group and decided that the infrastructure group would use it.

### 2.2. Group versus Team

The group consisted of five persons where each person had its own specific expertise: networking/security, desktop/office, servers and storage, application and middleware. Each person compiled a list of items that he felt were necessary to improve the situation. They would call this list their product backlog. Their manager put priorities on the list and the first sprint was defined by the question: what can you finish within two weeks? Group estimation did not work well, because often only one person could actually tell something about a certain task.

### 2.3. Tasks and tickets: a deadly cocktail

During the first sprint it became clear that incidents would often overrule the planning of a team member. For every incident there was a ticket logged. A new task was created on the Scrum-board called ‘Legacy’. This would host all the emerging incident tasks. After the first sprint the board was full of this small tasks. Analyzing the list of tickets would give a better understanding of how much time should be allocated for incidents and how much time was left for the improvement project.

The list of tickets turned out to be a mix of service requests, problems and incidents. Instead of planning these requests they were just thrown in the list of incidents. Projects took advantage of this and introduced new work or last minute changes as an incident, hard to be refused. Also team members took advantage of this, they would relate their own interests to incident, in a way to work on more interesting tasks.

## 2.4. Priority rules

To get a better overview, the tickets were reviewed and split in incident, improvement and project. In order to give focus to the group, the manager decided that the correct order would be, incidents first, secondly the improvement list and then helping projects. The reasoning was that once the improvement list was done, helping projects would be easy.

When project managers asked a status on their ticket, they were answered that they were not on the list for the next two weeks. Some started to ask exceptions to the manager, which often had to comply with their requests. Others changed tactics and came in person to the infrastructure team and used social pressure. On the Scrum board priorities changed every day, the project that shouted the hardest would get priority number one and focus was lost.

## 2.5. Trying to please more masters

Instead of deciding on the priorities in the next sprint-planning meeting, the manager invited all the project managers. When they saw the extensive backlog they started to understand why they were seeing the delays. They only had knowledge of their own project and not on the global list. In the infrastructure group everything came together.

The product backlog was reordered in a way to please every project. Eventually, this required the presence of a general manager, otherwise all projects including the infrastructure improvement project, would schedule themselves as the most important project. Now the team had one product owner instead of many.

To increase the resources, the infrastructural team was expanded with resources from the test and middleware team. These extra resources already had affinity with infrastructure and were used to work in project mode. The project tasks moved to the new team members, incidents and improvements stayed with the original team members.

## 2.8. We don't need another graveyard

During the next sprints, the work of the improvements did not advance, as these team members were still continuously overruled by incidents. The project related tasks, became dependent on the new environment. It would only be ordered if the design was completely finished and all details had been described.

In contrast, the project oriented people used virtualization on servers with spare capacity to accommodate the new applications. With every sprint they would improve a small temporary environment based upon the new application needs.

## 2.9. Agile versus Waterfall

Most projects would use Prince2 as their management style. One project used Scrum for their development. The Scrum managed project only asked what would change the next sprint and incorporated the changes into their product backlog.

The more traditional oriented project managers complained about the new incremental way of working: their developers had to constantly adapt their code according to the changes in the environment, these changes were not foreseen and resulted in additional work. They did not have easy ways to change their code and did not practice test driven development. They took the ever changing environment as a sign of incompetence even when it was improving over time.

## 2.10. Not everybody sees the big picture

The operational people were still present in the daily Scrum but they were becoming less and less interested: their day consisted of closing incidents and not helping projects. They did not agree that the project priorities were their priorities: they would install new servers/versions even if it was not the overall priority. They assumed their job was improving the infrastructure not on creating business value by new projects. Why did the improvements never made it to the top of the list?

This permanent discussion about priorities was increasing the tension between the infrastructural/operational manager and the overall manager. People really started to suffer from their small war. And even worse the discussion became personal and the operational manager resigned.

The overall manager introduced again the roles and responsibilities game: everyone in its own part and let me control the flow who does what. And stop this communication culture. The whole experience was buried within a month. Eventually they set up a new infrastructure and within the first weeks it was showing the same instabilities as the old platform. No use for doing a disaster recovery project if your problem lies beyond the technical problem.

## 3. Case 3: Application Server Upgrade

### 3.1. Shared software as part of infrastructure

In our third case a company that had been working several years in Scrum project mode was migrating to a new version of the application server. The developers had identified this as the solution for the performance problems. A task easy enough as running a wizard with next, next , next was taking a lot longer then foreseen: the monitoring system needed to be adapted, all security needed to be tested, and what about the high availability. This simple application server depended on a lot of shared infrastructural components. A small audit was started to enlist further improvements.

### 3.2. Production and test: similar?

Doing the upgrade in the development and test environment had been easy. The production environment was a lot more complex with clustering and management tools then had been installed in the test environment. The infrastructure had been set up at the early stages of the agile process within the company. It had been designed as a generic infrastructure and without knowing the real applications it had suffered from a featuritis: it included every bell and whistle that could be included because at the moment of build the requirements were not clear to the infrastructure group as both parties worked in two separate groups.

### 3.3. The whole stack please

Another difference was that the test and development were running in virtual environments, using light configurations of the application server to overcome the lack of machine resources. They used a newer version of the operating system. It would make sense to upgrade the OS together with the Application Server. Upgrading the OS would mean new monitoring and backup agents. But at least now the same JVM, JDBC drivers and database version could be used.

By extending the existing deployment scripts, the whole installation of a new virtual machine including OS, JVM, Application server, JDBC Drivers, application could be recreated. This would allow the inclusion of patches on each level to be tested every iteration. Integrated patching would prevent production surprises. During an interview on how to improve things, everybody remembered the iteration with the seven hotfixes and did not want this to repeat that nightmare.

### 3.4. Tracking changes and their impact

The problem with these integration scripts is that they need to be maintained, if the infrastructure is dedicated for the project then this does not pose so much of a problem. In a production environment often network, storage, monitoring and security are shared. And this means that all projects have to take note and investigate the impact of these changes. Often this had given problems, things had been changed in the production due to other projects and the prediction of the test environment had not been proved useful.

A production architect could track these changes as part of his job description. The operational manager became very interested because this would be a kind of gatekeeper. The agile project group understood very well the idea but they had abandoned the architect-only idea already a long time. Several discussions later it became clear that the candidate for this job needed to be neutral for both operations and project and that could only happen he worked aside the operational and project manager. Nobody from the existing group would be found neutral enough. Instead of assigning the responsibility to one person , the task was assigned to the infrastructural group within the project as a go between.

### 3.5. 80% project , 20% operational trap

Two infrastructure persons were assigned 80% to the project and 20% for other work. They performed the infrastructural work to support other team members but did not take part in the daily scrum meetings or did not work on part of the backlog, because their work just seem to fall off the list as miscellaneous. In reality they were somewhere in between project and operations mode, being driven by two different managers and they felt they always had to decide what needed priority. They also felt sorry for the project if they could not deliver what they had promised. The team felt that they were not committed enough sometimes. So they worked hard to try to please both groups.

While discussing the product backlog, their work was not listed as it did not create any direct business value. The customers mainly expressed new functionality and not infrastructural work. The next sprint planning meeting their work would be put on a separate product backlog: the application/developers group would become their customers for their user stories. On the same backlog they could suggest tasks to improve the infrastructure making their work visible to the customer. These tasks would include an estimate of the value that would be lost if things were not improved.

### 3.6. Joined design and estimates

During the migration the developers had done most of the initial design of the migration and that while they were comfortable with the new technologies, the infrastructural group came in the end. They had to learn the new versions under pressure of deadlines and commitments taken by the development project group. While these deadlines were a group commitment, the infrastructural people had not been consulted during the time of estimation and design. Once the task list grew developers started to see that there was a lot more than they could think of. This was not to gain time, just that they were not aware of all this dependencies. In the next design meeting the infrastructural people would be called in the meeting to have their ideas. Not anyone has an overview on everything but making it a group effort would compensate for this.

### 3.7. Who's the boss?

After making their work visible it became clear they were becoming crucial for a good flow to production. The operation manager was afraid of loosing the additional theoretical 20%, it was hard to find people with an affinity in both infrastructure and applications. Infrastructure people often saw applications as a nuisance, which is strange as the applications of bring in the value for the company. But changes by applications and projects increase the instability of the environment. The discussion between both managers was escalated and priorities needed to be addressed at the global company level and not on the local levels of one project or operations.

### 3.8. Flow is more important then backlogs

While solving all the problems at the technical and project level this company was still struggling with the operations and project conflict. Once they take this hurdle they will reach a new level of agile infrastructure. Instead of focusing on the local optimizations, they are now investigating new concepts like Kanban in order to get the flow going through the whole company. This would allow them to bring both their projects and operations in a rhythm that works.

## 4. Observed Patterns

In all three cases we discovered several patterns. We have grouped these patterns in three categories: Technical, Project and Operations. Technical relates to hardware and software used in the environment. Project is about the process that introduces the changes into the environment. Operations is the process of keeping the environment working. We consider this the three layers that Agile infrastructure relates to.

### 4.1. Technical

- A 'technical' migration is seen as the solution for years of problems.
- Technical migrations are enablers for the agile process and often use virtualization and automation techniques, converting the static infrastructure in agile enabled infrastructure.
- The use of these techniques requires new technical skills to be acquired.
- The infrastructure toolkit is not yet as extensive as the developer's toolkit f.i. for refactoring, so it is important to have some guidance to avoid a lot of refactoring for the basis architecture.
- Replacing all the hardware will not solve the problem. It will create a new 'uncontrolled' environment if only the platform is replaced.
- Non-agile infrastructure tends to grow old because changes are hard to execute in these environments. This can be seen as technical debt.
- A small environment can create enough space for a conversion process to take place. Often there is enough 'loose' hardware available to sparkle a new usage.
- Continuous build servers, test servers, code coverage tools and versioning servers are also part of the infrastructure and the infrastructural people can take care of these too. This will allow them to build up affinity with the developers' environment.
- Even scripts used by the infrastructure people benefit from a versioning system like CVS.

### 4.2. Project

- Technical skills need to be complemented with an agile mindset. Similar to the technical, to speed things up have experienced agile project managers guide the process.
- Development and infrastructure need to be seen as whole and not two separate projects. This is especially difficult in large enterprises where both belong to different entities.
- Design of the developments needs to include infrastructural and operational design to easy integration.

- Otherwise each project will create a local optimum but the company optimum is never achieved; Service Level Agreements (SLA's) express requirements that need to be included in the backlog.
- Every change will eventually require a management buy-in to allow it to persist.

### 4.3. Operations

- After the project finishes the operational group will benefit from the test and integration environment: In contrast with project mode it is now the environment that changes due to security patches or new versions.
- Infrastructural people often have strong bounds with the operational part of the company.
- Operations do not work in project mode and when including staff or dependencies will make you project unpredictable. Estimations will not work, as incidents are unpredictable.
- People between project and operations will loose focus and in the worst scenario can use this to never having to finish tasks.
- Additional staffing in the operational group is often not possible because they are perceived as not adding value to the company. When they are part of your deployment process their availability is crucial.
- Projects can be overproducing for the operations group: they are trying to optimize their production process but often do not consider the flow of the result through the whole company and operations become the bottleneck. Operations are traditionally the place were multiple projects come together and each project might not have a view on other projects. Making this clear is crucial and having agreement among project managers on priorities is hard to achieve from a support only perspective.
- Operations know very well that changes introduce incidents. Therefore they think their job is to minimize change to the production environment. Still they serve the same customer the project does. Therefore the project should have a view on the demands the customer puts on the operations.
- Changing your project process will have an impact on the way SLA's are discussed.

### 4.4 Conclusion

Successful introduction of Agile infrastructure consists of addressing both the technical, project and operations aspects. The technical part is the easiest, tools are becoming mature and integrated and this what IT-people most easily understand. Building on this technical foundation, the infrastructural work can now be integrated in the project. The hardest part is when integrating with the unpredictable process of operations and crossing the non-project boundary and sharing operational resources with projects.

## 5. Further Reading


[1] Pritchett, Dan, *Operational Manageability lessons learned from eBay*, 20 September 2007, http://www.infoq.com/news/2007/09/operational-mangeability

[2] Hendrickson, Elisabeth, *You're Kidding. It does WHAT in Production !?!*, 22 May 2007 , http://video.google.com/videoplay?docid=6277615041054958810

[3] Wilson, Scot, *Agile Operations* , 27 September 2007 , http://www.indigomoonsystems.com/status/status.php?/archives/259-Agile-operations.html

[4] Anderson, David J., *Operations Review* , 20 September 2007, http://www.agilemanagement.net/Articles/Weblog/OperationsReview-2.html

[5] Nicolette, Dave, *Managing non-functional requirements and enterprise standards*, 30 December 2007 http://dnicolet1.tripod.com/agile/index.blog?entry_id=1776642

[6]Berteig, Mishkin, *Agile Infrastructure Projects*, 28 May 2005 http://www.agileadvice.com/2005/09/28/agilemanagement/agile-infrastructure-projects-lessons-learned/

[7] Gibbs, Ed , *Agile With Infrastructure Projects* , 14 January 2008, http://edgibbs.com/2008/01/14/agile-with-infrastructure-projects http://times.usefulinc.com/2006/06/17-agile-infrastructure ,

[8] Coon, Mike, *Agile Infrastructure*, 10 November 2007 , http://blog.mikecoon.net/2007/11/10/agile-infrastructure.aspx

[9] Agile 2006 Conference, *Agile Infrastructure*, 2006, http://agile2006.stikipad.com/public/show/AgileInfrastructure

[10] Terry, Hamilton, *Agile for Infrastructure*, 2007, https://www-927.ibm.com/ibm/cas/archives/2007/workshops/workshop22.shtml

[11] Schiel, Jim, *Scrum and an 'operational' team*, Scrumdevelopment mailing-list, 6 March 2008, http://groups.yahoo.com/group/scrumdevelopment/message/27881

[12] Debois, Patrick, *Agile Infrastructure Operations* , http://www.jedi.be/agille-infrastructure