DELIVERABLE

Project Acronym: **CAP4Access**

Grant Agreement number: **612096**

Project Title: **Collective Awareness Platforms for Improving Accessibility in European Cities & Regions**

D3.1 Collaboration Infrastructure

Authors:

Hans Voss Fraunhofer IAIS

with contributions from all partners

Project co-funded by the European Commission within **FP7-ICT-2013-10**		
Dissemination Level		
PU	Public	■
PP	Restricted to other programme participants (including the Commission Services	
RE	Restricted to a group specified by the consortium (including the Commission Services	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Status, Abstract, Keywords, Statement of Originality

Date of delivery	Contractual:	31/03/2014	Actual:	02/04/2014
Status	final ☒ /draft ☐	Reviewed by: Karl-Heinz Sylla (IAIS), Karsten Gareis (Empirica), Christian Voigt (ZSI)		

Abstract (for dissemination)	CAP4Access applies Agile Development methods for the development of tools as well as for planning, coordination and pilot site activities. This enables the RTD to be driven by the needs and preferences of the beneficiaries and user groups rather than following a "waterfall model" sequential process as it is often used for technical development projects. The collaboration infrastructure is composed of GoToMeeting and Atlassian tools for Wikis (Confluence) and issue tracking (JIRA). The agile processes are divided into four domains with close correspondence to work packages: Tools, Pilot Sites, Communication, and Coordination.
Keywords	Participatory research, Scrum, Kanban, Agile Development, tele conferencing

Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Table of Content

Table of Content **3**
Executive Summary **4**
1 Introduction **5**
2 Atlassian Tools **8**
3 Agile Development within CAP4Access **10**
3.1 Activities and Communication 10
3.2 Scrum or Kanban? 11
4 Summary **12**
5 References **13**

List of Exhibits

Exhibit 1: Atlassian Confluence as used for collaboration in CAP4Access 9
Exhibit 2: Atlassian JIRA as used for collaboration in CAP4Access 9

Executive Summary

The project will apply Agile Development methods for the development of the CAP4Access tools as well as for planning, coordination and pilot site activities. This will enable the RTD to be driven by the needs and preferences of the beneficiaries and user groups rather than following a “waterfall model” sequential process as it is often used for technical development projects.

Technical support will be based on a communication and documentation infrastructure including GoToMeeting telecommunication and the Atlassian tools for Wikis (Confluence) and issue tracking (JIRA). 25 licenses for Confluence and JIRA have been obtained, installed and administered on a server at Fraunhofer IAIS, and users within the project have been registered and trained.

The agile processes within CAP4Access will be divided into four domains with close correspondence to work packages: Tools (software development, WP3), Pilot Sites (user engagement, WP2), Communication (WP5), and Coordination (WP6). WP4, which starts in the third year of the project lifetime, may be added later. Technically, these four domains will be assigned to individual Atlassian projects.

Communication between the four domains/projects will be facilitated by the weekly consortium’s teleconferences with all partners participating, access rights of all partners to all domains/ projects, by targeted online or physical meetings on demand between all or some domains/projects, and by individual subscriptions to parts of other domains/projects in order to receive alerts about relevant activities.

Regarding the concrete choice of the agile development methodology the project favours the Kanban approach, as it has been found to provide more flexibility to adapt to the demands of the individual four projects. In particular, it does not request a fixed length of sprints, and it has a higher focus on user needs and time-to-market.

1 Introduction

The project will apply Agile Development methods for the development of the CAP4Access tools. This will enable the RTD to be driven by the needs and preferences of the beneficiaries and user groups rather than following a "waterfall model" sequential process as it is often used for technical development projects.

A general discussion of agile development methods has already been provided in Deliverable 1.1 of this project. We want to start here with a commented list of the 12 principles[1] that were brought forward by the Agile Manifesto Group (in bold letters; our own comments in non-bold-letters):

- **Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.** Te ultimate and primary goal of a customer is not to fulfill an original plan but to get valuable software in the sense of his intentions. Early and frequent progress towards these intentions is the best way to achieve customer satisfaction.
- **Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.** Change based on customer feedback should be regarded as something positive, and be accommodated efficiently.
- **Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.** Ongoing and frequent delivery is satisfying for both customer and client, as it shows progress which can be evaluated early on. Heavy, lasting work which leads into blind alleys and huge frustration will be avoided.
- **Business people and developers must work together daily throughout the project.** The software customer is continuously ("daily") involved and committed to communicate with the developers. He is thus also co-responsible for the achieved results.
- **Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.** An agile development process creates an environment supporting the motivation of all people involved. The customer feels better and timely informed about the results, and has a better say within the process. Everyone feels better because of a continuous pipeline of delivered results and much more communication between all involved people and parties.
- **The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.** Next to face-to-face conversation comes video-conferencing, then telephone-conferencing. Written texts of documents can be helpful, but more in the long run as records for future management and maintenance of the work. Face-to-face or oral communication is much better suited for a better and direct understanding, and for eliciting tacit knowledge which is hard or impossible to print on paper.

[1] http://agilemanifesto.org/principles.html, retrieved 2014-03-20

- **Working software is the primary measure of progress. Most sprints are designed to produce some touchable, working piece of software.** So there is some progress all over time, albeit coming in small pieces every few weeks. One of the most positive effects is that all parties get a better feeling and idea about the status of the project with regard to the original or current expectations. There is no such a reunion after a 12 month period of hard developer's work, and then presenting results that are assessed as partial and unsatisfying by the customer.
- **Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.** Iterative development, with each step assigning a small piece of work to every developer, will quickly lead to a better assessment of each person's productivity and efficiency by every person itself, by the project's manager, and perhaps even by the customer with regard to the kind of things that can possibly be done within one sprint. The team as a whole should thus become much better able to assess the real workload of a new sprint in terms of available work time, e.g. 40 hours per week. Overnight sessions and hard working times under pressure should thus become the absolute exception, sustainable development at a durable pace becoming the rule.
- **Continuous attention to technical excellence and good design enhances agility.** Keeping the customer closely in the loop, communicating about results of sprints, and embracing change requests inevitably will lead to relatively early discussions of design decisions. In the customer's eye functionality and design tend to be inseparable. Also, general usability issues will be discussed not as an afterthought but relatively early. Design issues will be interwoven in many of the developing sprints, and care must be taken to assign sufficient time within sprints to address pending design decisions.
- **Simplicity – the art of maximizing the amount of work not done--is essential.** The logic here is that it is easier to add something to a simple thing than to take something away from a complex thing. Also, the rules and standards of the methodology should be kept simple. Dee Hock, former CEO of Visa International, commented on this: “Simple, clear purpose and principles give rise to complex, intelligent behavior. Complex rules and regulations give rise to simple behavior.”
- **The best architectures, requirements, and designs emerge from self-organizing teams.** The classical approach to software engineering starts with getting the requirements right, and then on the basis of these requirements makes a plan for the software development. This plan will only be meaningful if the requirements were really “right”. It is meanwhile common knowledge that this practically never happens in any more or less complex development task. The final product of any architect, let it be a software, a building, or a household product, typically develops out of a series or network of guesses, trials, failures, and improvements. Very rarely, or never, a novel product comes out of the blue, or manifests as the final step of a streamlined plan.
- **At regular intervals, the team reflects on how to become more effective**, then tunes and adjusts its behavior accordingly.

The primary focus on users and reactivity to their concerns, always and through all stages of the development process should become the guiding principle of our project.

Also described in D1.1 are agile process frameworks and different proposals for agile development methodologies. In particular, we introduced the Scrum and Kanban development methods, compared them, and argued for our preference for CAP4Access of the Kanban method, primarily due to the greater flexibility it provides.

In the following, we will briefly introduce the software development tools that Fraunhofer-IAIS has installed for the project according to plans and schedule. We will then describe how we intend to principally start the development and usage of these tools. We will only define principles of how to define the processes, and not detailed instructions, because it is highly important that the exact behaviors and activities will evolve gradually and be agreed between partners bottom-up, rather than be defined and deployed top-down.

2 Atlassian Tools

Agile Development without the support of technical tools would be very inconvenient, if not impossible for medium to large projects. Only using physical spreadsheets and cards would make documentation difficult, one would soon lose a clear view of the whole process, and it would cost time and nerves to retrieve something. Fortunately, there are a plenty of software tools available supporting all or parts of the processes. At the least, they all promote collaboration and variants of digital dashboards. An essential effect of using these tools is that the collaborative actions are "automatically" documented, activities and results become searchable, and it becomes easy to define and install one's personal view on the project without losing connections to the activities of others.

There exist some tools in the Open Source Domain[2], such as the quite advanced IceScrum and Agilefant tools, and commercial tools like Microsoft Visual Studio Team Foundation Server (TFS) or the widely-used agile tools from Atlassian[3]. The commercial tools tend to be superior in terms of usability and visual appeal. For CAP4Access, it was not planned to execute a comprehensive market survey and comparison of available tools. Rather it was expected to work with the same tools that the partner Fraunhofer IAIS had already experienced in the EC project BioVeL (FP7-283359). This was also reflected in the project proposal and contract.

Actually, the project's approval and the expectation to further invest into agile development and licenses of Atlassian tools led to a strategic decision by the director of Fraunhofer-IAIS to let more existing and future projects of the Institute apply agile methods and share the experience of common tools. Shortly after the start of CAP4Access the Institute ordered a bundle of 100 licenses. CAP4Access will use a share of 25 of these licenses. A valuable effect of this transaction was a considerable reduction in the costs of these shared licenses as compared to an independent package of 25 licenses for the project.

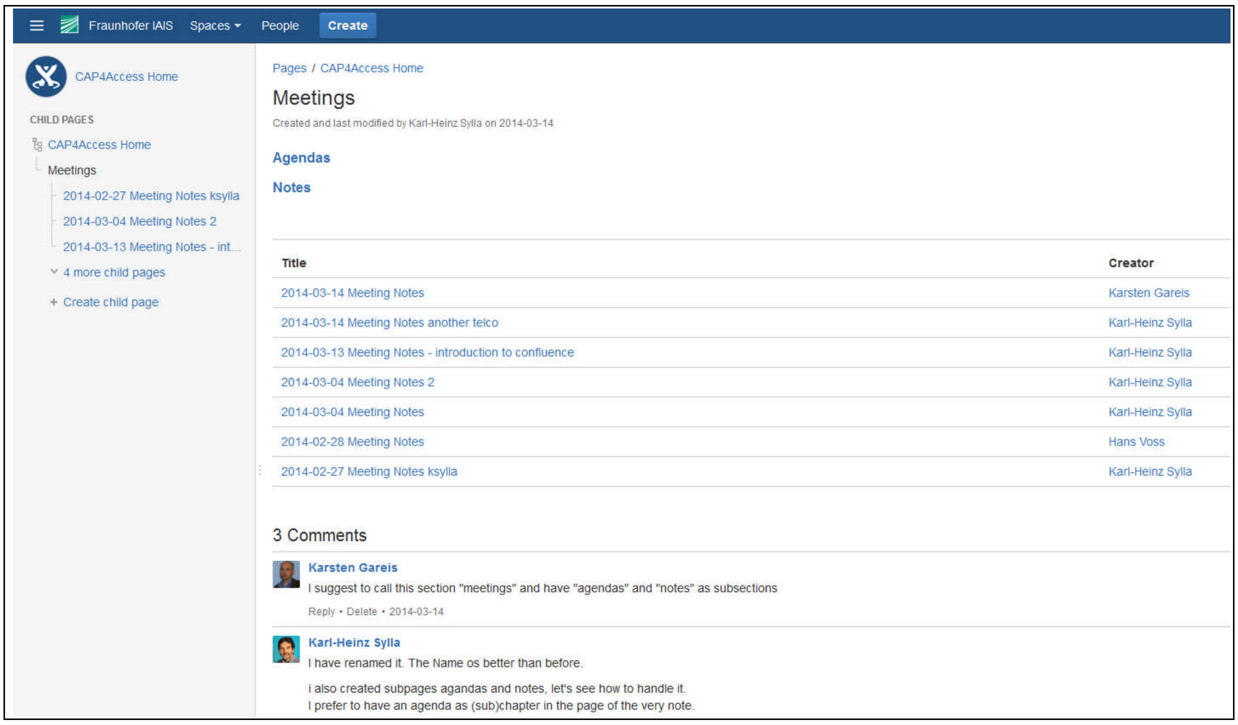
The planning within CAP4Access has so far been supported by weekly teleconferences using GoToMeeting. We have now started to introduce the Atlassian tools, which consist of an issue tracking system (JIRA, cf Fig. 4) and a wiki as a workspace (Confluence, cf Fig. 3), where Confluence is the document-oriented and JIRA the workflow- and action-oriented part. Confluence will primarily be used for the secure exchange of project information, project results, deliverables, meeting minutes, etc. It facilitates easy and timely inter-project communication. JIRA will be employed for defining workflows for the management of service development and deployment of work items. Technical and functional requirements and corresponding development steps are defined and controlled from here.

In general, there is an indefinite space of options how these tools are exactly applied for a concrete project. Our generic strategy how to evolve this is described in the next chapter.

[2] http://www.agile-tools.net/, http://sourceforge.net/directory/development/agile/os:windows/freshness:recently-updated/ retrieved: 2014-02-10

[3] **http://en.wikipedia.org/wiki/Atlassian** retrieved: 2014-02-10

Exhibit 1: Atlassian Confluence as used for collaboration in CAP4Access



Pages / CAP4Access Home

Meetings

Created and last modified by Karl-Heinz Sylla on 2014-03-14

Agendas

Notes

Title	Creator
2014-03-14 Meeting Notes	Karsten Gareis
2014-03-14 Meeting Notes another telco	Karl-Heinz Sylla
2014-03-13 Meeting Notes - introduction to confluence	Karl-Heinz Sylla
2014-03-04 Meeting Notes 2	Karl-Heinz Sylla
2014-03-04 Meeting Notes	Karl-Heinz Sylla
2014-02-28 Meeting Notes	Hans Voss
2014-02-27 Meeting Notes ksylla	Karl-Heinz Sylla

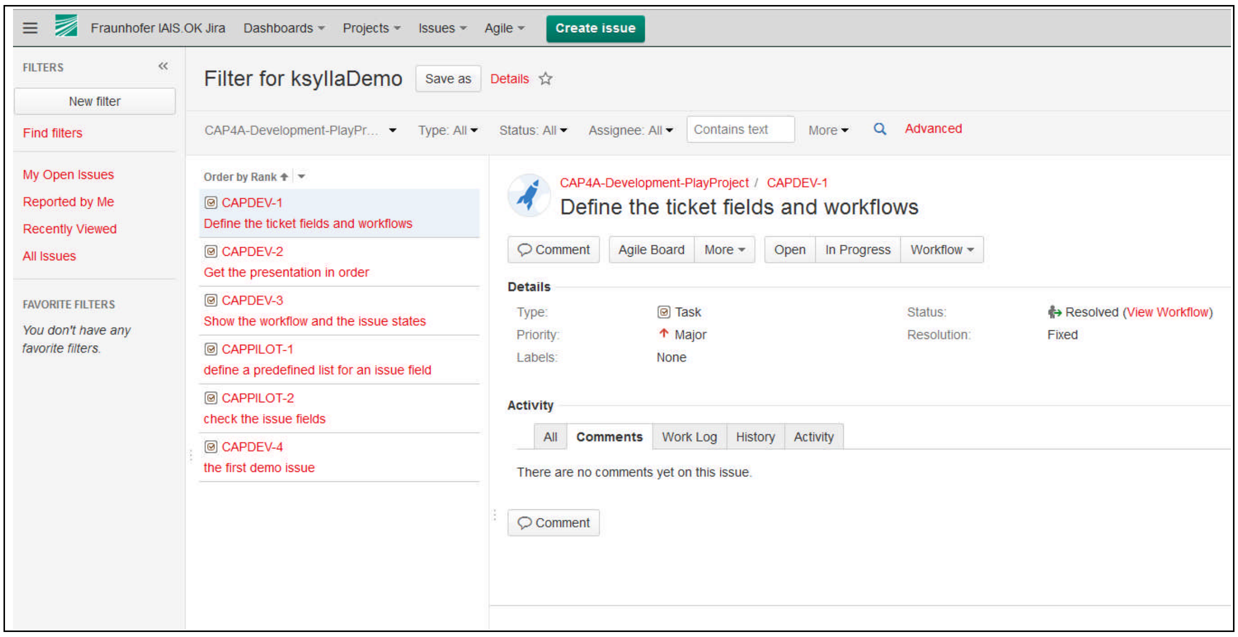
3 Comments

Karsten Gareis
I suggest to call this section "meetings" and have "agendas" and "notes" as subsections
Reply • Delete • 2014-03-14

Karl-Heinz Sylla
I have renamed it. The Name os better than before.

i also created subpages agandas and notes, let's see how to handle it.
I prefer to have an agenda as (sub)chapter in the page of the very note.

Exhibit 2: Atlassian JIRA as used for collaboration in CAP4Access



You don't have any favorite filters.

Filter for ksyllaDemo

CAPDEV-1 Define the ticket fields and workflows
CAPDEV-2 Get the presentation in order
CAPDEV-3 Show the workflow and the issue states
CAPPILOT-1 define a predefined list for an issue field
CAPPILOT-2 check the issue fields
CAPDEV-4 the first demo issue

CAP4A-Development-PlayProject / CAPDEV-1

Define the ticket fields and workflows

Details

Type:	Task	Status:	Resolved (View Workflow)
Priority:	Major	Resolution:	Fixed
Labels:	None		

Activity

There are no comments yet on this issue.

3 Agile Development within CAP4Access

3.1 Activities and Communication

It is widely accepted that the concrete instantiation of agile concepts should reflect the actual shape, background and structure of a project. A European project consortium with partners working at different places and rare face-to-face meetings is challenging because regular (weekly) meetings and frequent communications are important concepts of agile development. A good support by the collaborative software tools is therefore essential for mitigating this drawback.

Close and frequent communication is particularly needed in the beginning of a project in order to form a common understanding of the objectives and tasks. While this is a substantial problem in itself, in CAP4Access we are facing the additional dimension of acquainting several partners with the agile development methodology, which is new to them, and of acquainting all partners with the use of the Atlassian tools.

It would be very unwise to try starting with everything at once. On the one hand, to become familiar with agile development and at the same time with a set of new tools is complex per se. On the other hand, it is one thing to understand the individual pieces of the tools, and another to learn how these pieces are applied and how they work in concert. And even more important is the fact that these tools are not ready-made for any concrete project but they must be configured and adapted to the needs of specific circumstances.

We therefore propose to start small and then gradually improve. Seeing it in another way: The materialization of the agile development processes for CAP4Access itself will be some kind of agile project. We already started with some basic introductions for working with the tools. We will now start with some basic structure of the project's processes and corresponding configurations of the tools.

The basic idea is that we want to foresee separate projects (in terms of the Atlassian vocabulary), which can work rather independently with their own dashboard and cards, and having their own pace (in terms of lengths of sprints). However, there will be connections between the projects so that they are able to communicate with each other.

The individual projects will start with a very basic infrastructure, and they will gradually develop further individual components and structures. Projects may learn from each other and mutually provide assistance to each other, but they are not forced to eventually establish exactly the same structure and components.

We therefore propose to discern the project's major activities into the following categories:

- **Pilot Sites**: User community engagement, use case development, participatory evaluation. These essentially are the WP2 activities.
- **Tools**: Development of CAP4Access tools & applications. These are the WP3 activities.
- **Communication**: exploitation and dissemination. These are the WP5 activities.
- **Coordination**: project management, overall coordination. These are the WP6 activities.

WP4 (Policy Recommendations) will start in M30. We will later see and decide if this will be added to this list or become part of one of the existing projects.

A maybe simplifying but useful view of these activities is: Pilot Sites and Tools have are having their interchanges, but are managed quite independently. Essentially, Pilot Sites will deliver user stories and use cases as “requirements”, Tools will implement these as software or provide required data, and deliveries of tools will be evaluated by Pilot Sites. Communication is a separately managed activity with own plans and deliverables. Its activities will be triggered by results becoming available from the project, but also many factors or events external to the project. Coordination is the activity keeping the others together and streamlining them to meet the targets by enforcing communication and collaboration. In particular, the agile development cycles must somehow be synchronized with the requirements according to the project’s deadlines and milestones. Concretely, a first prototype must become available after six months. Three further prototypes as particularly stable releases must be published and presented at annual reviews and milestones at project months M12, M24, and M30.

We expect that the consortium will continue having a weekly Telco where all partners participate. This will ensure a continuous exchange and notification of important events and progress within the individual “projects”, explicitly signaling the need of communication between the various activities. In addition to the whole consortium’s weekly Telco the individual projects will have their “local” Telcos at their own convenience.

Every partner will have access rights to all Atlassian projects (Pilot Sites, Tools, Communication, and Coordination). At any time, everyone can therefore actively inspect the current state of work within the projects (pull information). For those parts of the work in other projects relevant for one’s own work information can be pushed by subscription.

3.2 Scrum or Kanban?

There exist many concrete conceptualizations and process variants for agile development. When writing the project proposal, we intended to focus on the Scrum methodology because Fraunhofer had gathered positive evidence of its suitability in the BioVeL project. Meanwhile, we think that the Kanban method as described in chapter 3 would be even more suitable. In particular, its higher flexibility in terms of non-fixed sprint lengths, and a more dynamic communication between the various processes seems to be more realistic and appropriate for this particular project. As an advantage, the Atlassian projects that are being established now and further evolved over the next few months will have more freedom to set up their own communication structures and their own pace of sprints.

In general, installing Kanban or Scrum is not an exclusive decision. Certain mixed Kanban and Scrum variants are not unusual.[4] There are a lot of similarities between Scum and Kanban[5]. Both enforce a strict prioritization of tasks so that is always clear what will be done next. The individual teams are expected to establish their own internal organization. Generally, a high level of transparency showing everyone’s work, results, and current and

[4] Kniberg, H. and Skarin, M. (2010). Kanban and Scrum - making the most of both. ISBN:0557138329 9780557138326

[5] Tomas Björkholm (2009) What is Best, Scrum or Kanban? http://www.agileconnection.com/print/article/what-best-Scrum-or-Kanban, *retrieved 2014-02-10*

next tasks, and good communication and collaboration processes and infrastructure are paramount for both.

It is the major difference in the timing of the two variants that lets us favor the Kanban approach as the one to start with. We think that the lack of co-location of workers in a European project will make fixed-length iterations too difficult to manage over a longer period. Our experience in the BioVeL project supports that as there the rule of fixed-length sprints somewhat degraded over time, giving way for some more flexible treatment. In fact the BioVeL project moved the organisation of work from a timeboxed Scrum style to a Kanban style with weekly telcos for reporting the progess and deciding the next activities

This more loose interpretation was not experienced as negative, and the project advanced very successfully. For CAP4Access, we rather think that directly starting with Kanban is the better way. Kanban sees development as a forever ongoing flow of things to do, decisions are taken when they are needed (just-in-time), it has a clear focus on time-to-market, and it tries to minimize lead-time by minimizing the number of parallel tasks in work.

4 Summary

The whole project's planning, coordination, pilot site and software development will be based on an agile approach. Technical support will be based on a communication and documentation infrastructure including GoToMeeting telecommunication and the Atlassian tools for Wikis (Confluence) and issue tracking (JIRA).

25 licenses for Confluence and JIRA have been bought, installed and administered on a server at Fraunhofer IAIS, and users within the project have been registered. We have currently run two GoToMeeting sessions as first introductions of the tool usage.

The agile processes within CAP4Access will be divided into four domains with close correspondence to work packages: Tools (software development, WP3), Pilot Sites (user engagement, WP2), Communication (WP5), and Coordination (WP6). WP4, which starts in project month M30, may be added later. Technically, these four domains will be assigned to individual Atlassian projects.

Communication between the four domains/projects will be facilitated by the weekly consortium's Telco with all partners participating, access rights of all partners to all domains/ projects, by targeted Telcos or physical meetings on demand between all or some domains/projects, and by individual subscriptions to parts of other domains/projects in order to receive alerts about relevant activities.

Regarding the concrete choice of the agile development methodology we have argued to favour the Kanban approach. The main argument is that Kanban leaves more flexibility to adapt to the demands of the individual four projects. In particular, it does not request a fixed length of sprints, and it has a higher focus on user needs and time-to-market.

5 References

The Agile Manifesto. http://www.agilealliance.org/the-alliance/the-agile-manifesto/ [retrieved 2013-01-20].

Björkholm, T. (2009): What is best, Kanban or Scrum? Published online on *AgileConnection*, http://www.agileconnection.com/print/article/what-best-Scrum-or-Kanban [retrieved: 2014-02-10]

Kniberg, H. and Skarin, M. (2010). Kanban and Scrum – making the most of both. C4Media.

http://www.agile-tools.net/

http://sourceforge.net/directory/development/agile/os:windows/freshness:recently-updated/ [retrieved: 2014-02-10].

http://en.wikipedia.org/wiki/Atlassian [retrieved: 2014-02-10]