A Comparative Study of Agile Project Management Software Tools

Marius-Constantin BRAD, Florian-Cristian BIRLOI, Alexandra BRATULESCU, Ioana-Bianca BLAGA
Bucharest University of Economic Studies, Romania
mariusconstantinbrad@yahoo.com, birloiflorian@gmail.com, b_ioana.bianca@yahoo.com

The paper aims to present a comparative analysis between 2 selected agile software tools: Atlassian Jira and Microsoft Team Foundation Server. In the next sections, we will present the methods of our study, expose the analysis of results, make a few discussions on the subject and draw our conclusions. The paper highlights the similarities and differences between these tools with respect to some identified functional requirements. The study will be done in two phases: (1) identification of the key functional requirements for agile management tools, and (2) a comparative analysis of the selected 2 tools. The study has shown that the identified key functional requirements that belong to the groups User stories and epics management, high-level release planning and low-level release planning have been mostly well covered by the examined tools. However, not all the basic functionalities are currently fully covered by some tools. The need for acceptance testing support has been recognized and efforts are being in this direction, although the current state is not satisfactory. User role modeling and personas support has not been covered entirely.

Keywords: *Agile Methods, Agile Software Tools, User Story Management, Software Support, Atlassian Jira, Team Foundation Server, Project Management, Team Build, Process Tracking, Burndown Chart*

1 Introduction and Literature Review

Modern organizations depend on software and software systems in many ways. Business processes are often implemented in a digital flow and without software to support it, even small companies would experience problems. For most companies, the business has changed and is still changing at rapid pace. The development of software has changed as well. Software requirements tend to evolve quickly and become outdated. Traditional project management techniques and especially traditional SDLC (Software Development Life Cycle) methods (i.e. waterfall, "V" or incremental methods) cannot cope with that any more. Nowadays many organizations have large development teams working on software to support the business. Many times these teams are spread globally. This poses many potential problems, such as collaboration issues, source code maintenance, requirements management and so on. Without processes to support modern software development, business will likely suffer. This explains why agile methods have become more and more adopted lately [1].

Agile methods have emerged as a reaction to traditional approaches in software engineering known as documentation-driven and heavyweight software development processes. Although developed separately, different agile methods comprise practices (or techniques) that are based on values and principles defined in the document entitled "Agile Manifesto". They are characterized by iterative and incremental approach to software development and close communication with customers or end users. This family comprises a number of methods. The most widely known are: eXtreme Programming (XP), Scrum, Dynamic Systems Development Method (DSDM), Feature-Driven Development (FDD), Lean Software Development and the Crystal family. DSDM is probably the original agile development method. DSDM was around before the term "agile" was even coined. Scrum concentrates particularly on how to manage tasks within a team-based development environment. Scrum is the most popular and widely adopted agile method, it is relatively simple to implement and addresses many of the management issues that have plagued IT develop-

ment teams for decades. On the other side, Extreme Programming (XP) is a software development methodology, which intends to improve software quality and responsiveness to changing customer requirements in a more radical way than Scrum does.

Every agile method has its own tools. The core of an agile development project are user stories (sometimes called epics) which were initially hand written by end users on index cards and placed on pin boards. Nowadays agile teams are able to use computer-based tools that offer virtualization of index cards, pin boards (now called taskboards) and more other features.

This study aims to present a comparative analysis between 2 selected agile software tools: Atlassian Jira and Microsoft Team Foundation Server. We try to reveal similarities and differences between these tools with respect to some identified functional requirements. We test them on our own project environments and we analyze their features. We also seek for the practitioners' worldwide opinion and personal satisfaction in using them. In particular, the study was done in two phases: (1) identification of the key functional requirements for agile management tools, and (2) a comparative analysis of the selected 2 tools.

The literature review on this topic is somehow underrepresented. Although we have dived in to some well-known academic and scientific sources such as IEEE, ACM, Springer or Google Academics few studies concerning functional qualitative comparisons between agile tools have been conducted so far. As for agile tooling oriented surveys, the sources usually come directly from software vendors [1][2].

In 2006, "Agile Project Management (APM), Tooling Survey Results" focused on collecting statistics on tools used in requirements management, and also there are some statistics on agile method used and reasons for selecting an agile project management tool. In 2009 "Agile tools: the good, the bad, and the ugly" mainly focused on tools used in agile projects. It focused on gathering statistics on company structure and maturity of agile methods using TargetProcess trial versions. Although the paper has published a couple years ago and in recent years, many new tools have captured the market, it is beneficial as a reference to choose most important tools and metrics. In 2013, "8th Annual State of Agile," written by the VersionOne Company includes a normalized and wide distribution of responses of multitude of channels from companies, engineers, scrum masters, product owners and even self-employed engineers. The respondents are from different countries and questions have focused on details such as reasons for adopting Agile, agile techniques used. The main points of the paper are detailed statistics in the agile methods in projects, and the information about adopting agile methods.

The Swedish telecommunications company Ericsson coined one paper that stands out as a general and independent survey focused on the tool usage and needs. Its main purpose was to collect statistics about tools usage and to get a clear idea on what features are most desired by companies. It provides a list of some top-ranked features: (1) reporting features, (2) virtual task board, (3) interface improvements, (4) project status tracking (e.g. burndown charts and charts showing epic and story completion rates). Some other needed features were mentioned, thus in small percentages, such as requirements and user stories prioritization, sprint planning and better handling of tasks, collaboration modules and virtual boards.

Also in 2012, Azizyan described a process to select an agile tool for a specific anonymous company. This paper gives a short description of the company, lists and presents the metrics used for evaluating currently existing tools. What is more important is that this paper introduces a methodology to select the right tool.

In "New Generation Project & Resource Management for Atlassians's Jira by Gerald Aquila, Founder & CEO" the author wants to take a closer look on how the project lifecycle looks like with Onepoint's two-way JIRA integration and how the workload is typically divided between both systems (Onepoint PROJECTS and Atlassinan JIRA). According

to this paper JIRA is one of the most appreciated and widely deployed issue tracking and agile task management solution worldwide. IT departments love it for its ultra-simplistic approach to workflow-style task management. Developers like its tight integration with IDEs and source code management systems such as GIT and SVN. Atlassian also provides great add-ons to JIRA such as Fish Eye, which brings more value to developers and quality engineers.[5]

Another paper "Collaboration Tools and Trends by Clearvisions-cm, 2015" mentioned that one of the bestselling development tools is Atlassian's Jira, ', which is a tracking tool designed for teams planning and building big products. It helps to organize work tasks, assign responsibility, and follow team activity. On Fortune.com, Rich Wong, general partner at Accel Partners and a longstanding Atlassian board member, says workflow platforms such as JIRA Service Desk are redefining business collaboration software.[4][6]

The aforementioned papers and surveys do not offer far-reaching proper study methods nor the possibility to elect the perfect agile software tool due to smallness and immaturity of projects. In addition, none of them offers a standard methodology to choose out of an eclectic range of products. The majority of them are still realized by major agile software vendors. Therefore, the question remains firm: how do we choose the best agile tool for our company's requirements?

In the next sections, we will present the methods of our study, expose the analysis of results, make a few discussions on the subject and draw our conclusions.

2 Methods

In order to present a functional-technical comparative study of agile project management tools we have pursued in identifying some key functional requirements that every agile tool should meet. Then we chose for our tool comparison the standalone Atlassian Jira agile project management software product and the Microsoft Visual Studio Team Foundation Server plugin for Application Lifecycle Management. These ones have been selected according to high rating and recommendations made by community members on the World Wide Web (e.g. userstories.com) and the availability for review.

We have decided to conduct our analysis by answering the following guideline questions:

a) Do the features of the 2 agile software tools cover the key functional requirements as defined in the relevant literature and confirmed by practitioners?
b) Are there any unique features that the tools can offer?
c) What is the general opinion of practitioners about each tool?

At the end of our analysis we provide a full-featured parallel comparison summative table. The individual functional requirements in the list shown below have been elicited based on the importance given to the corresponding agile practices in the primary literature and/or the frequency of taking these requirements into consideration in the literature on agile tool development. The key functional requirements identified are as follows:

1) User role modeling and personas support
2) User stories and epics management (estimation of user stories in different time-work units of measure, decomposition of epics/bigger user stories into the corresponding smaller stories, ways to display epics/user stories)
3) User Acceptance Testing (UAT) support (writing and tracking acceptance tests)
4) High-level release planning (decomposition of a release into iterations/sprints, calculating iteration length, managing a release plan, prioritization of user stories/epics and product backlog and taskboard support)
5) Low-level iteration planning (decomposition of user stories into tasks, assignment of task responsibility and estimation of task work and remaining time)
6) Progress tracking (release burndown charts/bar charts, iteration charts, daily workload graphics and charts, velocity tracking, key performance indicators)

3 Analysis and Results

A) Microsoft Team Foundation Server

Microsoft has been in the business of creating sophisticated software for a long time. Large teams crank out and maintain complex code bases over multiple releases continuously. To be successful at producing software, they had to develop effective approaches for version control, defect and work item tracking and build management. At the same time, they have spent considerable time with customers and industry experts to understand the broad spectrum of project management approaches employed by enterprise customers on a regular basis. With the help of the Microsoft Solutions Framework team, they have distilled the essence of these techniques into a set of flexible project management elements.

After combining the results of their experience and investigation in software creation and methodologies to produce a set of new technologies and techniques that aim optimizing the process of developing software in teams, the result was Microsoft Team Foundation Server (TFS). This tool is a collection of features that are shared by the various members of a project team to enable them to work together more effectively. Team members can share project plans, work products and progress assessments easily and in an effective way.

The main features that are included in Team Foundation Server:

On one hand, TFS was created to provide a hub for all members of the development team to collaborate, representing the team project portal. It also includes project management functions, which allow the shaping of a team project based on a user-specifiable software process and which enable planning and tracking using Microsoft Excel and Microsoft Project.

TFS provides a unified solution for storing source code (along with a history of changes), work item tracking (which can include bugs, requirements and so on) and automated builds. By providing a single solution with all of these capabilities, Microsoft delivered the ability to link all these artifacts for end-to-end traceability, reporting, process enforcement and project management. Visual Studio seamlessly integrated with TFS, but much of this tooling could also be used independently or with third-party source control solutions.[4]

Elements of Team Foundation Server:

1. **Project Management**
2. **Version Control**
3. **Work Item Tracking**
4. **Team Build**
5. **Data Collection and Reporting**
6. **The Project Portal**
7. **Shared Services**

Each tool offered in TFS is highly customizable and automatable. Work item definitions, source control policies, build scripts, process templates, and programmability interfaces all enable customers to tailor their TFS installation to their needs. In addition, at the core of TFS is a set of mechanisms intended to enable outside tools to integrate into the TFS environment as first-class citizens.

Process work item types and workflow using Team Foundation Server:

Teams use the work item types provided with the Agile process template to plan and track progress of software projects. Teams define user stories to manage the backlog of work and then track progress by updating the status of those stories. To gain insight into a portfolio of features, scenarios or user experiences, product owners and program managers can map user stories to features. When teams work in sprints, they define tasks that automatically link to user stories.[7]

Define user stories: User stories define the applications, requirements, and elements that teams need to create. Product owners typically define and stack rank user stories. The team then estimates the effort and work to deliver the highest priority items. Using TFS you can create user stories from the quick add panel on the product backlog page.

Story Points: By defining the Story Points, teams can use the forecast feature and velocity

charts to estimate future sprints or work efforts. By prioritizing the user stories on the backlog page (which is captured in the Stack Rank field), product owners can indicate which items should be given higher priority.

Track progress: Teams can use the Kanban board to track progress of user stories, and the sprint task board to track progress of tasks. Dragging items to a new state column updates the workflow **State** and **Reason** fields. The client can customize the Kanban board to support additional swim lanes or columns.
A typical workflow progression for a user story follows:

- The product owner creates a user story in the **New** state with the default reason, **New user story**.
- The team updates the status to **Active** when they decide to complete the work during the sprint.
- A user story is moved to **Resolved** when the team has completed all its associated tasks and unit tests for the story pass.
- A user story is moved to the **Closed** state when the product owner agrees that the story has been implemented according to the Acceptance Criteria and acceptance tests pass.

By updating the workflow, teams know which items are new, in progress, or completed. Most WITs support transition both forward and backward from each workflow state.

Agile workflow states: These diagrams show the main progression and regression states of the feature, user story, bug, and task work item types.

Map user stories to features: The client can view the scope and progress of work across the product portfolio by defining features and mapping user stories to features. From the Feature backlog page, he can quickly add features, in the same way that added user stories. From the backlog page with **Mapping** turned on, you can drag user stories to the feature that they implement. The links tab captures the links to mapped user stories. From the backlog page with **Mapping** turned on, you can drag user stories to the feature that they implement. The links tab captures the links to mapped user stories.
This mapping creates parent-child links from feature to user stories, which is captured in the links tab. Using portfolio backlogs, the client can drill down from one backlog to another to view the level of detail he wants. Also he can use portfolio backlogs to view a rollup of work in progress across several teams when they setup a hierarchy of teams.[8].

Define tasks: When the team manages their work in sprints, they can use the sprint backlog page to break down the work to be accomplished into distinct tasks. Using agile processes, teams forecast work and define tasks at the start of each sprint, and each team member performs a subset of those tasks. Tasks can include development, testing, and other kinds of work. For example, a developer can define tasks to implement user stories, and a tester can define tasks to write and run test cases.
When teams estimate work using hours or days, they define tasks and the **Remaining Work** and **Activity** (optional) fields.

Test user stories: From the web portal or Test Manager, the client can create test cases that automatically link to a user story or bug. Or he can link a user story to a test case from the links tab. The test case contains a number of fields, many of which are automated and integrated with Test Manager and the build process. For a description of each field, see Build and test integration field reference.
The links tab captures the links to user stories and bugs in a test case. By linking user stories and bugs to test cases, the team can track the progress made in testing each item. By defining these links, you support information that appears in the Stories Overview Report report.

Track code defects: The client can create bugs from the web portal, Visual Studio, or when testing with Test Manager.

Track issues: Issues are used to track events that may block progress or shipping a user

story. Bugs, on the other hand, are used to track code defects. You can add an issue from the New work item widget added to a team dashboard or from the **New** menu on the Queries page. Work items you add from the widget are automatically scoped to the team's area and iteration paths.

Track business value: The Priority field can be used to differentiate the value of various stories. Or to add a custom field to the User Story Work Item Type that tracks the relative value of stories.

Backlog list order: The Stack Rank field is used to track the relative ranking of user stories, however by default it doesn't appear on the work item form. The sequence of items on the backlog page is determined according to where it should be added the items or moved the items on the page. As the items are dragged, a background process updates this field which is assigned to type="Order" in the ProcessConfiguration file.

B) Atlassian Jira

JIRA Software is a proprietary issue-tracking product, developed by Atlassian that unlocks the power of agile by giving your team the tools to easily create & estimate stories, build a sprint backlog, identify team commitments & velocity, visualize team activity, and report on your team's progress. According to Atlassian, over 25.000 customers around the globe use JIRA for issue tracking.

A really nice thing about Jira is the flexibility it provides, becoming very helpful for whom work with multiple types and sizes of projects. You can customize screens, fields, workflows, share configurations between projects, import issues from Github, smart commits boards with multiple projects. You can have a board for each project or a board with multiple projects, making easier to plan the week since you have to switch between different projects daily. Another great feature is its powerful search. You can create very complex queries in a kind of SQL syntax if you want.

User role modeling: With JIRA project roles are a flexible way to associate users and/or groups with particular projects. Project roles can be used in: permission schemes, email notification schemes, issue security levels, comment visibility and workflow conditions.

JIRA has 3 default project roles which are created immediately after installing the product. These roles are: *Administrators, Developers* and *Users*. You can create, edit and delete project roles according to your organization requirements. After a role is created it can be assigned to any user of that particular project including the project administrator.

Deleting a project role will remove any assigned users and groups from that project role, for all projects, this kind of operation must be treated very carefully because if a role is going to be deleted then all the permissions associated with it will be nullified.

User stories and epics management: JIRA has a hierarchy for organizing work: initiatives, epics (that are a single feature or initiative), issues (user stories and tasks) represent the pieces of a feature, and sub-tasks are even smaller chunks of work that comprise the parent story or task.

A *story point* is an estimate of the relative complexity of a story. In JIRA Agile, you can choose to perform estimation for each board based on either Story Points, hours, or any other numeric field of your choice. If an issue's description sounds more like a feature, or the workload is morphing into a larger ambition, the issue should be turned into an epic, then linked with its component user stories and tasks.

In JIRA, there is only really support for two levels of issue hierarchy, task and subtask. An Epic is unique and behaves more like an attribute of another issue. Kind of like how versions are just attributes that help with structuring the work in your project.

In textbook scrum, the only hierarchy is stories and the subtasks that get them done (there is no such thing as a 'Feature', so to speak). When a story is too big to fit into a single sprint, it is split into smaller stories, and the original 'big story' becomes an Epic and is

used like a label to indicate the smaller stories have a common origin.

Acceptance testing support: Acceptance tests allow you to express specific needs for your software product in a way that is testable and measurable. It is also invaluable for breaking down the barriers in software development.

If you need to do UATs, JIRA puts at your disposal: 'Behave'. Behave for JIRA is a tool for agile testing and requirements discovery within JIRA. It allows users to easily add acceptance tests to any issue in your JIRA projects. Acceptance tests are written in a natural language, e.g. English, but in a structured way so that those needs can be matched up to the software that is created to satisfy them. To have a better understanding of this concept you can think of it as "specification by example," an agile testing method where automated acceptance criteria are defined early in the development cycle and used in the development process itself, rather than as validation after testing is completed.

An example acceptance test would be:

Given *a specific situation* - When *something occurs* - Then *you will get a specific outcome.*

Behave for JIRA is for

- **Product owners** to define requirements early in the development cycle and attach them to user stories, which is critical to establishing the real requirements in the product and responding to customer demands.
- **Developers**, who can use Gherkin to define tests and automate them with Cucumber, speeding up the development process by clarifying requirements and ensuring that written code has the functionality customers want.
- **Testers,** who can read tests in natural language and understand the entire context of the code, and can track any broken functionality up to the scenario level.

Release planning: If your team is geographically distributed, planning and communicating not only becomes more of a challenge, but also more critical to the success of your project. That is where a centralized release-planning page comes in.

Every product release requires a lot of hard work and a ton of coordination between individuals and teams. At Atlassian feature releases are planed using a page on their internal wiki (Confluence, which is their collaborative tool) that organizes all the relevant information in a central place that is accessible to the team and anyone else who needs to know what's going on.

Planning and communicating this way solves a slew of problems all in one go. It is essential that anyone who comes to your release-planning page can quickly identify who is involved, and what the goals and expected outcomes are.

You can start by creating a blank page and then add a two-column page layout so you can fit all your key information above the fold. Add a table inside it that displays the high-level details of your release. Should use the profile picture macro, which you can find in the macro browser and then typing "profile picture", to display each person's image. This helps people put a face to a name. Hover over someone's name in an @mention and you'll find a bit more information about that person.

You will also want to capture critical details like the name of the release, expected ship date, and the status of the planning page or even the release itself. Should use the status macro to communicate status of pages or individual line items. You will find it in the "Insert more content" button in the editor toolbar. Change the text and the color to indicate changes in status.

To visualize your plans you should sketch a roadmap to get a rough idea of how various streams of work will fit together and to communicate the timing within the team and to others. You can use the roadmap macro in Confluence to visualize the plans at a high level. Insert the *roadmap macro* just like any other macro. You can then create as detailed or as simple a roadmap as you need by adding additional lanes for work streams, bars for epics, and markers for milestones.

The final piece in your release planning page is all about connecting people to the information they need, thus connecting the dots.
Pages that are typically created for a release include the following:

- Design Hub - for all the relevant UX/UI designs
- Competitor insights - an overview of how other tools tackle this problem
- Analytics - how will we measure usage? What other feature usage might be affected?
- Success criteria - what metrics do we want to hit?
- Workshop notes - takeaways and whiteboards from related spikes and workshops
- User testing - plans and notes around testing the new features with users

Commonly the page is finished off with a release checklist using tasks.

Low-level iteration planning: In JIRA the iteration planning is made through the Tempo Planer. With Tempo Planner, product managers can now view story points on a feature level and thereby track the progress of features for teams that work in relative estimations.
Story points mark the effort (rather than actual hours) that a project requires – they represent an abstract scale for measuring the amount of focus, work, risk and complexity that goes into a story. The majority of users incorporate story points into their planning and Tempo may well add to the range of options for those users in the future. Many teams estimate in days or hours when beginning the agile journey. Breaking user stories down into component tasks that last no more than 8 to 12 hours is a huge step forward in taming uncertainty.
Product managers will be able to view aggregated story points in the Team Backlog and plan out individual sprints based on the amount of effort that they estimate would go into the sprint. Provided that story points have been set for the issue, they will be visible in the header of each issue contained within a selected iteration. A sum of story points can be seen in the metrics bar in the Team Backlog.
Story points can also be viewed in the Program Kanban, where they are shown in the header of each epic, next to the hour estimate. This constitutes a new way to determine the scope of an epic with ease.
The new Iteration Timeline in the Team Backlog opens up a wealth of possibilities to help team leaders create reliable forecasts for their team's workloads.
An iteration section has been added to the team overview page. There, users can see at a glance the status of the iteration a team is currently working on. Users can view the estimated and remaining capacity to see if an iteration is on track and dive right into it if necessary. If the iteration is over 100% capacity then it will show as "over capacity".
You can't improve your processes if you don't measure them. To help teams keep track of time, JIRA supports a feature called time tracking, which allows teams to estimate work and log the amount of time spent on an issue into JIRA. JIRA then aggregates that data in several useful reports. JIRA has a panel in the issue detail view that exposes time information.
The time tracking helps the team understand how work got done, and it gives everyone measurable results from the iteration that can then drive planning for the next iteration.
Users can also view an upcoming iteration and the plan items that compose the iteration by clicking the drop-down arrow. For each plan item you will see the type, key, summary, associated epic, and remaining estimate. Clicking on an iteration will take you to the view for the corresponding iteration in the Team Backlog.
In the Team Backlog, users are greeted by the brand-new iteration timeline in place of the previous drag-and-drop view for member availability within the iteration. However, both views are still there and users can choose which views they would like visible by clicking the corresponding symbols in the metrics bar. With the Iteration Timeline, team members can forecast their iteration planning comfortably against actual visualizations of the plan. When you assign an issue to a team

member, the forecast will show a plan for the team member on the iteration timeline.

Progress tracking: When backlog items are linked with your JIRA application issue, you can track their status and progress directly from your plan. Portfolio for JIRA supports different ways to track progress depending on your team's requirements. In the backlog progress column, progress is displayed for individual story items, and for epics and initiatives as an aggregate of all their sub-stories. For items that are un-estimated, the option exists to show those items in relation to items that are estimated. This allows you to see the percentage of work done on estimated items (which could be 100%), while still seeing that some un-estimated items are still outstanding and require work. The progress and status columns in the backlog allow you to see the progress of your plan items.

- Issue Status - Only exists if the backlog item links to one or multiple JIRA applications issues. It shows the actual workflow status of these issues. In case of multiple issue links, an icon is shown for the status of each linked issue.
- Progress - Sum of work logs on the linked JIRA applications issue(s), as well as their child elements.

Progress bars show the progress of any linked items in the backlog. If your active filters are hiding issues, the progress of those issues will not be represented in the progress bar. To see the detailed progress of any item on the backlog click on the item's progress bar.

1. When an epic or initiative is expanded, the bar shows the total progress of all the child stories. If the 'display un-estimated stories' option is set in progress tracking options, the progress bar is displayed as a grey bar that represents the ratio of un-estimated items to the total number of items.
2. Items with no progress are displayed as a faded grey line.
3. Stories that are in progress show the percentage of progress completed as a green line.
4. Completed issues show a full green line.
5. Un-estimated items show a dark grey line.

Progress tracking types:

- Time based progress tracking

The progress is calculated based on the time spent that is entered into the issue's work log in your JIRA application. If work time is logged for an issue, its progress is calculated as follows:

Progress = Time Spent / (Time Spent + Remaining Estimate)

- Resolved issue count progress tracking

Progress is calculated based on the issue's Resolution field and the progress of sub-issues. If the issue does have sub-issues progress is calculated as follows:

Progress = Number of Resolved Child Elements / Total Number of Child Elements

- Story point progress tracking

Story point progress is calculated from the estimates set in JIRA application issues. The stories progress that do not have sub-tasks will be 0 until the issue is resolved, at which point its progress will be 100% (complete). The progress of an epic is computed as:

Progress = ∑(Estimate (Story 1) x Progress (Story 1) … Estimate (Story n) x Progress (Story n)) / Estimate (Epic).

The ***Burndown Chart*** is another useful tracking tool, which can help you visualize your team's progress, as well as determine whether your team is on target to achieve the sprint goal. The grey line in your Burndown Chart is a guide showing the rate of work required to complete the sprint. The red line, on the other hand, shows the actual work completed by your team. If your Burndown Chart shows the red line above the grey line, your team may not achieve the sprint goal. You may want to consider removing some issues from the sprint.

Below is the resulting summative table after our analysis.

Table 1. Summative considerations

Features	Microsoft Team Foundation Server	Atlassian Jira
User role modeling and personas support		
User role modeling	X	✓
Personas Support	X	X
User stories and epics management		
Estimation of user stories in different time-work units of measure	✓	✓
Decomposition of epics/bigger user stories into the corresponding smaller stories	X	✓
Ways to display epics/user stories	✓	✓
UAT support		
Writing and tracking acceptance tests	X	✓
High-level release planning		
Decomposition of a release into iterations/sprints, calculating iteration length	✓	✓
Managing a release plan	X	✓
Prioritization of user stories/epics	✓	✓
Product backlog and taskboard support	✓	✓
Low-level iteration planning		
Decomposition of user stories into tasks	✓	✓
Assignment of task responsibility and estimation of task work and remaining time	✓	✓
Progress tracking		
Release burndown charts/bar charts	✓	✓
Iteration charts	X	X
Daily workload graphics and charts	X	✓
Velocity tracking	✓	X
Key performance indicators	X	X

4 Discussions

In this section, we discuss the results of our research from the perspective of the three guiding methodology questions:

a) Do the features of the 2 agile software tools cover the key functional requirements as defined in the relevant literature and confirmed by practitioners?
b) Are there any unique features that the tools can offer?
c) What is the general opinion of practitioners about each tool?

Judging by our summative table, the Atlassian Jira software covers more key functional requirements than TFS. It has support for user role modeling and it can decompose big user stories into smaller ones. It can also provide the environment for writing and tracking acceptance tests. There are some drawbacks regarding personas support, iteration charts, velocity tracking or key performance indicators. However, these improvements must not be at the expense of usability as it is found to be of utmost importance for agile project management tools.

On the other hand, TFS has some unique features that make it different from other agile tools. TFS is an application lifecycle management (ALM) solution, while Jira is more like an issue tracker. It has features like source control and automatic builds, check-in, check-out mechanisms and it is integrated with Visual Studio.

Although both tools are commercially licensed, each one has good reviews on many practitioners' web sites. The essence of user satisfaction lies in their needs which, based on the results of the qualitative analysis, may significantly differ from team to team. For instance Atlassian Jira has been preferred in many open source projects such as JBoss and Spring. When it comes to specific agile methods, both tools have been widely used for Scrum projects and Scrum like processes. In fact, when learning the Scrum methodology or any other Agile method, users prefer to experiment with the help of Atlassian Jira.

There is a general need for integrated "software development life cycle" platforms that would combine software development and testing processes with agile project management processes. There is also a need for an integration with collaboration tools, as SDLC becomes more and more distributed with teams spread all over the globe. Microsoft tries to bring all these elements together with its core development IDE Microsoft Visual Studio 2015 (as its latest edition) in perfect harmony with TFS.

5 Conclusions

In the present study, the key functional requirements for our tools have been identified and a comparative analysis has been made between the two of them. The study has tried to show how these tools implement the identified functional requirements and on the potential differences in the corresponding set of features, they offer, especially in terms of support for agile concepts and practices.

The identified key requirements belong to one of the following six groups:

- user role modeling and personas support;
- user stories and epics management;
- acceptance testing support;
- high-level release planning;
- low-level iteration planning;
- process tracking

The following tools were selected for the comparative analysis: Atlassian Jira and Microsoft Team Foundation Server. The selection criteria were: diversity, high rating and availability for review.

The tools have been compared based on the following criteria: coverage of the key functional requirements by the provided set of features, support for basic agile concepts and practices and user satisfaction with the tool.

The study has shown that the identified key functional requirements that belong to the groups User stories and epics management, high-level release planning and low-level release planning have been mostly well covered by the examined tools. In general, there is a noticeable trend to further enrich process tracking. However, not all the basic functionalities are currently fully covered by some tools. The need for acceptance testing support has been recognized and efforts are being in this direction, although the current state is not

satisfactory. User role modeling and personas support has not been covered entirely.
However, the coverage of the corresponding functional requirements itself does not say enough about the quality of support for agile concepts and practices. The study has revealed that there may be significant differences in the way agile concepts and practices have been supported by different tools, if supported at all. This is corroborated by the result of the qualitative analysis of user reviews which showed that agile professionals invest considerable time and efforts to find a tool, and if is really necessary, a tool that is easy to customize.

Acknowledgement
This research paper is made possible through the help and support from everyone, including: parents, teachers, family, friends, and in essence, all sentient beings. Especially, please allow me to dedicate our acknowledgment of gratitude toward the following significant advisors and contributors:
First and foremost, we would like to thank Professor Constanta BODEA for her most support and encouragement. She kindly read our paper and offered invaluable detailed advices on grammar, organization, and the theme of the paper.
Secondly, we would like to thank our colleagues who work in the same field for reading our research paper and providing valuable advices, as well as all the other professors who have taught us over the past two years on our pursuit of the master degree.
Finally, we sincerely thank to our parents, family, and friends, who provide the advice and financial support. The product of this research paper would not be possible without all of them.

References

[1] James Shore, ”The Art of Agile Development”, O’Reilly Media, 2007
[2] Stephen Haunts, ”Agile Software Development Succintly”, Syncfusion Inc., 2015
[3] L. Constantine, ”Users, Roles and Personas”, http://www.foruse.com/articles/rolespersonas.pdf
[4] Mickey Gousset, Ed Blankenship, Martin Woodward, Grant Holliday,”Proffesional Application Lifecycle Management with Visual Studio 2012”, Apress, 2012
[5] Brian Blackman, Gordon Beeming, Michael Fourie, Willy-Peter Schaub, ”Managing Agile Open-Sourse Software Projects with Microsoft Visual Studio Online”, Microsoft Press,1st editon,2015
[6] Jennifer Greene, Andrew Stellman,”Learning Agile”, O’Reillu Media,2014
[7] “Buyer’s Guide Agile Project Management Software, http://technologyadvice.com/smart-advisor/downloads/technologyadvice-agile-pm-buyers-guide-43d.pdf
[8] “About agile work innovation”, http://planbox.com/about-agile-work-innovation/