|  |
| --- |
| Hansoft AB |
| Hansoft Scaled Agile Support Kit |
| An add-on to Hansoft for working with the Scaled Agile Framework  2013-09-25, Version 0.93 (Beta) |

|  |
| --- |
|  |

Contents

[Introduction 4](#_Toc368489626)

[Terms and conditions 4](#_Toc368489627)

[What is in the template database? 4](#_Toc368489628)

[Overview 4](#_Toc368489629)

[Portfolio project 5](#_Toc368489630)

[Program projects 5](#_Toc368489631)

[Schedule 5](#_Toc368489632)

[Program backlog 6](#_Toc368489633)

[Working on the portfolio level 7](#_Toc368489634)

[The feature summary field 8](#_Toc368489635)

[Reporting in the portfolio backlog 9](#_Toc368489636)

[Adding features to Epics 11](#_Toc368489637)

[Working on the program level 12](#_Toc368489638)

[Adding features to the program backlog 12](#_Toc368489639)

[Release Planning 12](#_Toc368489640)

[Preparations 13](#_Toc368489641)

[Teams flesh out stories and making preliminary sprint plans 13](#_Toc368489642)

[Checking and adjusting the overall plans 13](#_Toc368489643)

[Finishing up the release planning 14](#_Toc368489644)

[Reports supporting release planning 14](#_Toc368489645)

[Working in a team 16](#_Toc368489646)

[Working with the team backlog 16](#_Toc368489647)

[Working in a sprint 17](#_Toc368489648)

[Understanding the big picture 18](#_Toc368489649)

[Understanding what other teams are doing 18](#_Toc368489650)

[Understanding and managing dependencies 20](#_Toc368489651)

[Facilitating team communication and learning 21](#_Toc368489652)

[Program governance 21](#_Toc368489653)

[Release status 22](#_Toc368489654)

[Velocity by Sprint 22](#_Toc368489655)

[Velocity by Team 23](#_Toc368489656)

[Blocked/Started/Completed 24h 23](#_Toc368489657)

[Blocked/Started/Completed 1w 24](#_Toc368489658)

[Features by Epic 24](#_Toc368489659)

[Customizations in the Program 24](#_Toc368489660)

[Handling non-functional requirements 25](#_Toc368489661)

[Non-functional requirements applicable for a specific functionality 25](#_Toc368489662)

[Non-functional requirements for the system as a whole 26](#_Toc368489663)

[Installing the database and the customizations 26](#_Toc368489664)

[Prerequisites 26](#_Toc368489665)

[Importing the database 27](#_Toc368489666)

[Preparing the database 27](#_Toc368489667)

[Adapting users and groups 27](#_Toc368489668)

[View presets 28](#_Toc368489669)

[Adapting custom columns 29](#_Toc368489670)

[Adapting the structure of schedules and backlogs 29](#_Toc368489671)

[Installing the SAFe Hansoft client plugin 29](#_Toc368489672)

[Installing and configuring Jean for Hansoft 30](#_Toc368489673)

# Introduction

This document describes a setup of Hansoft for working with the Scaled Agile Framework[[1]](#footnote-1) (SAFe). The description is based on a “template” Hansoft database which defines a set of custom columns, user groups, reports, view presets and suggested structures for the product backlog and the project schedule in Hansoft. To complement the template database there are additional customizations to automatically calculate/update summary fields with Jean for Hansoft[[2]](#footnote-2) and also a Hansoft client plugin to add custom menu options to the Hansoft client. It is assumed that the reader of this document is familiar with the SAFe and Hansoft in general as the document only discusses the implementation aspects of supporting SAFe in Hansoft.

The approach described in this document should, much as SAFe itself, be seen as a framework and a useful starting point for implementing SAFe and Hansoft in an organization. The approach should be adapted to the organizations’ specific needs and working procedures, the information needed to support different stakeholders, meetings and so on. Such information can either be captured as additional custom fields in Hansoft or in documents that are attached to the different items in Hansoft (e.g. epics and features). This should not be seen as a limitation of the approach as a complete boiler-plate solution would be an impediment for the experimentation, continuous learning and improvement that are at the very heart of lean and agile.

# Terms and conditions

Hansoft Scaled Agile Support Kit for Hansoft by Svante Lidman (Hansoft AB) is licensed under what is known as an MIT License[[3]](#footnote-3).

This program is not part of the official Hansoft product or subject to its license agreement. The program is provided as is and there is no obligation on Hansoft AB to provide support, update or enhance this program. Questions can be sent to svante.lidman@hansoft.com and will be answered when other obligations so permit.

Scaled Agile Framework and SAFe are trademarks of Leffingwell, LLC. These trademarks are only used for reference purpose and their use does imply any kind of endorsement of Hansoft Scaled Agile Support Kit by Leffingwell, LLC.

# What is in the template database?

## Overview

In the template database you will find three different projects. The project called Portfolio is what represents the portfolio level in SAFe as described in the following sections. The projects called Program 1 and Program 2 are example development programs or agile release trains and they will contain all the information related to the development of features and components in each respective program. This includes the breakdown of features into stories and how they are distributed across teams, PSIs, and sprints and so on.

## Portfolio project

In the Portfolio project only the Product backlog functionality in Hansoft is used (called Portfolio backlog in this project). Hence when you first open this project there will be an empty schedule view and you will need to click Portfolio Backlog to see the Investment themes and the Epics of the portfolio as described below. The Portfolio backlog contains example investment themes and epics called Theme 1, Theme 2, Business Epic 1, Architectural Epic 1 and so on. See also Figure 3 - The hierarchy view of the portfolio backlog.

## Program projects

The Program 1 project is populated with example data in the schedule and the product backlog sections of Hansoft (called Program backlog in the example database).

### Schedule

In the schedule section you will find an example structure as shown in Figure 1.



Figure 1 - The schedule section in Program 1

The schedule is organized so that all Milestone items for releases of PSIs and features are placed in the Roadmap part of the schedule. First you have PSI 1 which is the release date for the PSI that is currently worked on in the example database. Directly after the PSI 1 milestone item there is a group called Features that contain the feature milestones for all the features that are targeted for PSI 1. Then follows PSI 2 and PSI 3 which are the release milestones for the two following PSIs. No feature milestones have been created for these PSIs yet.

In the Features section of the schedule there will be one child item for each feature that is worked on currently, F1, F4, F5, and F6 in this case. Below each feature there will be one child item for each team that works on the feature. As you can see Feature Team A is working on Feature F1 and three sprints have been created in the schedule for the purpose of this work. For Feature F4 it is indicated that Feature Team B and Feature Team C will work on this feature together but no sprints have yet been created in the schedule. No team container items have yet been created below features F5 and F6.

In the Components section of the schedule there is a similar structure as for features but for the work that is done by Component teams that have responsibility for maintaining and enhancing specific components as opposed to working on cross-component features.

### Program backlog

In the Program backlog section of the Program 1 project you will find an example structure as shown in Figure 2.



Figure 2 - The Program backlog section of Program 1

On the top level the program backlog is structured into three parts or “buckets”:

* Development – This part contains all features, stories and any other program backlog items that are related to the PSI that the program is currently working on.
* Release planning – This part contains features that may become part of the PSI after the current one. This area also acts as a holding ground during the Release planning event in SAFe.
* Feature backlog – This part contains features that have been identified and that may become part of a future PSI. Typically these features are not yet broken down into stories.

In the example database features are called F1, F2, F3 and so on and are broken down into stories as indicated by Story 1, Story 2, Story 3 and so on.

# Working on the portfolio level

In SAFe it is defined that you on the portfolio level have business epics and architectural epics that follow a Kanban flow with the steps/phases:

* Funnel
* Backlog
* Analysis
* Implementation

The different epics are associated with the investment themes that are defined by the business. Once an epic reaches the Implementation step/phase it is broken down into features that are assigned to the feature backlogs of the different programs/release trains.

The Portfolio project in Hansoft supports this by having epics as backlog items in the product/portfolio backlog. The epics are grouped by their investment themes as show in Figure 3. Links are then created in Hansoft between each Epic and its associated features in the different programs to provide navigability and traceability between the portfolio and the development programs.

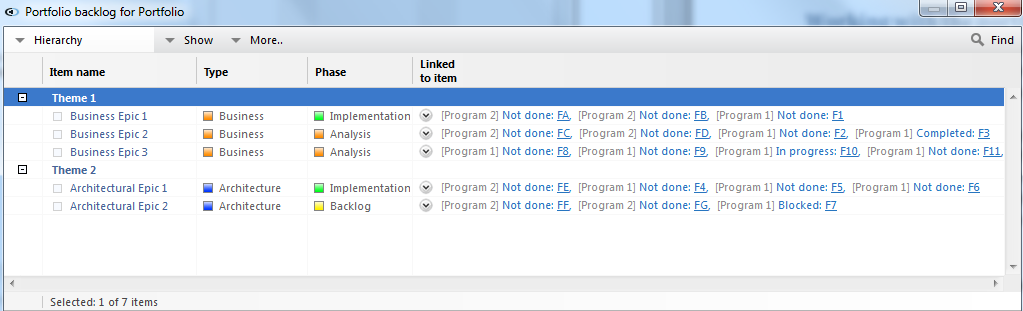


Figure 3 - The hierarchy view of the portfolio backlog

A few custom fields/columns are added in the product backlog to capture SAFe specific information as follows:

* Type – Indicates whether an epic is a business epic or an architectural epic.
* Phase – The step/phase that an epic currently is in, i.e., Funnel, Backlog, Analysis, or Implementation. You can set this value either by changing the property itself or dragging a card representing an epic to a new column in the board view (provided that Phase is shown as columns).
* Theme – In addition to the theme being reflected by the grouping in the product backlog an explicit Theme label is automatically attached to epics.
* Feature summary – This field provides a summary of the status of the different features that are associated with an epic. This is field is automatically updated in real time as the status of the features changes in their respective programs. Features and programs are described in more detail further down in this document.

The flow across the portfolio steps are visualized in the board view of the product backlog as show in Figure 4.

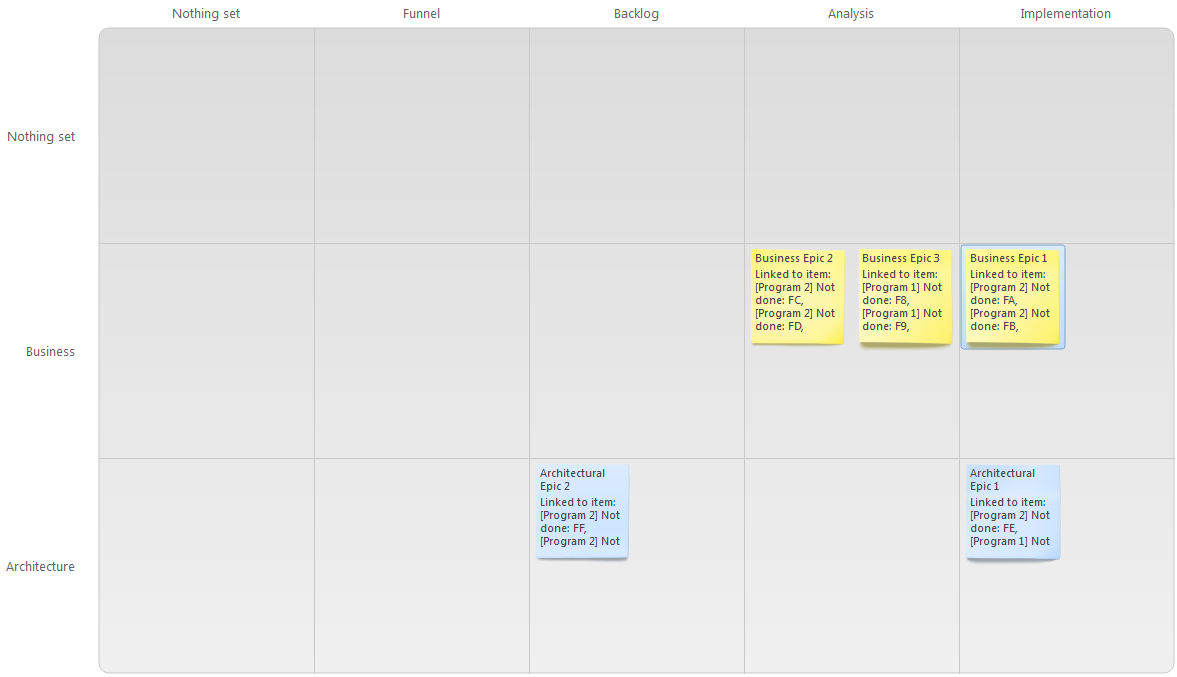


Figure 4 - The board view of the portfolio backlog

## The feature summary field

A custom column/field called feature summary has been added in the backlog of the portfolio project that for each epic will display an always up-to-date summary of all the features that are attached to an epic as show in Figure 5. The following information is displayed for each feature:

* Features in development, i.e., features that are part of the ongoing PSI
  + Name – The name of the feature
  + Status – The status of the feature
  + Done – The number of completed points/ideal days vs. the total number of points/ideal days for the feature.
  + ↓14 – The number of points/ideal days completed in the last 14 days.
  + Est. Done. – The estimated completion date for the feature as determined by the product owner responsible for the feature in the program where it is worked on.
  + Product Owner – The product owner that is responsible for the feature in the development program.
* Features in release planning, i.e., features that are planned to be part of the next PSI
  + Name – The name of the feature
  + Est. – The estimated size of the feature in points/ideal day.
  + Team – The team that is assigned to the feature in release planning.
  + Product Owner – The product owner that is responsible for the feature in release planning.
* Features in the feature backlog, i.e., features that possibly will be included in a future PSI
  + Name – The name of the feature
  + Est. – The estimated size of the feature in points/ideal day.
  + PSI – The tentatively planned PSI for this feature.

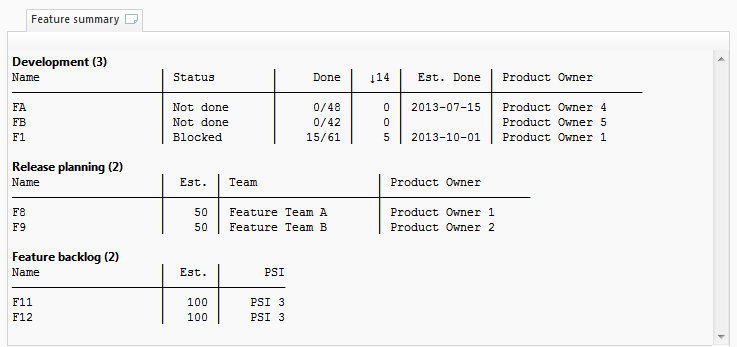


Figure 5 - The Feature summary for an epic

## Reporting in the portfolio backlog

In the portfolio backlog view there are a few example reports:

*Epics by phase* – This report provides a focused view of all the epics in the portfolio grouped by phase to provide a high level view of the portfolio as a whole.

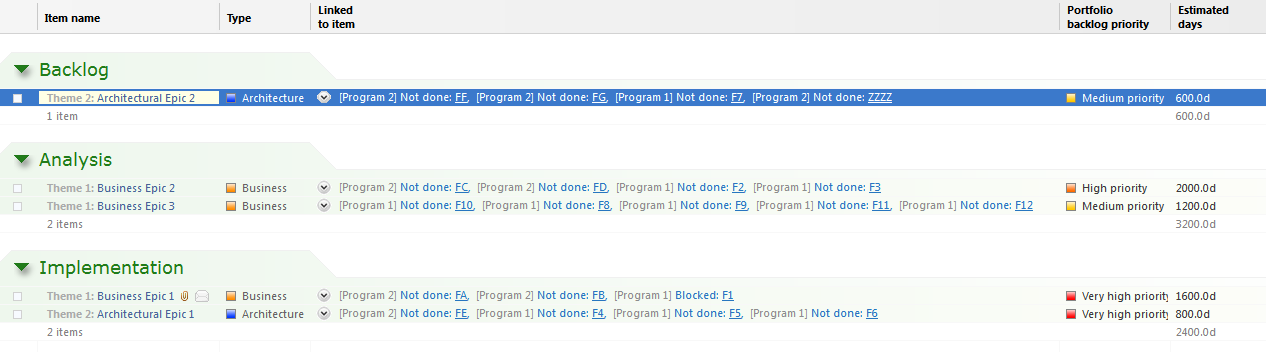


Figure 6 - The Epics by phase report

In addition to reporting in the portfolio project in Hansoft it is useful to consider reporting across the different development programs/agile release trains. This can be done in the Portfolio reporting tab in Hansoft, see also Figure 7.

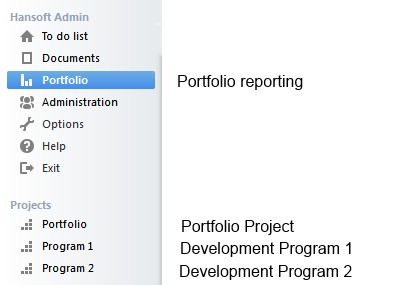


Figure 7 - The portfolio and program tabs

An example of such a report called Feature Overview is included in the example database. This report will show all features across different programs grouped according to Epic and feature and their status.

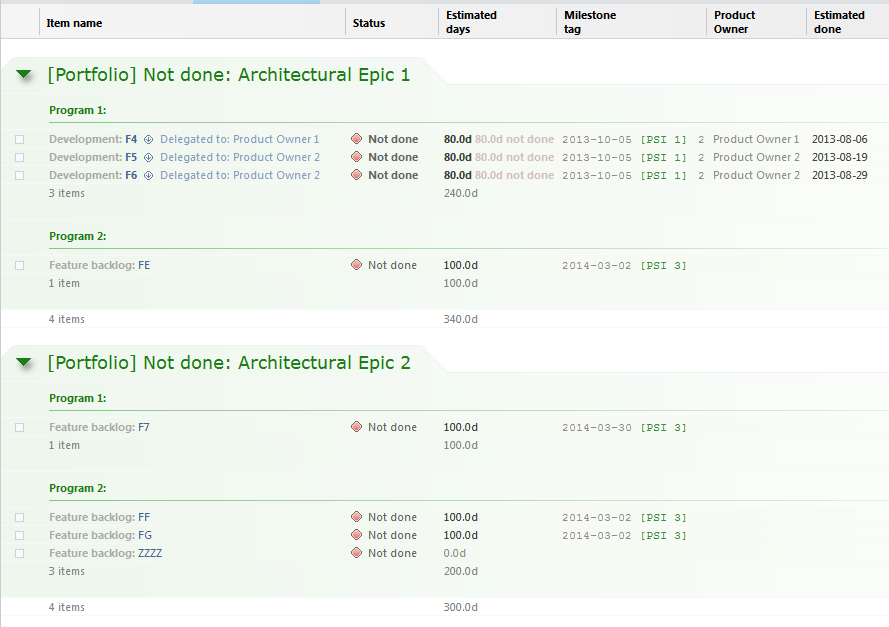


Figure 8 - The Feature Overview report

## Adding features to Epics

A new feature is typically created in the *Feature backlog* section of the product backlog of the program where the feature belongs. To manifest that the feature belongs to a particular Epic a Hansoft link should be created between the epic and the feature. To help with this a menu item has been added to the Hansoft right-click menu to help creating the feature in the right section of the right program backlog and also creating the link. To add a new feature, right click on the epic that it should belong to and select SAFe/Add feature… from the popup menu. Select a program and give the feature a name in the displayed dialogue. See also Figure 9, Figure 10, and Figure 11.

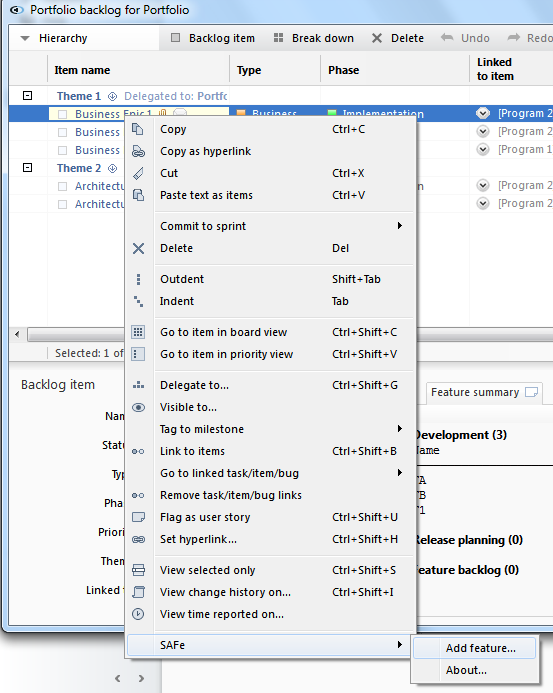


Figure 9 - Adding a new feature from the right-click menu

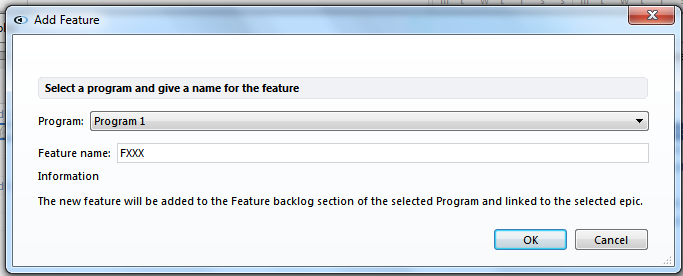


Figure 10 - Giving the feature a name and assigning it to a program

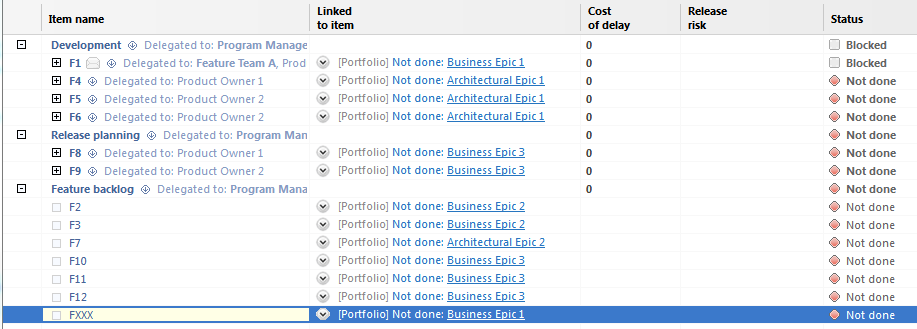


Figure 11 - The created feature in the Program

# Working on the program level

Much of the work on the program level, and more specifically the team level, in SAFe is akin towards Scrum and generally accepted strategies for scaling Scrum to multiple teams. Therefore we do not describe in general how you work in a multi team environment in Hansoft but only things that are specific to SAFe.

A key idea in SAFe is that one program is working towards a series of potentially shippable increments (PSI) of whatever software they are developing. At the beginning of every PSI there is a big meeting called Release Planning where it is decided what features should be developed in the PSI. The team backlogs for the coming PSI are also established during release planning by assigning features to teams that then break them down into stories and make a preliminary assignment of the stories to sprints. To support this flow the product backlog on the program level is structured into three different sections as follow:

* *Feature backlog* – In this section you find features that have been identified but they have not yet been through release planning or committed for development into a PSI.
* *Release planning* – This section is a temporary holding ground for features that will be considered in the next release planning session and also for the planning work that is done during the Release Planning session.
* *Development* – This area of the backlog holds the features that are currently in development and to be released (potentially) as part of the next PSI. This area will hold the team backlogs for the ongoing work during the development of the PSI.

In the following sections it is briefly described how the planning, execution and governance of the development of a PSI is done on the program level in SAFe.

## Adding features to the program backlog

Even though features can be added directly in the program backlog and then linked to their epic in the portfolio it is best to create new features from the Portfolio as described earlier in the section *Adding features to Epics*.

## Release Planning

Note: This is not a full description of the release planning event as such but is focused on how Hansoft is used to facilitate it.

### Preparations

When you are approaching the Release Planning meeting Product Management will, considering portfolio priorities, input from business owners and so on, populate the Release Planning section of the backlog with a set of candidate features for the coming PSI[[4]](#footnote-4). For each candidate feature you also create a release/milestone in the schedule to represent the planned feature completion date. You also create a release/milestone item in Hansoft for the completion date of the PSI itself if it has not already been done. These candidate features are then assigned and delegated in Hansoft to the different Product Owners and/or Teams. After presentation of vision, business strategies and the like the preliminary assignment of features to teams is presented, discussed and adjusted if needed.

### Teams flesh out stories and making preliminary sprint plans

If this has already not been done each team creates the series of sprints that they will work through in the PSI. Allocation during the period (i.e., known forthcoming changes to the team) is updated and out-of-office tasks are created to account for vacations, leaves and other known out-of-office events. Out of office tasks are typically and best kept in a separate out-of-office project in Hansoft[[5]](#footnote-5).

The team will then, together with the product owner for each of their assigned features identify a set of user stories and enter them into Hansoft and make sure that they are estimated, prioritized and of the right granularity. All identified stories should be assigned to the Team itself in the *Team* column in Hansoft.

When stories have been identified, estimated, and prioritized the team makes a preliminary plan of how the stories should be realized across the sprints in the PSI. For this purpose a custom column called *Planned sprint* is available in Hansoft. This preliminary sprint planning is typically best done from the Priority View in Hansoft. Assuming that we know that the team has a typical velocity of X Ideal days/Points per sprint (see Velocity by Team), you can then multi-sort the backlog in Hansoft in the order you want to develop the stories (by risk, priority and size as an example). From the top of the list you then select the amount of stories that correspond to the teams’ velocity and assign them to the first sprint (Planned sprint column), then select the next set corresponding to the team velocity and assign them to the second sprint and so on until you are done with all stories or sprints.

The teams can then review and adjust the sprint plan as required using the *Sprint plans by Team (Release Planning)* report.

As a final measure the team will tag the created stories and sprints to the relevant feature milestones and PSI milestone in the schedule.

### Checking and adjusting the overall plans

In parallel with and after teams make their plans, the planning for the program as a whole can be monitored in the following ways:

* Simply by looking in the hierarchy view and seeing how the different features in the Release planning section of the backlog are broken down.
* By looking at the following reports: *Release planning by feature*, *Release planning by team*, and *Sprint plans by Team (Release Planning).*

### Finishing up the release planning

When there is an agreed set of features and stories for the upcoming PSI you close up the event from a Hansoft point of view as follows:

* Any excess features that didn’t fit in the PSI are cleaned from milestone tags for the upcoming PSI and moved back to the Feature backlog section of the program backlog.
* The features that have been agreed for the upcoming PSI are moved to the Development section of the program backlog.

Note: In Hansoft 7.1 you need to tag sprints to releases to get release burn downs. This requirement will be relaxed in a forthcoming release for Hansoft (fall 2013). In addition to reducing the number of steps required in release planning this will also make it easier for component teams or teams working on multiple features at the same time to get fully accurate burn downs.

### Reports supporting release planning

In the template database a number of reports are available that are useful during Release Planning.

#### Release planning by feature

This report shows all stories that so far have been identified in the Release planning section of the backlog grouped by feature. Essentially this is a cut out of the hierarchy view of the product backlog providing a focused view of the release planning progress.

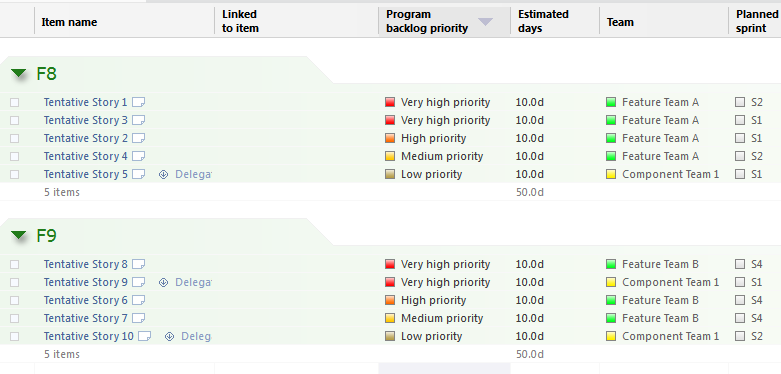


Figure 12 – Report: Release planning by feature

#### Release planning by team

This report shows all stories that so far have been identified in the Release planning section of the backlog grouped by team. This is useful for:

* Monitoring progress in the different teams during the team breakout planning sessions
* Assessing load across teams after team breakout planning sessions

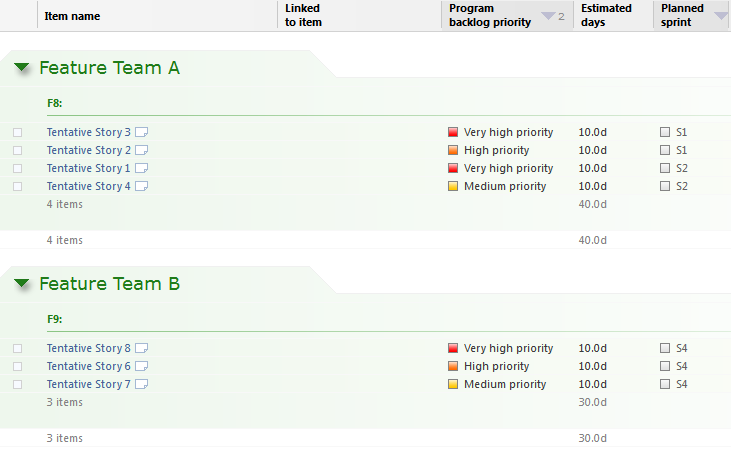


Figure 13 - Report: Release planning by team

#### Sprint plans by Team (Release Planning)

This report presents the sprint plans, i.e., how stories are assigned to sprints for each team. The report helps you to make sure that a reasonable level of planning has been made across all teams.

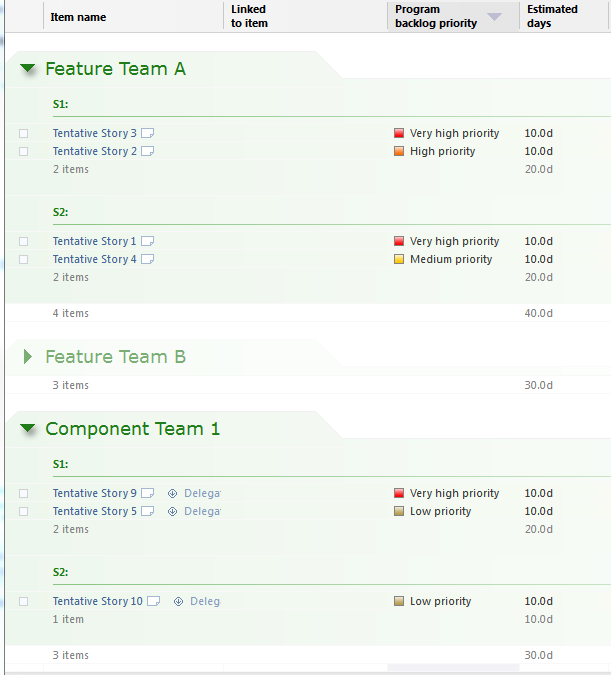


Figure 14 - Report: Sprint plans by Team (Release Planning)

## Working in a team

On the individual team level the Scaled Agile Framework is not much different from what can be considered as mainstream agile practice (Scrum with some XP practices). Therefore this is not described further in this document but backlog grooming, sprint planning etc. is done according to standard practices. A few things are still worth highlighting though as they become more important when working in an environment with many teams:

### Working with the team backlog

The main product backlog is structured by features meaning that all the stories attached to a feature will be grouped under the feature item in the hierarchy. This provides a view of the requirements on the product to make it easy to understand how different requirements relate to one another and if there are dependencies between them and so on.

Ideally a single team will work on one feature at a time and if delegation is used in Hansoft to give teams edit rights to their assigned features (which probably is a good idea), teams can then simply use the *Show editable by me only* option under the View menu in Hansoft (selected by default) to view their team backlog.

However sometimes things are quite not that simple. For very large features you may have multiple teams working on a single feature. Another common situation is that you have a situation where a specialized component team needs to be involved to implement some stories. In these cases you will need to use the Find function in Hansoft to view the backlog for a team. In the figure below you can see how this is done to access the backlog for *Component Team 1* for *PSI 1*.

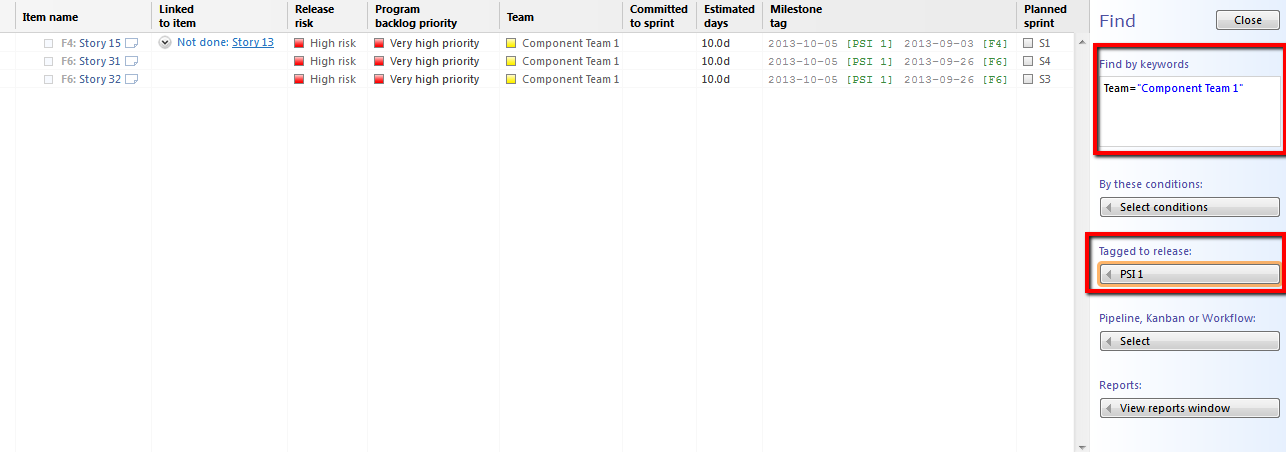


Figure 15 - The team backlog for a component team using the Find function

### Working in a sprint

Most of the work in relation to sprint planning, execution and so on is not any different when you work according to SAFe compared to mainstream practices such as Scrum. See hansoft.com for tutorials[[6]](#footnote-6) on this. In the following section you find additional tips and tricks for working in a sprint in a scaled up environment.

#### Capturing impediments

Impediments are circumstances that are blocking progress on the work in the project and they are by definition related to some existing item in the Hansoft backlog or schedule such as a task or a user story. Therefore, instead of keeping a separate list of impediments they are captured in line with the item as such. This is done as follows:

* The impeded item is set to the status Blocked
* A comment is written in the comment stream of the item describing what the impediment is and how it possibly can be resolved.
* If the impediment is related to some other item that another team is responsible for/working on it is recommended that you create a link to the item you are depending on and also write a comment on the item that you are depending on to describe to the other team how you are depending on the item and why it is blocking your progress. Such a link indicates a need to communicate to resolve the dependency/impediment on a team-to-team level.
* Optionally the impeded item is assigned to whoever is responsible for resolving the impediment, e.g. the teams Scrum Master.

#### Monitoring impediments

Impediments can be monitored on different levels such as Team, Program and even Portfolio by creating reports pulling out items that are currently blocked, that have become blocked during the last day, week, or whatever timespan is desirable. On the Program and Portfolio level it is particularly useful to have reports that shows items that have been impeded/blocked for a long time, for example one whole sprint, which would indicate that the team probably need support in resolving the impediment. In the template database the following reports are defined on the program level to help monitoring impediments.

* *Blocked in sprints* – This report shows impediments in all non-archived sprints on the program level.
* *Blocked stories* – This report shows blocked items in the program backlog.
* *Blocked last week* – This report shows product backlog items that have been blocked during the last week.
* *Blocked for 3 weeks* – This report shows product backlog that have been in the blocked status for 3 weeks or more.

There are also a few reports for looking at the Portfolio level. These reports are found in Portfolio find under the Portfolio tab (not the project) in Hansoft.

* *Blocked last week* – This report shows all items in all programs that have been blocked during the last week.
* *Blocked for 3 weeks* – This report shows all items in all programs that have been in the blocked status for 3 weeks or more.

### Understanding the big picture

Self-organizing teams are one of the panaceas of agile to solve motivational issues, uninformed and slow decision making, to name a few things. However, to successfully self-organize towards the goals of the enterprise as a whole each team needs to have a good understanding of the big picture of the product and how their work is related to the work of other teams.

### Understanding what other teams are doing

As a member in a Hansoft project you by default have read access to all items in the project meaning that you can at any time look at other team’s backlogs, sprint plans and so on. This means that you can look at the full backlog and plans for other teams to understand what they are working on in case there are dependencies. Additionally the board view in Hansoft provides a visual overview of what all teams are working on as shown in the figure below.

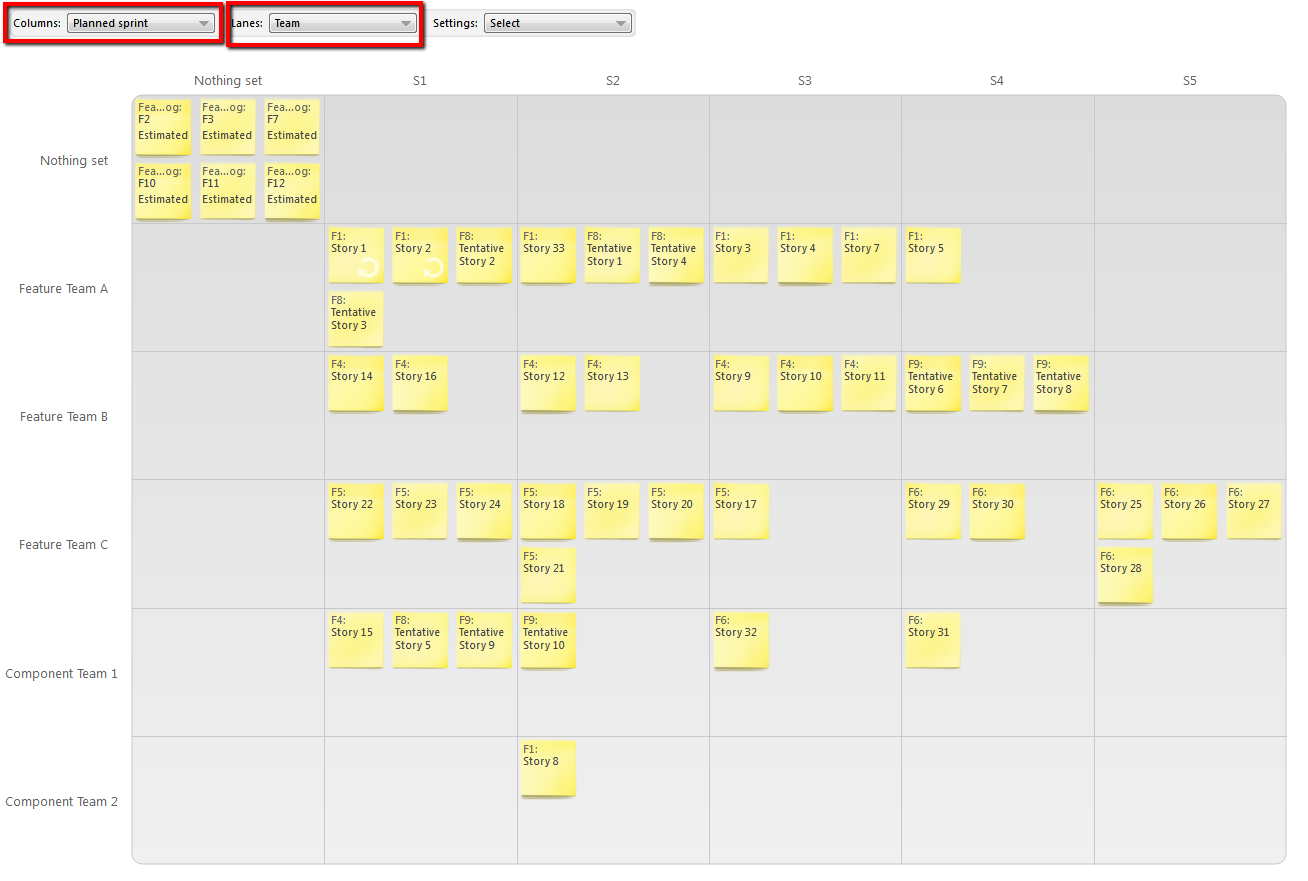


Figure 16 - Overview of stories across teams and sprints

You can then use the Find function to look at just a single feature to understand how the feature is distributed across teams to manage any dependencies proactively.

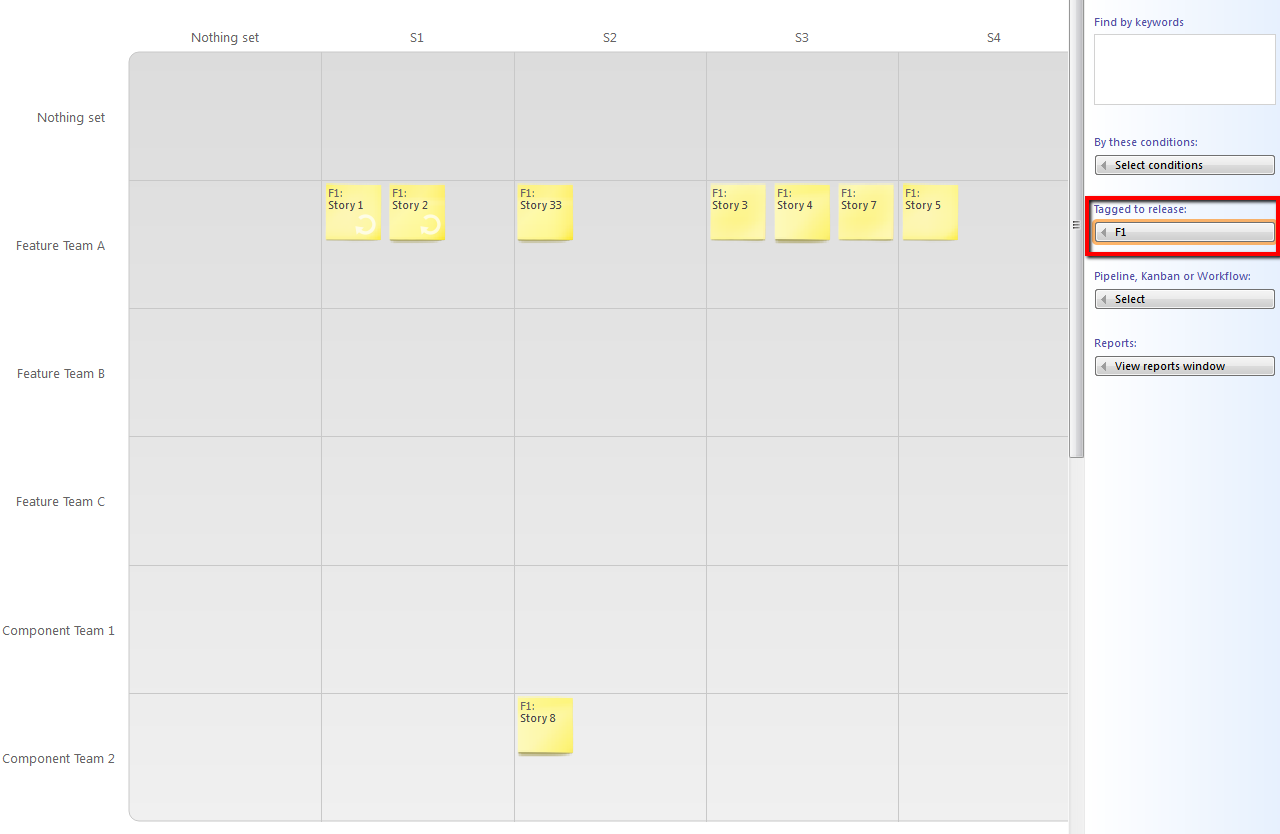


Figure 17 - Overview of Feature F1 across teams and sprints

### Understanding and managing dependencies

The following discussion is not directly tied to the scaled agile framework but is a general discussion of how you can handle dependencies in a large program in general.

Apart from general strategies for softening dependencies such as stubbing/mocking interfaces the most important thing is that the parties with the dependency at all times have an understanding of the dependency and that the communication between dependent teams is open and frequent. Linking, comments, and the chat are some of the features in Hansoft that supports this.

Dependencies between backlog items should generally speaking be avoided. However this is not always possible. Below is a discussion of a few common cases.

#### Multiple teams working on the same feature

To be able to complete a large feature within a certain market window it can be required to have multiple teams working on the same feature. Some recommendations in this case:

Do not try to model the dependencies between all the stories of the feature. For a large feature containing potentially hundreds of stories this will become unmanageable and will not provide the wanted overview of the dependencies. Instead it is recommended to create a high level overview of the dependencies between higher level functional capabilities of the feature/product. Such an overview can be created in any available drawing tool and attached to the feature item itself in the product backlog. The nodes in such an overview diagram may map to *groups of user stories* in the product backlog rather that individual user stories.  
  
This overall view of the dependencies will typically be fairly stable and provide a more usable and easier to maintain overview of how different parts of the feature are dependent on each other compared to creating a detailed dependency mesh between the lowest levels of product backlog items.

#### Part of a feature/story needs to be completed by a component team

It is not uncommon that some part of a complex software product can only be worked on by a team with a special skillset and experience. Typically this means that a small subset of the stories is done by a component team.

Generally speaking it is recommended to not fragmentize the overall backlog by, e.g., by moving the component-related story to a section for the backlog dedicated for generic component work. To keep things in context and understandable it is better to keep the item where it functionally belongs, i.e., as a story of the feature and then use the Team column as an indicator to which team is working on the feature. This way you can keep the whole feature together in a cohesive way while still each team has access to their team backlog as described above. The exception is of course when the component team is realizing a functionality that is impacting multiple features. In this case it is better to move the component story to a component area of the backlog and to create a link from any dependent stories to the component story.

#### A feature is fully dependent on the availability of a new component/architectural feature

This should of course be handled in such a way that the architectural work is completed in a previous PSI according to the idea of the architectural runway in the scaled agile framework.

### Facilitating team communication and learning

#### Standup

There are a number of tools in Hansoft that are useful for facilitating the standup, in particular if you have distributed teams. The standup works best if all team members have updated their tasks in Hansoft before the standup meeting. Things you typically want to look at quickly during the standup include:

* The work allocation bars in the sprint
* Reports:
  + *Feature Team A blocked in sprints* – This report shows impediments in the non-archived sprints that are tagged to Feature Team A. This report is available for everyone in Feature Team A.
  + *Standup Team A* – This report shows all items in Feature Team A’s part of the schedule that have changed during the last 24 hours and that are either Blocked, In progress, or Completed. This report is available for everyone in Feature Team A.
* The board view for the sprint.

#### Retrospectives

Once again the same tools that are useful at the standup are also useful during the retrospectives. In preparation for the retrospective it is suggested that reports, screenshots and the like are prepared and printed on paper before going into the retrospective. The reason is to avoid distractions and keeping the retrospective focused and on track.

Things that are useful during the retrospective include:

* The sprint burn down chart, possibly with overlays such as added work.
* Inspect and discuss items completed/not completed during the sprint.
* Particularly noteworthy task histories, comment streams etc.

Some possible ways to capture and act upon the results of the retrospective are:

* Write the notes as a comment on the sprint item itself.
* Attach any photos, documents, notes as documents to the Sprint item in the schedule.
* Any agreed upon action items can be created as tasks in the following sprint and assigned to their respective owner.

## Program governance

In addition to simply looking in Hansoft at the different feature burn downs, sprint burn downs and so on there are a number of reports that are defined in the template database to help with following up on progress and to identify features/teams that are struggling and need support. These reports are:

### Release status

This report shows the calculated Release risk for all stories that are not completed. The stories are grouped by Feature and Risk level and sorted by priority. For a description of the Release risk concept see *Customizations in the Program*.

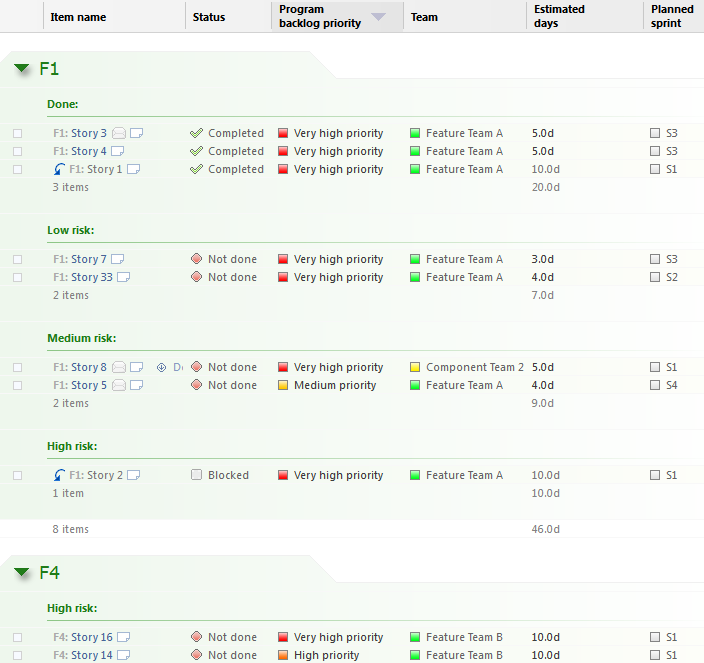


Figure 18 - Report: Release status

### Velocity by Sprint

This report shows the program level velocity per sprint.

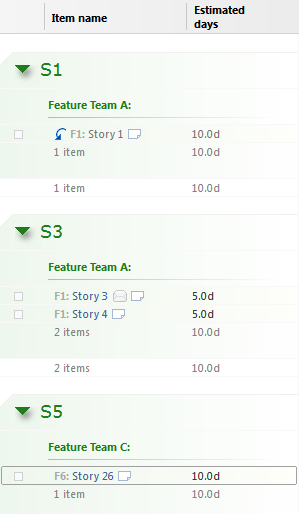


Figure 19 - Report: Velocity by sprint

### Velocity by Team

This report shows the velocity for each team over past sprints.

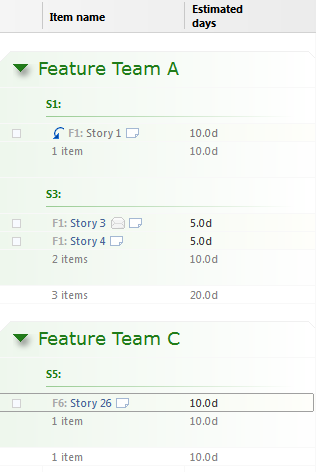


Figure 20 - Report: Velocity by Team

### Blocked/Started/Completed 24h

This report shows all product backlog items that have changed status to Blocked, Started, or Completed during the last 24 hours. It will give a quick overview of the progress and any impediments that have emerged over the last 24 hours.

### Blocked/Started/Completed 1w

This report shows all product backlog items that have changed status to either Blocked, Started, or Completed during the last week. It will give a quick overview of the progress and any impediments that have emerged over the last week.

### Features by Epic

This report shows a summary of all features in a program across the Development, Release planning, and Feature backlog sections of the program backlog. The features are grouped according to what epic in the portfolio they are linked to.

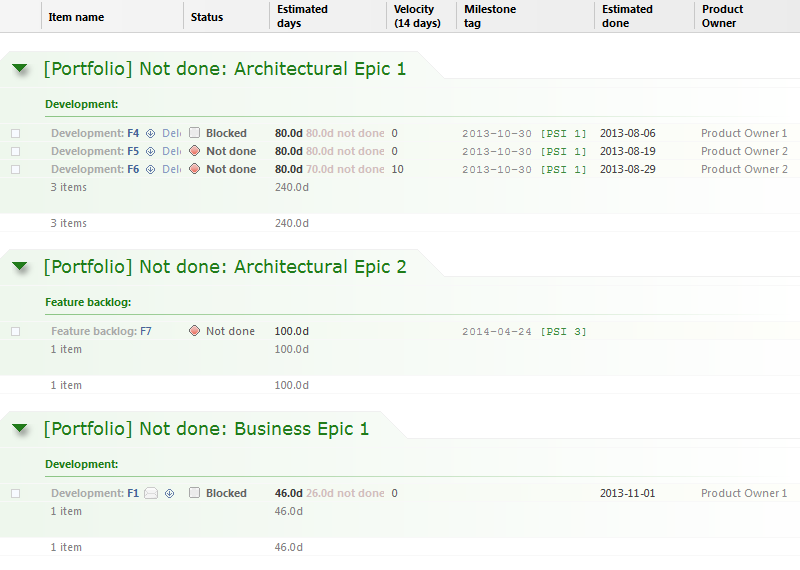


Figure 21 - Report: Features by Epic

## Customizations in the Program

The following custom columns are added in the programs:

* *Acceptance criteria* – This is a typical customization of Hansoft to capture the acceptance criteria (Test case slogans) for a user story.
* *Aggregated estimated days* – This field is automatically calculated as the sum of the Estimated days for all leaf items below this item. The field is typically not displayed to end users by default in their view presets but is used for aggregating information up to the Feature summary available for epics in the Portfolio project.
* *Aggregated status* – This field is automatically calculated as the aggregated status of all leaf items below this item. The field is typically not displayed to end users by default in their view presets but is used for aggregating information up to the Feature summary available for epics in the Portfolio project.
* *Completion count* – This field is automatically updated and shows the number of completed items and the total number of items below this item on the form. n-completed/n-total, e.g., 5/10.
* *Days completed* – This field is automatically calculated as the sum of the Estimated days for all completed leaf items below this item. The field is typically not displayed to end users by default in their view presets but is used for aggregating information up to the Feature summary available for epics in the Portfolio project.
* *Estimated done* – The idea with this field is that a primary responsibility of a Product Owner for a feature is to always have a clear understanding of the likely completion date for the feature in question. It is the Product Owners responsibility to keep this field updated on the feature level. The value will be displayed in the Feature summary and reports in the portfolio project.
* *Product Owner* – Set on the feature level to define which Product Owner is responsible for this feature. The value is used in the Feature summary for epics and reports in the portfolio project.
* *Release risk* – This field is automatically calculated based on the current date for the completion of the feature that the story belongs to (as defined by the feature release/milestone it is tagged to), remaining working days, and predicted velocity. Predicted velocity is calculated in the same way as in the Hansoft client based on the “Sprint prediction method” set in the project options. The release risk for each story is then set as follows:
  + Remaining stories for the feature are sorted by priority (in the same way as the priority view in the Hansoft client)
  + Predicted remaining capacity is calculated as predicted velocity (per day) times the remaining number of working days.
  + Items that fit within remaining capacity – 20% are set to Low risk.
  + Items that fit within remaining capacity + 20% are set to Medium risk.
  + Remaining items are set to High risk.
  + Optionally items that are in status Blocked can be automatically elevated to High risk.
  + Optionally the Release risk can automatically be set to the Risk level of Hansoft’s regular Risk column (for example used to capture technical risk) if it is at a higher level than what has been calculated as described in the preceding bullets.
  + The percentages, i.e., 20% above, can be changed as required.
* *Status last changed* – This field is automatically updated with the date/time when the status of the item was last changed. The field is typically not displayed to end users by default in their view presets but is primarily used for creating different kinds of reports. For example showing items completed/started/blocked in a time span like the last day or week.
* *Team* – Should always be set to the team that is responsible the feature/story. The field is used for reporting and also filtering out a team backlog, possibly across multiple features and components.

## Handling non-functional requirements

Non-functional requirements can either be related to a particular functional requirement or they can apply for the system as a whole.

### Non-functional requirements applicable for a specific functionality

Non-functional requirements that are connected to a particular use-case/user story should be documented together with that particular use-case/user story. This is typically done on the Acceptance criteria tab in Hansoft. Later on when you groom the backlog you can then split the user story into different stories that comply with the non-functional requirements on different levels. For example if the requirement is to run on different hardware, smartphones for example, you could split Android and iOS into separate stories. You can think in a similar way for other kinds of non-functional requirements such as performance, reliability and so on.

### Non-functional requirements for the system as a whole

Non-functional requirements that are valid for the system as a whole can for example be:

* Maximum allowable CPU usage given a certain usage load and hardware configuration
* All user facing text should be internationalized

Such non-functional requirements are best considered as part of the definition of done. They can be documented as part of the backlog or as document(s) in the document system in Hansoft. Typically you would have two different levels of definition of done. One that is applicable for individual stories/backlog items and one that is applicable for the feature as a whole. To track the progress against the definition of done you can add a multi-select custom column mapping to the different aspects of the definition of done to track this on a per story and/or feature level. You can then pull out reports that show status of the different aspects of the definition of done and use them at sprint reviews/demos and the like.

# Installing the database and the customizations

On a high level there are three steps to deploying the Hansoft Scaled Agile Support Kit.

1. Importing and configuring the template database
2. Installing and configuring Jean for Hansoft.
3. Installing the Hansoft SAFe client plugin.

To perform the following steps you need to have Administrator access to Hansoft and also have Windows administrative privileges on the machine where you intend to install Jean for Hansoft.

The Hansoft SDK license is required for installing the Hansoft Scaled Agile Support Kit and it is recommended that you first test and configure your Hansoft SAFe setup on a testing sandbox server. Please contact [support@hansoft.com](mailto:support@hansoft.com) to get a license for a sandbox server if you don’t already have one setup.

## Prerequisites

* The latest version on Hansoft Scaled Agile Support Kit[[7]](#footnote-7).
* Hansoft SDK License
* A server to run Jean for Hansoft to provide automation. Windows 2008 Server or later is required as the operating system. It is recommended to run Jean on a separate server from the Hansoft database server to isolate it from potential performance impacts. On this server the following Microsoft components needs to be installed
  + NET framework 4.5[[8]](#footnote-8)
  + Visual Studio 2012 VC Redist[[9]](#footnote-9) (Make sure to install the right version for your system. The x86 version is applicable for 32 bit systems and the x64 version is applicable for 64 bit systems).

## Importing the database

Use the Hansoft Server Administrator tool to import the template database as described under “Moving a database” in Chapter 1 of the *Hansoft System Administrator’s guide[[10]](#footnote-10)*.

## Preparing the database

The database contains generic information, like projects, epics, features, users and so on. These should be changed to reflect your particular setup. Be however careful to not change the structural aspects of the product backlog or the schedule as the predefined reports are depending on this structure.

### Adapting users and groups

There are a number of user groups and example users defined in the template database. You should modify and complement these to fit your organization.

#### Groups

A number of groups are defined in the template database and these groups are used for report sharing and delegation in the template database as follows.

|  |  |  |  |
| --- | --- | --- | --- |
| **Group name** | **Comment** | **Delegated rights** | **Shared reports** |
| Portfolio Management | - | Delegated to all themes in the Portfolio backlog. | All reports |
| Program Management | - | Delegated to all sections of the schedule and the backlog in the program projects. | All reports |
| Product Owners | - | Delegated to “their features” in the program backlogs and the parts of the schedule that belongs to “their teams”. | All reports |
| Scrum Masters | - | Same as “their teams”. | All reports |
| Feature Team A | This is an example feature team. | Delegated to “their” part of the schedule and “their” features in the program backlog. | All reports except velocity reports |
| Component Team 1 | This is an example component team. | Delegated to “their” part of the schedule and “their components” in the program backlog. | All reports except velocity reports |

#### Regular users

The following regular users are defined in the template database.

|  |  |  |  |
| --- | --- | --- | --- |
| **User name** | **Member of projects** | **Member of Groups** | **Main manager in** |
| Hansoft Admin | Portfolio, Program 1, Program 2 | - | Portfolio, Program 1, Program 2 |
| Portfolio Manager 1 | Portfolio, Program 1, Program 2 | Portfolio Management | - |
| Program Manager 1 | Program 1 | Program Management | - |
| Program Manager 2 | Program 2 | Program Management | - |
| Product Owner 1 | Program 1 | Product Owners | - |
| Product Owner 2 | Program 1 | Product Owners | - |
| Product Owner 3 | Program 1 | Product Owners | - |
| Product Owner 4 | Program 2 | Product Owners | - |
| Product Owner 5 | Program 2 | Product Owners | - |
| Release Train Engineer 1 | Program 1 | Scrum Masters, Program Management | - |
| Release Train Engineer 2 | Program 2 | Scrum Masters, Program Management | - |
| Scrum Master 1 | Program 1 | Scrum Masters, Feature Team A | - |
| Scrum Master 2 | Program 1 | Scrum Masters, Component Team 1 | - |
| Member 1 (Component Team 1) | Program 1 | Component Team 1 | - |
| Member 1 (Feature Team A) | Program 1 | Feature Team A | - |

#### SDK users

There is a single SDK user named *Jean* in the template database. The intent is that this user should be used for the Jean service. See *Installing and configuring Jean for Hansoft* for more details.

#### Passwords

All accounts have the password *welcome* in the template database. This includes regular users, the built-in *Administrator* user as well as the *Jean* SDK user.

### View presets

There are initial view presets defined in the database for the different user groups. When you have assigned the specific users in your organizations to groups you should review and re-apply these presets. The predefined view presets are:

* In the Portfolio backlog section of the Portfolio project there is a view preset defined for the group Portfolio Management.
* In the Program backlog section of the two Program projects there are two view presets, one for regular Program members (team members, scrum masters, product owners and so on) and one for the Program management and Portfolio management groups.

### Adapting custom columns

In the Program projects there are a few custom columns that need to be adapted to your specific environment. They are:

* Team – Edit/add values to the column definition to reflect the teams that work in each program
* Planned sprint – Edit/add values to the column definitions to reflect the set of sprints that you have in each PSI in your programs.

### Adapting the structure of schedules and backlogs

The example database is populated with placeholder data that should be replaced with your specific data as described below.

#### Portfolio project

In the Portfolio backlog of the Portfolio project you need to remove the existing data and replace with your own investment themes and epics. Note that investment themes are top-level items and that epics are child items of these. When features are added in the programs they should be linked to the epics as described in *Adding features to Epics*.

#### Program projects

Rename the existing Program projects, Program 1 and Program 2 and add new ones to match up with your organization. Note that you will need to replicate reports and view presets to any additional projects you create.

In the Schedule part of the program projects you need to add/update the existing PSI milestones and Feature milestones in the Roadmap section of the schedule to be representative of your specific programs. The Features and Component teams sections of the schedule needs to be updated too to be representative of the features and components you have under development and what teams are working on them.

In the Program backlog part of the program projects you need to replace everything below the top level containers called *Development*, *Release planning*, and Feature backlog with the features and stories that are applicable for your programs. Be careful to tag any stories you create to their appropriate PSI and feature milestones.

## Installing the SAFe Hansoft client plugin

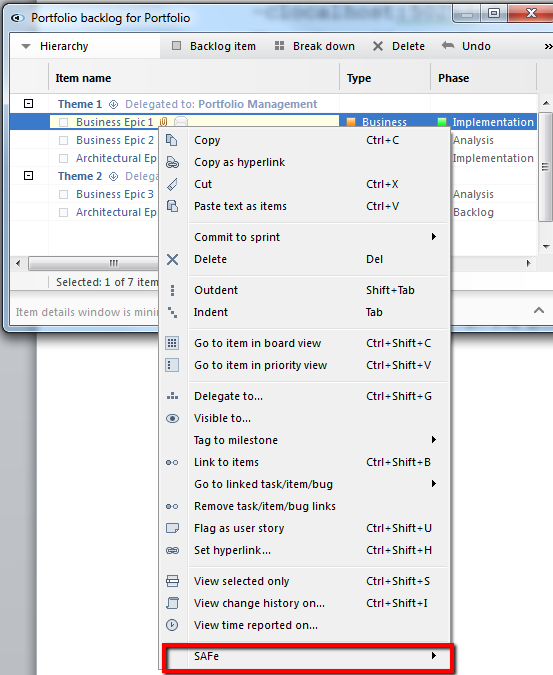
1. Copy the Plugin and PluginInstaller directories in the Hansoft Scaled Agile Support Kit to local disk. It does not matter where you put them or on which machine but make sure to keep them in a place where you can return and reinstall/uninstall the plugin if needed.
2. Open a command prompt and navigate to the Plugin directory.
3. Edit the files Install.bat and Uninstall.bat so that the –c option is correct for your environment. Example:  
   -clocalhost:50256:"SAFe - Template":Jean:welcome  
   This will install/uninstall the plugin on the Hansoft server running on the local machine on port 50256 in the database “SAFe – Template”. The Hansoft SDK user with name Jean and password welcome will be used for the installation. You need to change these settings to reflect your Hansoft setup.
4. Run the Install.bat script. The first time you do this a message saying that a file could not be found will be displayed. This message can safely be ignored.
5. Check that the plugin is installed by logging in to Hansoft and open up a project in the database where the plugin was installed. There should now be a new menu item called SAFe at the bottom of the context menu in the Product backlog.  
   

Figure 22 - The SAFe context menu

1. To uninstall the plugin run the Uninstall.bat script.

## Installing and configuring Jean for Hansoft

1. Download Jean and install it on a suitable server as described in the documentation for Jean[[11]](#footnote-11). (Note: if you download executable files or zip-files with executable files in them from the internet Windows will mark them as unsafe by default. To overcome this right-click on the file in question and select properties from the menu. In the properties dialogue then click unblock. If you don’t do this you will get strange error messages in the event log when starting Jean)
2. Copy SAFeExtension.dll found in the Jean directory of the Hansoft Scaled Agile Support Kit to the directory where you have put the Jean executable.
3. Based on the example JeanSettings.xml file in the Jean directory of the Hansoft Scaled Agile Support Kit, create a JeanSettings.xml file that is adapted for your server and setup. There are comments in the JeanSettings.xml file that describes this in detail.
4. Restart the Jean service and make sure that columns are updated as they should. Also check the Application Log under Windows Logs in the Windows Event viewer for any Jean related messages. If Jean starts successfully the two last messages from Jean should be “Jean was loaded” and “Jean was started”.

1. See: http://www.scaledagileframework.com [↑](#footnote-ref-1)
2. See: <http://github.com/Hansoft/Hansoft-Jean-Jean> [↑](#footnote-ref-2)
3. See: <http://github.com/Hansoft/SAFeKit/blob/master/LICENSE.md> [↑](#footnote-ref-3)
4. An alternative to this is to systematically apply WSJF based prioritization systematically as described in **Error! Reference source not found.**. [↑](#footnote-ref-4)
5. <http://www.hansoft.com/manuals/71/English/projects_more_out-of-office-planning.htm> [↑](#footnote-ref-5)
6. <http://www.hansoft.com/support/videos/> [↑](#footnote-ref-6)
7. <http://github.com/Hansoft/SAFeKit/> [↑](#footnote-ref-7)
8. <http://www.microsoft.com/en-us/download/details.aspx?id=30653> [↑](#footnote-ref-8)
9. <http://www.microsoft.com/en-us/download/details.aspx?id=30679> [↑](#footnote-ref-9)
10. <http://www.hansoft.com/uploads/PDF/HansoftSysAdminGuide.pdf> [↑](#footnote-ref-10)
11. <http://github.com/Hansoft/Hansoft-Jean-Jean> [↑](#footnote-ref-11)