Macro-Micro Roll-up

User Guide

Version 1.0

Developed by lateralworks

lateralworks >

Introduction

When managing a very large program (where the may be >10,000 tasks), there are multiple options available to better mange such complexity. Each option has its own pros and cons. They are to manage:

1. All tasks in a single schedule
2. Sub-projects embedded in a master schedule
3. Macro tasks in a single schedule with detail in sub-projects, linked through “virtual dependencies” (macro-micro)

The pros and cos for each approach are discussed.

Overview of Different Methods to Manage Large Programs

1. Manage all Tasks in a Single Schedule

Clearly the simplest approach. A single program manager manages a single schedule that is refreshed weekly. The program manager meets with each of the sub-teams to scrub their part of the schedule.

Although this appears to be the optimum approach (and for the majority of programs it is), there is a point when this becomes unwieldy as the number of sub-teams and tasks goes beyond a certain point. The problems can be compounded if the sub-teams are geographically separated, especially across multiple time zones. The program manager then ends up spending a significant part of their time simply managing the schedule and keeping it up to date, where what they should really be focused on is determine how best to accelerate the schedule. Teams get lost in the detail and with many tasks on the critical path, it becomes difficult to pull-in.

Pros

- All tasks are in one schedule, eliminating the complexity of managing multiple schedules
- Changes in any task are seen instantly in the schedule
- The tasks will follow the natural flow of the program
- Only one program manager needs to be trained to manage the schedule

Cons

- The schedule gets so detailed it becomes difficult to see where strategic pull-ins can be made
- There is no distributed ownership and management of the schedule
- The program manager can be overloaded if the number of sub-teams is large
- The information bottleneck is the single program manager

2. Embed Sub-projects in a Master Schedule

This idea is very appealing. The schedule is partitioned (either all or in part) into a number of sub-projects. Each subproject is separately managed by its own project manager. A “master” schedule glues all the schedules together and is managed by an overall program manager. Dependencies between the master schedule and sub-projects and between sub-projects are made as any other dependency are made.

Pros

- Sub-projects allow sub-teams to manage their project and the detail independently of the program manager
- Changes in any task in the sub-projects are instantly seen in all schedules
- The workload is distributed across the program manager and project managers, relieving the program manager of “managing everything”
- Ownership and management of the schedule if distributed

Cons

- Sub-projects need to follow the natural flow of the project, which (practically) can be difficult, and can create more problems than it attempts to solve
- Pull-in can still be difficult because the detail is still inherently there

- Some coordination is needed as part of the refresh - ideally all schedules would be updated on the same day
- A thorough understanding and discipline is needed to mange the master schedule and the sub-projects, otherwise the files can very quickly get littered with broken links (links between sub-projects) that can cause numerous problems
- Multiple project managers need to be trained in proper scheduling practices
- All the project tasks are within one integrated schedule, which be comes difficult with tens of thousands of tasks
- The critical path flows through every micro task, making to too big to follow and of little value when trying to pull-in the schedule

3. Manage Sub-projects Through Virtual Dependencies (Macro-micro Roll-up)

This is similar in concept to embedding sub-projects in a single schedule in the sense that parts of a larger program schedule are split between sub-projects. Where this approach differs is the “master” schedule is really a macro-schedule (typically only contains macro-level tasks), whereas the sub-projects (micro-schedules) contain the detail for the macro-level tasks. So rather than splitting a program into large sub-projects, this approach is much finer grained and the sub-projects are split at the task level.

This has one huge advantage in that the sub-projects do not have to follow the flow of the macro-schedule. Thus, the macro-schedule can be an integrated cross-functional schedule, whereas the micro-schedules can be a functional schedule (or whatever other partitioning is desirable). The detail is not see at the macro-level, but rather than being a disadvantage, this turns out to be an advantage, as teams are now focused on the bigger picture macro-schedule.

The purpose of the macro-schedule is to represent the macro tasks of a program that follow the natural flow of the program. Micro-schedules on the other hand contain the detailed tasks for the macro tasks, and therefore are not constrained to follow the flow of the macro-schedule.

Synchronization of the macro-schedule and micro-schedules (typically in the form of dates and durations) are occurs through a roll-up/roll-down procedure, where dates required by the micro-schedules are “rolled down” from the macro-schedule to the micro-schedules, and durations or dates from the micro-schedules to the macro-schedule are “rolled up”. This makes the process a little more computationally complex, but is out weighed by the benefits of being able to manage the overall program as one integrated and interconnected system.

Pros

- The macro-schedule is high-level, allowing the team to focus on finding strategic pull-ins
- Micro-schedules allow sub-teams to manage their detailed tasks independently of the program manager
- The micro-schedules do not need to follow the project flow, so can be e.g. functional
- The workload is distributed across the program manager and project managers, relieving the program manager of "managing everything”
- Ownership and management of the schedule if distributed
- Critical path can be seen at the macro level (a few tasks) while it can also be seen at the micro level (many detailed tasks, specific to each Module)

Cons

- Changes in any of the schedules are not instantly seen - a roll-up has to be performed to see the impact of changes
- An extra step is added to the refresh process to perform the roll-up
- If the detail is needed to be seen, additional schedules need to be opened
- Some coordination is needed as part of the refresh: ideally you would want all scheduled to be updated on the same day and rolled-up on the same day
- Multiple project managers need to be trained in proper scheduling practices
- Strict procedures and rules must be followed to maintain the databases (however, this is simplified through data validation software in the roll-up/down system)

At the start of a program, there will only be just one schedule - the macro-schedule. This will likely exist in this form for many weeks or even months. At some point, if the team gets too large or the schedule gets too cumbersome to manage, one of the other two different approaches would be adopted. There are no hard rules about which method to choose. From experience however, we tend to use these guidelines:

- For small-medium size projects, we will choose the first method - it is by far the easiest to implement.
- If suppliers are involved and they already have Microsoft Project schedules (albeit some clean-up work may be needed), and the schedules (at least roughly) follow the natural flow of the program, we would select the second method.
- For mega/large programs (e.g. build and start-up a new fab), we would select the third method. One recent project had >35,000 tasks, so managing this as a single schedule was unrealistic. By parsing the detail of the macro-tasks out to functional sub-teams, the detail could was managed separately by the sub-teams, while the PM and core team focused on the bigger picture (macro-level) schedule looking for big or strategic pull-ins.

Rules for Setting up a Macro-micro Roll-up

Setting up a Macro-micro Roll-up is relatively straightforward. The only thing the user has to focus on is setting up the tasks in the micro-schedule to roll-up to the macro-schedule. The roll-up settings will be setup automatically on the first roll-up (or when needed) by fastProject.

Before describing the setup in detail, a set of rules need to be established:

1. **Task names in both the macro-schedule and micro-schedules must be identical, including capitalizations and spaces**. For example “macroA” in macro schedule is the summary of “microA” tasks in the micro schedule.
2. ***Task names must be unique in both the macro-schedule and micro-schedules, i.e. only one version of the task name must exist***. And following from above, for example there can only be one “macroA” in macro schedule and one “microA” tasks in the micro schedule. We can’t have another group of micro tasks summarized in the micro schedule with the “microA” nor can there by another “macroA” in the macro schedule.
3. Roll-up tasks must be auto-scheduled, i.e. they cannot manually scheduled tasks.
4. A roll-up task in the micro-schedule can be a task or a summary task. The corresponding task in the macro-schedule must be a task.
5. Roll-up tasks in both the macro-schedule and micro-schedule must be part of a WBS group, i.e. they must be contained within a summary-milestone group.
6. A dependency from a Touchpoint to a roll-up task in the micro-schedule must also exist in the macro-schedule (the dependency can be from a Touchpoint to roll-up task or from a Touchpoint to a sub-task of a roll-up task).

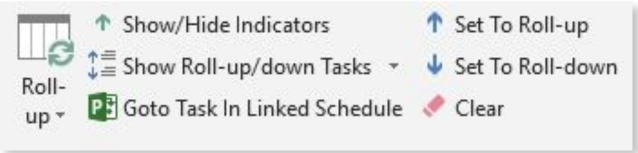
There are a few additional rules, but typically the user does not need to be concerned with these as they are handled automatically if the “Automatically setup” flag is checked during the roll-up.

1. If a task is marked as a roll-up in the micro-schedule, it must also me marked as a roll-up in the macro-schedule.
2. If a task is marked as a roll-down in the macro-schedule, it must also me marked as a roll-down in the micro-schedule.
3. The predecessor of a roll-up task in the macro-schedule must also exist in the micro-schedule
    - This is typically in the form of another roll-up task in the micro-schedule or from a Touchpoint with the same name as the predecessor in the macro-schedule
    - Note that the reverse is not true. A predecessor of a roll-up task in the micro-schedule does not exist in the macro-schedule. In this case, the predecessor task would be local to the micro-schedule. If this task ends up driving the start date of the roll-up task, it is done by applying a date constraint to the roll-up task in the macro-schedule.
4. If a dependency exists between roll-up tasks in the macro-schedule, they must also exist in the micro-schedule.

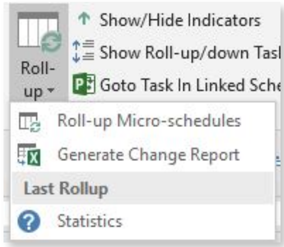
Note that the roll-up process is independent of the calendar 5-day/7-day week. Just bear in mind that if different calendars are used, some of the dates may be off by 1-2 days. We always recommend adopting the same calendar for all schedules. In other words, use either a 5 day calendar (Monday through Friday) or a 7 day calendar (Monday through Sunday). Don’t mix calendars between macro and micro schedules, use the same one at each level. The software works best with 7 day calendars at all levels.

Macro-micro Roll-up Interface

To support the management of the Macro-micro Roll-up, a range of tools have been developed. These can be found in the Refresh tab in fastProject as seen below.

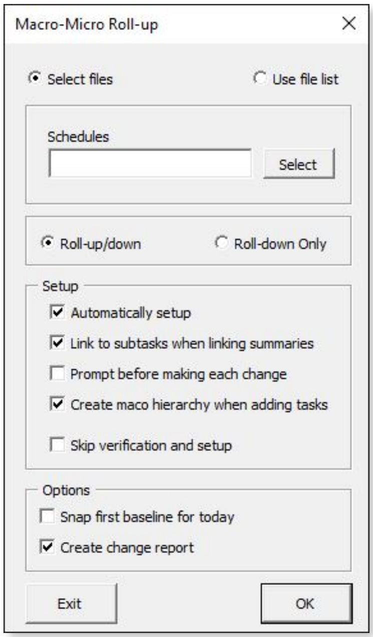


- **Roll-up**: Performs the Macro-micro Roll-up/down. There are some additional options:
  - **Generate Change Report**: Will generate a change report (based on changes since just before the last roll-up)
  - **Statistics**: Displays statistics from the last roll-up (number of files and time taken)



- **Show/Hide Indicators**: Shows/Hides a column indicating whether a task is a Roll-up task (up arrow) of a Roll-down task (down arrow)
- **Show Show-up/down Tasks**: Shows only (filters) those tasks that are either a Roll-up or a Roll-own task. A sub-menu allows you to show only Roll-up or Roll-down tasks.
- **Goto Task in Linked Schedule**: If on a Roll-up or Roll-down down in the macro-schedule, will open (if needed) and go to the same task in the micro-schedule. If on a Roll-up or Roll-down down in the micro-schedule, will open (if needed) and go to the same task in the macro-schedule.
- **Set To Roll-up**: Sets a task as a Roll-up task
- **Set To Roll-down**: Sets a task as a Roll-down task
- **Clear**: Clears the task as a Roll-up or Roll-down task

On clicking Roll-up, there user is presented with the following interface:



● Select files ○ Use file list

Schedules

● Roll-up/down ○ Roll-down Only

Setup

☑ Automatically setup

☑ Link to subtasks when linking summaries

□ Prompt before making each change

☑ Create maco hierarchy when adding tasks

□ Skip verification and setup

Options

□ Snap first baseline for today

☑ Create change report

- **Select files**: The user selects the micro-schedules to roll-up
- **Use file list**: The user selects a text file which contains the locations of the micro-schedules - this has the advantage that the micro-schedules do not need to be in the same location (they do when Select files is used)
- **Schedules**: By clicking the Select button, a dialog box is opened to other select the micro-schedules or select the file list.
- **Roll-up/down**: Performs the Roll-up/down procedure (we will do this next)
- **Roll-down Only**: Only performs a Roll-down from the opened schedule to the selected schedules
- **Setup**: Options to set the Roll-up and Roll-down:
  - **Automatically setup**: Automatically sets up the Macro-micro Roll-up when needed, such as adding Touchpoints, making dependencies, etc. This is checked by default. If not checked and there are problems in the setup, they will be reported to the user, where the user will then have to set up the Macro-micro Roll-up manually.
  - **Link to subtasks when linking summaries**: When a dependency exists between two macro tasks, that dependency also needs to be made in the micro-schedule. However, those tasks in the micro-schedule may be summary tasks (that we don't want to link to). This setting applies the following rules:
    - If the task is a summary task and is a predecessor task, the last task (milestone) is used
    - If the task is a summary task and is a successor task, the first task is used
  - **Prompt before making each change**: This simply prompts the user what change is about to be made, and the user has the option of accepting the change, rejecting the change or cancelling. This would typically only be used after the initial setup so the user can see the any changes before they are made.

  - **Create macro hierarchy when adding tasks**: When a dependency exists between a task (that is not part of the micro-schedule) and a roll-up macro-task in the macro-schedule, a Touchpoint is created in the micro-schedule with the same name as the task and a date representing the Finish date of the task (this would be marked as a roll-down milestone). When the Touchpoint is added and this option is selected, the same hierarchy (the task's summary tasks) is created before being added. If the hierarchy already exists in the micro-schedule, it simply adds the Touchpoint to that hierarchy. In addition, the hierarchies and Touchpoints are added in the same order as it exists in the macro-schedule, so there is always a 1:1 mapping between the structure of the macro-schedule and the micro-schedule. If this option is unchecked, all Touchpoints added under a single summary that is placed as the first activity in the schedule.
  - **Skip verification and setup**: By default, the macro-schedule and all micro-schedules are verified that there are no errors (see Rules for Setting up a Macro-micro Roll-up) for the checks that are made). Any errors are listed in an error report and the roll-up cancelled. If any setup changes are made, the user is prompted if they want to review the changes before the roll-up (or the user has he option to cancel). This setting should always be on by default as it does not take a significant amount of time to perform the error checking. I should only be unchecked if you are 100% sure there are no errors and no setup is needed and there is a significant time saving.
- **Options**
  - **Snap first baseline for today**: Snaps the baseline before the roll-up. By default this is not checked as the roll-up is usually the last step in the refresh. However, it is possible to set an additional baseline specifically for the roll-up if needed.
  - **Create change report**: Creates a detailed list of everything that changed in the macro-schedule due to the roll-up. It lists each task that changed, either in it's finish date or duration, how much it changed and the micro-schedule that caused the change.

Even though there are multiple options, the defaults have been set such that the normal steps are:

1. Click Roll-up
2. Select the schedules or the file list
3. Click OK

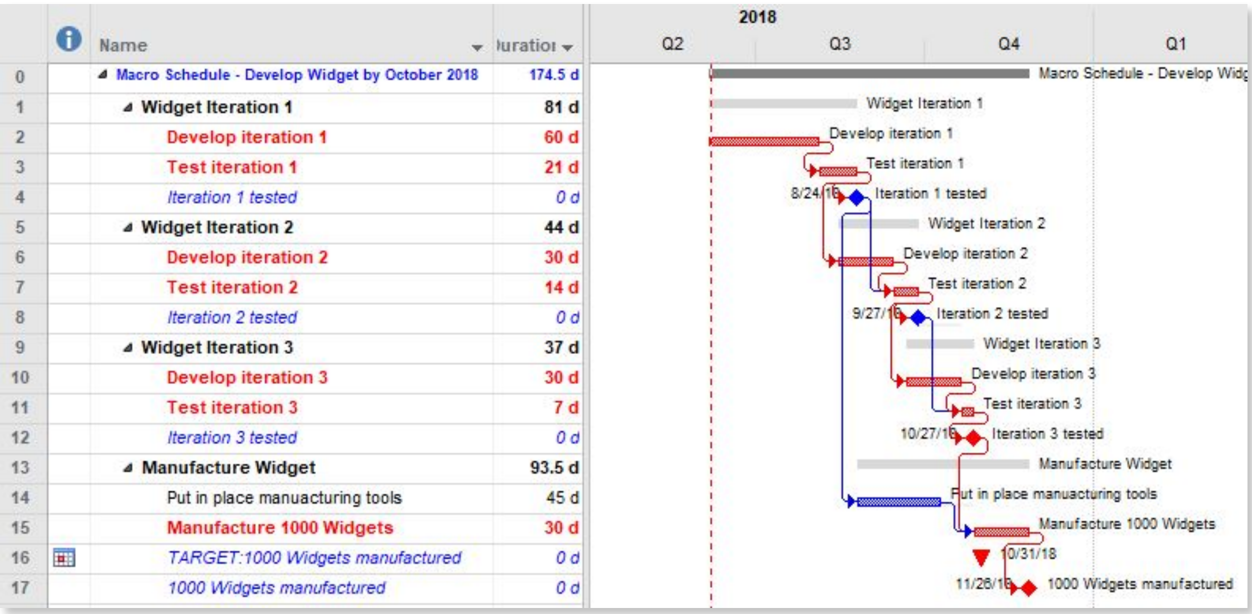
What follows is a case study showing using the Macro-micro Roll-up.

A Case Study Using the Macro-micro Roll-up

The ACME Corporation has decided to build a new Widget. After some preliminary research, the core team consisting of the PM, marketing, development, test and manufacturing got together to map out a high-level schedule. They determined it will take 3 iterations of development and testing before it can be released to manufacturing, and will declare success once 1000 Widgets have been manufactured. They need to be ready by October 2018, in time for the holiday season.

The first schedule was relatively risk-free, starting the next iteration after the previous one. On reviewing the critical path and seeing how late they were, they pulled-in the schedule by assuming some risk and starting the next iteration half-way through the previous iteration's testing cycle. They also concluded that after iteration 1, they should have enough information to start the process of developing the tools for manufacturing, but not until after the completion of the last test cycle do they want to commit to releasing the Widget to manufacturing.

The resulting macro-schedule is shown below. The team recognizes there is still a gap of about a month, but feel that if things go well, they may not need Iteration 3, in which case they'd be able to hit the target date of October 2018.

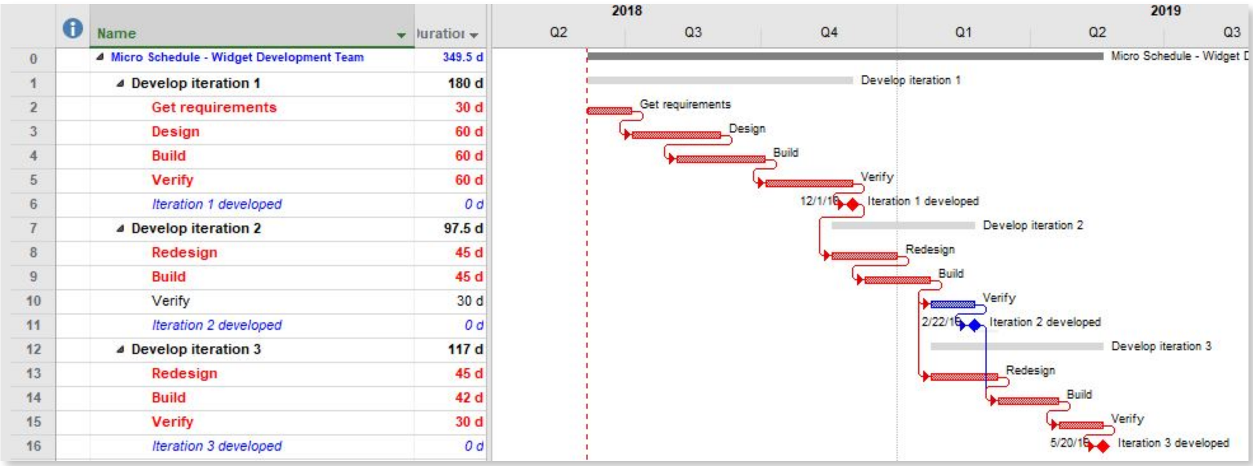


	ℹ	Name	uratio
0		◢ Macro Schedule - Develop Widget by October 2018	174.5 d
1		◢ Widget Iteration 1	81 d
2		Develop iteration 1	60 d
3		Test iteration 1	21 d
4		*Iteration 1 tested*	0 d
5		◢ Widget Iteration 2	44 d
6		Develop iteration 2	30 d
7		Test iteration 2	14 d
8		*Iteration 2 tested*	0 d
9		◢ Widget Iteration 3	37 d
10		Develop iteration 3	30 d
11		Test iteration 3	7 d
12		*Iteration 3 tested*	0 d
13		◢ Manufacture Widget	93.5 d
14		Put in place manuacturing tools	45 d
15		Manufacture 1000 Widgets	30 d
16	▦	*TARGET:1000 Widgets manufactured*	0 d
17		*1000 Widgets manufactured*	0 d

There are three functional teams on the project: Development, Test and Manufacturing. Splitting the schedule into sub-projects would only work if it followed the natural flow of the program, i.e. splitting by iteration. However, the teams want to manage their schedules functionally so they decide to adopt the Macro-micro Roll-up approach. Each team will manage their own functional schedule and the detail within their schedule. Thus, the development team will be responsible for the "Develop iteration x" macro-level tasks, the Test team will be responsible for the "Test iteration x" macro-level tasks, and the Manufacturing team will be responsible for all tasks under "Manufacture Widget". Each week, each team will report back to the macro-schedule the dates/duration of each of their macro-tasks.

The Development team got together the following day and mapped out their work in more detail, specifically, they made sure that the schedule was realistic. The image below depicts the resulting Development schedule (the Test and Manufacturing schedules would look similar). Note that:

1. The task names (Develop iteration x) only exist once in the macro-schedule and micro-schedule and match exactly.
2. The schedule only contains the development tasks. When we perform the roll-up for the first time, the test and manufacturing dependencies will be added.
3. It is immediately obvious that the timeline does not match the macro-schedule, i.e. the initial durations were initially underestimated. On top of this, adding the Test and Manufacturing dependencies after the roll-up may even make the schedule even later.

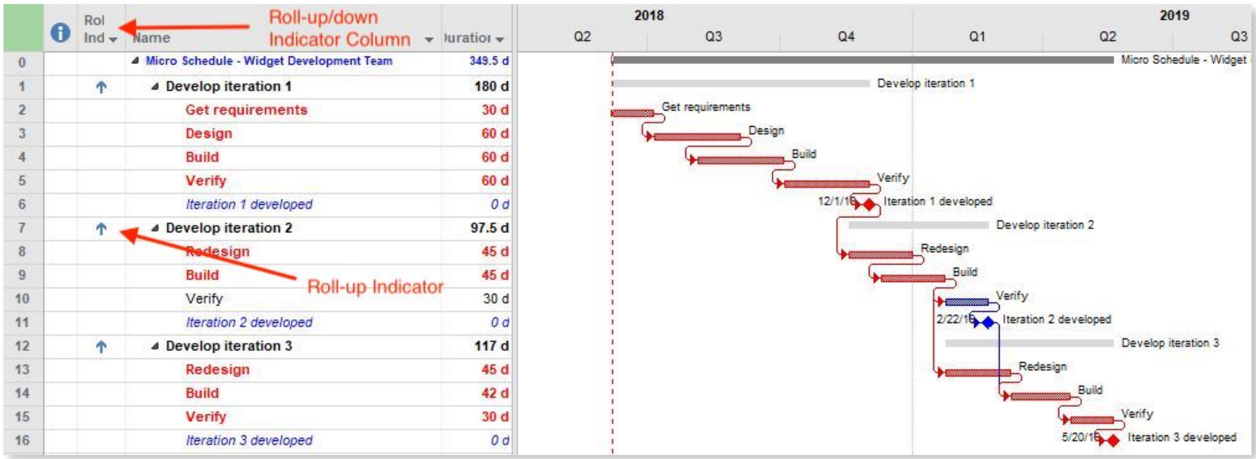


	Name	ɟuratioɾ
0	Micro Schedule - Widget Development Team	349.5 d
1	Develop iteration 1	180 d
2	Get requirements	30 d
3	Design	60 d
4	Build	60 d
5	Verify	60 d
6	*Iteration 1 developed*	*0 d*
7	Develop iteration 2	97.5 d
8	Redesign	45 d
9	Build	45 d
10	Verify	30 d
11	*Iteration 2 developed*	*0 d*
12	Develop iteration 3	117 d
13	Redesign	45 d
14	Build	42 d
15	Verify	30 d
16	*Iteration 3 developed*	*0 d*

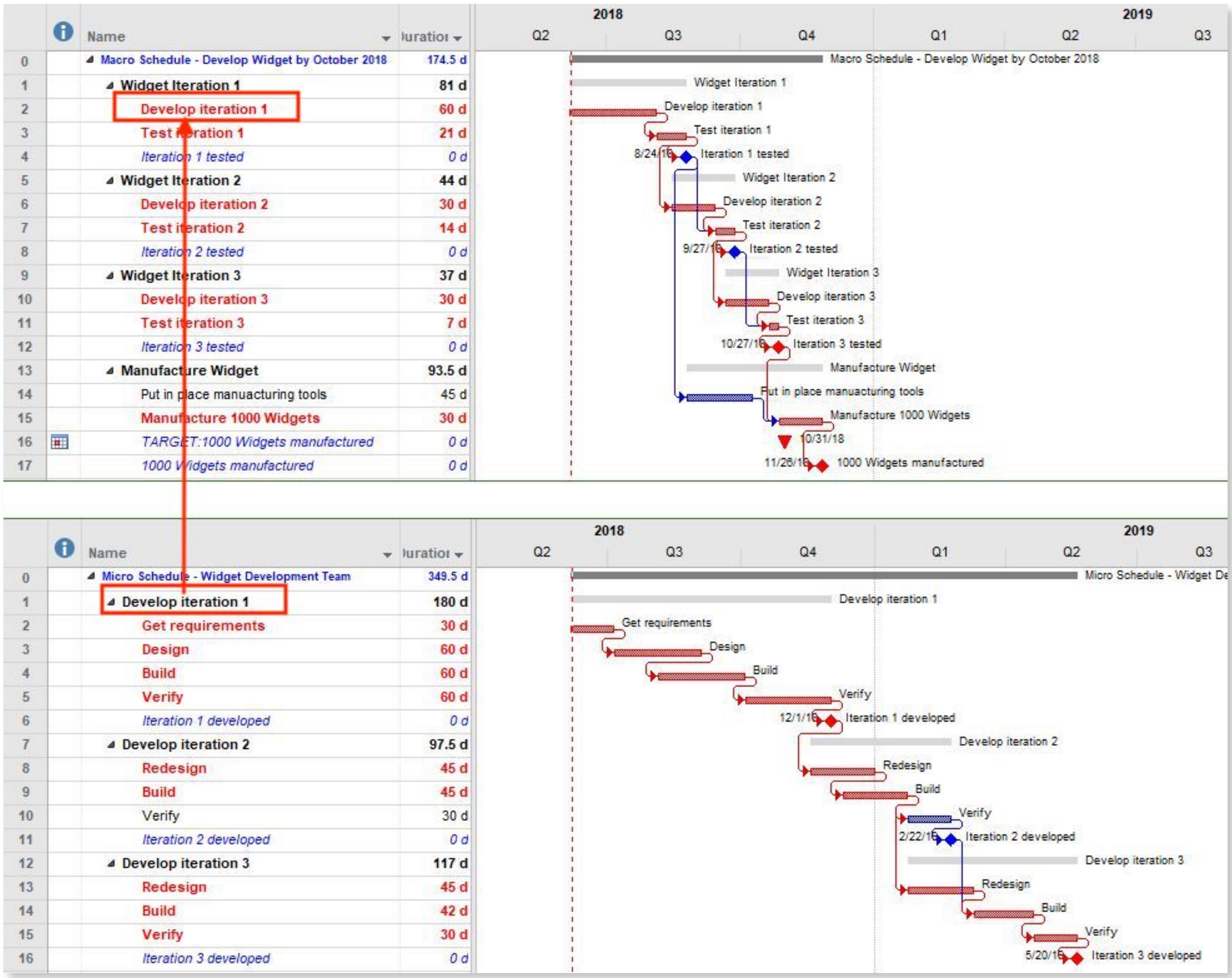
Now that we have a macro-schedule and our micro-schedule(s) (at least for Development), we need to set the Development (micro-) schedule up for roll-up. As we mentioned earlier, we only need to concern ourselves with setting up the roll-up tasks in the Development schedule. The Macro-micro Roll-up will handle the rest. We do this as follows:

4. Click “Show/Hide Indicators” to display the Roll-up/Roll-down indicator column (optional but useful so you can see what tasks are being rolled-up and which are being rolled-down)
5. Select the summary tasks “Develop iteration 1”, “Develop iteration 2” and “Develop iteration 3” and click “Set To Roll-up”.

That’s it! The result is shown below.



	Rol Ind	Name	uratio
0		Micro Schedule - Widget Development Team	349.5 d
1	↑	Develop iteration 1	180 d
2		Get requirements	30 d
3		Design	60 d
4		Build	60 d
5		Verify	60 d
6		*Iteration 1 developed*	0 d
7	↑	Develop iteration 2	97.5 d
8		Redesign	45 d
9		Build	45 d
10		Verify	30 d
11		*Iteration 2 developed*	0 d
12	↑	Develop iteration 3	117 d
13		Redesign	45 d
14		Build	42 d
15		Verify	30 d
16		*Iteration 3 developed*	0 d



		Name	uratio
0		Macro Schedule - Develop Widget by October 2018	174.5 d
1		Widget Iteration 1	81 d
2		Develop iteration 1	60 d
3		Test iteration 1	21 d
4		*Iteration 1 tested*	0 d
5		Widget Iteration 2	44 d
6		Develop iteration 2	30 d
7		Test iteration 2	14 d
8		*Iteration 2 tested*	0 d
9		Widget Iteration 3	37 d
10		Develop iteration 3	30 d
11		Test iteration 3	7 d
12		*Iteration 3 tested*	0 d
13		Manufacture Widget	93.5 d
14		Put in place manuacturing tools	45 d
15		Manufacture 1000 Widgets	30 d
16	▦	*TARGET:1000 Widgets manufactured*	0 d
17		*1000 Widgets manufactured*	0 d

		Name	uratio
0		Micro Schedule - Widget Development Team	349.5 d
1		Develop iteration 1	180 d
2		Get requirements	30 d
3		Design	60 d
4		Build	60 d
5		Verify	60 d
6		*Iteration 1 developed*	0 d
7		Develop iteration 2	97.5 d
8		Redesign	45 d
9		Build	45 d
10		Verify	30 d
11		*Iteration 2 developed*	0 d
12		Develop iteration 3	117 d
13		Redesign	45 d
14		Build	42 d
15		Verify	30 d
16		*Iteration 3 developed*	0 d

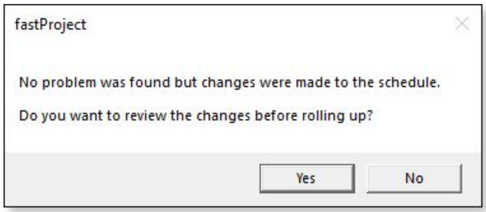
The image below shows both schedules before the roll-up. The top schedule is our integrated cross-functional macro-schedule. The bottom schedule is the Development team's micro-schedule. "Develop iteration 1" in the micro-schedule is a summary task with the detail as subtasks. When we perform the roll-up, the timeline of "Develop iteration 1" in the macro-schedule will adjust to that of the micro-schedule. As we have not rolled-up the schedules yet, the timelines don't match. For example, the project in the macro-schedule finishes in November 2018, but "Develop iteration 3" in the micro-schedule doesn't finish until May 2019. The team who built the initial macro-schedule clearly underestimated how long the development tasks would take.

The next step is to perform the Macro-micro Roll-up. As this is the first time we will be performing a roll-up, it will go through an initial setup. For future roll-ups, it will only perform a setup as needed.

To perform the roll-up the steps are:

6. Open the macro-schedule
7. Click Roll-up in the Refresh tab
8. Select the Development (micro-) schedule - we would also have selected the Teat and Manufacturing schedules if available
9. Leave the defaults as-is
10. Click OK

As this is the first roll-up and changes were made to the schedules, we are prompted with the following.

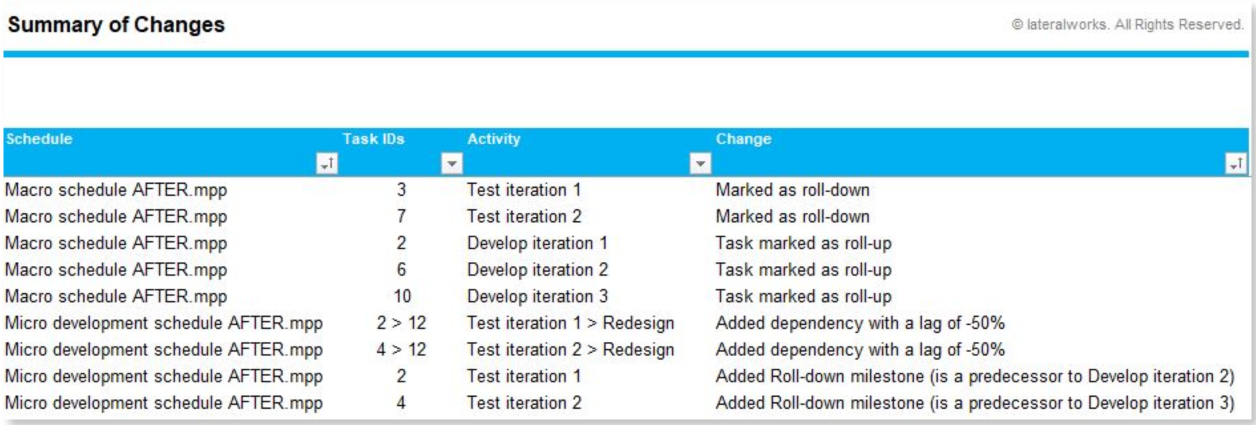


fastProject

No problem was found but changes were made to the schedule.

Do you want to review the changes before rolling up?

Clicking Yes opens up a report detailing all the changes that were made (see below). If there were any problems that prevented the roll-up, these would also be displayed on a separate sheet.

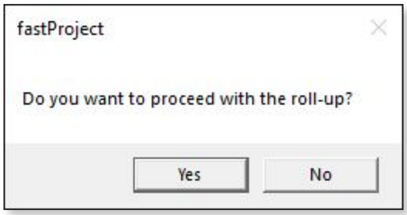


Summary of Changes

© lateralworks. All Rights Reserved.

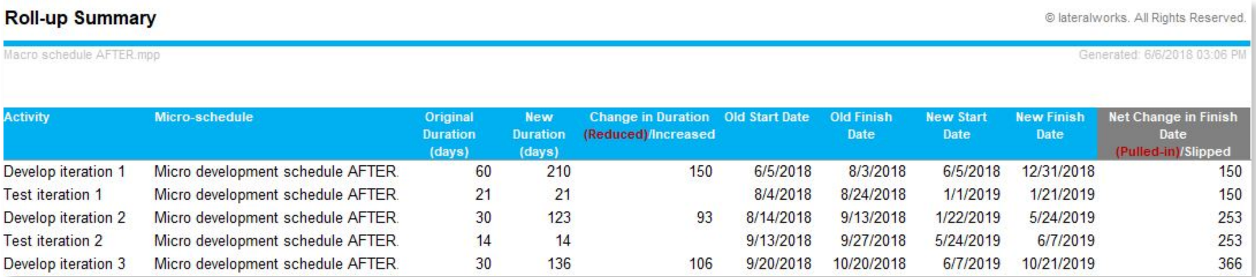
Schedule	Task IDs	Activity	Change
Macro schedule AFTER.mpp	3	Test iteration 1	Marked as roll-down
Macro schedule AFTER.mpp	7	Test iteration 2	Marked as roll-down
Macro schedule AFTER.mpp	2	Develop iteration 1	Task marked as roll-up
Macro schedule AFTER.mpp	6	Develop iteration 2	Task marked as roll-up
Macro schedule AFTER.mpp	10	Develop iteration 3	Task marked as roll-up
Micro development schedule AFTER.mpp	2 > 12	Test iteration 1 > Redesign	Added dependency with a lag of -50%
Micro development schedule AFTER.mpp	4 > 12	Test iteration 2 > Redesign	Added dependency with a lag of -50%
Micro development schedule AFTER.mpp	2	Test iteration 1	Added Roll-down milestone (is a predecessor to Develop iteration 2)
Micro development schedule AFTER.mpp	4	Test iteration 2	Added Roll-down milestone (is a predecessor to Develop iteration 3)

This shows the details of all changes made to the both schedules: the name of the schedule, the Task ID, the name of the task and a description of the change. As changes were made, we are also prompted whether we want to continue with the roll-up. Click Yes.


fastProject

Do you want to proceed with the roll-up?

A summary of the changes (see below) to each schedules after the roll-up is shown below.


Roll-up Summary

© lateralworks. All Rights Reserved.

Macro schedule AFTER.mpp

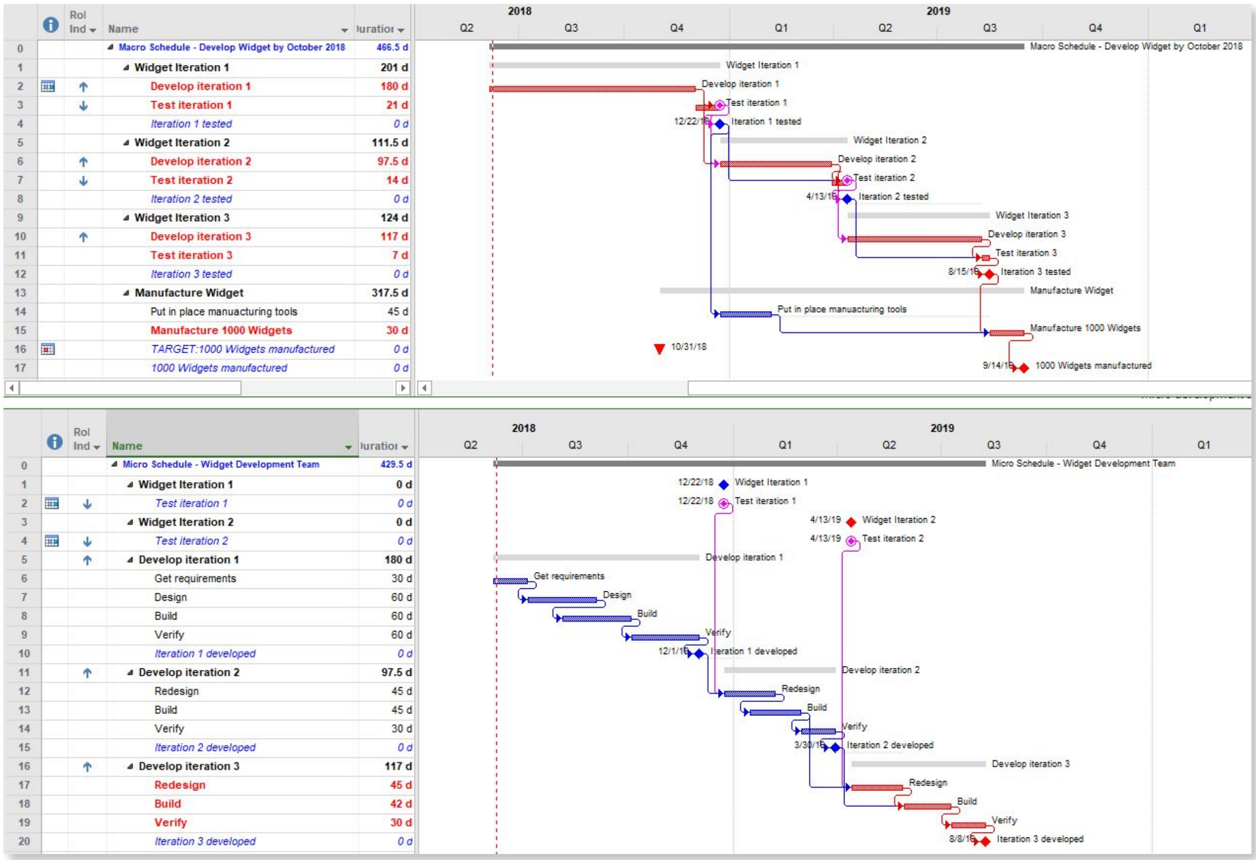
Generated: 6/6/2018 03:06 PM

Activity	Micro-schedule	Original Duration (days)	New Duration (days)	Change in Duration (Reduced)/Increased	Old Start Date	Old Finish Date	New Start Date	New Finish Date	Net Change in Finish Date (Pulled-in)/Slipped
Develop iteration 1	Micro development schedule AFTER.	60	210	150	6/5/2018	8/3/2018	6/5/2018	12/31/2018	150
Test iteration 1	Micro development schedule AFTER.	21	21		8/4/2018	8/24/2018	1/1/2019	1/21/2019	150
Develop iteration 2	Micro development schedule AFTER.	30	123	93	8/14/2018	9/13/2018	1/22/2019	5/24/2019	253
Test iteration 2	Micro development schedule AFTER.	14	14		9/13/2018	9/27/2018	5/24/2019	6/7/2019	253
Develop iteration 3	Micro development schedule AFTER.	30	136	106	9/20/2018	10/20/2018	6/7/2019	10/21/2019	366

On completion of the roll-up, all micro-schedules are saved and closed and the macro-schedule is left open. Note the new timeline (see below), specifically:

- The durations of the “Develop iteration” tasks in the macro-schedule are aligned with the micro-schedule
- The dependencies to/from the “Test iteration 1” and “Test iteration 2” tasks in the macro-schedule are implemented as Touchpoints in the Development (micro-) schedule.
- The result is that the end date has moved out from November 2018 to August 2019 - a 9 month gap

In practice, we would also have had a Test team micro-schedule and a Manufacturing team micro-schedule. These schedules would have been selected and rolled-up along with the Development schedule.



	ⓘ	Rol Ind	Name	uratio
0			Macro Schedule - Develop Widget by October 2018	466.5 d
1			Widget Iteration 1	201 d
2	▦	↑	Develop iteration 1	180 d
3		↓	Test iteration 1	21 d
4			*Iteration 1 tested*	*0 d*
5			Widget Iteration 2	111.5 d
6		↑	Develop iteration 2	97.5 d
7		↓	Test iteration 2	14 d
8			*Iteration 2 tested*	*0 d*
9			Widget Iteration 3	124 d
10		↑	Develop iteration 3	117 d
11			Test iteration 3	7 d
12			*Iteration 3 tested*	*0 d*
13			Manufacture Widget	317.5 d
14			Put in place manuacturing tools	45 d
15			Manufacture 1000 Widgets	30 d
16	▦		*TARGET:1000 Widgets manufactured*	*0 d*
17			*1000 Widgets manufactured*	*0 d*

	ⓘ	Rol Ind	Name	uratio
0			Micro Schedule - Widget Development Team	429.5 d
1			Widget Iteration 1	0 d
2	▦	↓	*Test iteration 1*	*0 d*
3			Widget Iteration 2	0 d
4	▦	↓	*Test iteration 2*	*0 d*
5		↑	Develop iteration 1	180 d
6			Get requirements	30 d
7			Design	60 d
8			Build	60 d
9			Verify	60 d
10			*Iteration 1 developed*	*0 d*
11		↑	Develop iteration 2	97.5 d
12			Redesign	45 d
13			Build	45 d
14			Verify	30 d
15			*Iteration 2 developed*	*0 d*
16		↑	Develop iteration 3	117 d
17			Redesign	45 d
18			Build	42 d
19			Verify	30 d
20			*Iteration 3 developed*	*0 d*

Example of a refresh with Macro-micro Roll-up

A typical program refresh process using a Macro-micro Roll-up would be as follows:

1. Update the macro-schedule
    - Check “Skip-roll-up tasks (macro plan)”
2. Each team refreshes their micro-schedules
    - Update - check “Skip-roll-down tasks (micro plan)”
    - Pull-in
    - Run the Wigglechart
3. Macro-micro Roll-up
    - Rolls-down the latest dates from the macro-schedule
    - Rolls-up the latest dates from the micro-schedules
4. Pull-in the macro-schedule
    - Where a roll-up is on the critical path, select it and click on Goto Task In Linked Schedule to open the micro-schedule and go directly to the same task in the micro-schedule
    - Look for pull-in opportunities and the micro-level
5. If changes were made to cause a pull-in, re-do the Macro-micro Roll-up
6. Run the Wigglechart
    - At the macro-schedule
    - At the micro-schedules (if needed - this will overwrite the previous points from step 2)