processmodel™

SELF TEACHING GUIDE

Building a Software Development Model

A guided first path through ProcessModel for a delivery team: open a finished model, run the starter model and change five settings, then build one from an empty canvas.

Contents

A guide for someone opening ProcessModel for the first time.

Before you start

This guide assumes you have never opened ProcessModel and that nobody is sitting beside you. Every instruction names the menu, the button, and the field to use, and says what should happen next. Where a number is given, it is the number this build produces, so you can check your screen against the page.

You need ProcessModel installed and about two hours. Work through it in order. Parts 1 and 2 explain what you are looking at, Part 3 opens a finished model, Part 4 runs a small one and changes it five times, and Part 5 builds a model from an empty canvas.

What is in this guide

Part 1. The window, and the few habits that make the rest easy.

Part 2. What a lens is, and how the **Software Development** lens changes it.

Part 3. Open a finished model and read its answer.

Part 4. The starter model: run it, then change five things and watch each one move a number.

Part 5. Build a model from an empty canvas: five developers, two testers, twenty stories.

Part 6. Reference: the words, the numbers, the mistakes, and where everything lives.

> ⓘ Two conventions. Words in bold are exactly what is written on screen. A path like **File ▸ Demo Models** means open the **File** menu and click **Demo Models**.

Part 1. The window

1.1 What ProcessModel is for

ProcessModel runs a process forward in time and tells you what happened. You draw the steps, say who does them and how long they take, and press a button. The product moves imaginary work through the drawing, minute by minute, and writes a report: how long things took, what waited, how busy people were, what it cost.

Two things follow from that, and they catch out everyone who arrives from a spreadsheet.

The times you type are spreads, not single numbers. Real work takes eight hours sometimes and twenty hours other times, so you type a range, and the product draws a different value each time. That is the point: the answer comes back as a range too.

One run is one possible future. Because of those spreads, running twice gives two answers. Asking for thirty runs and reading the spread is how you get a number you can promise.

Nothing you do here changes the real system, and nothing is sent anywhere. A model is one file on your computer.

1.2 Opening the product

Start ProcessModel. The **Welcome screen** opens.

Down the left are **Home**, **New Model**, and **Open from Computer**, with **Settings**, **What's New**, and **Help & Tutorials** at the bottom.

The middle of the **Home** page has **Start New Model** with a row of tiles (the first is **Blank Canvas**), then **Recent Projects**.

Do not click anything yet. Part 3 opens a finished model first, which is the fastest way to see what the product is for.

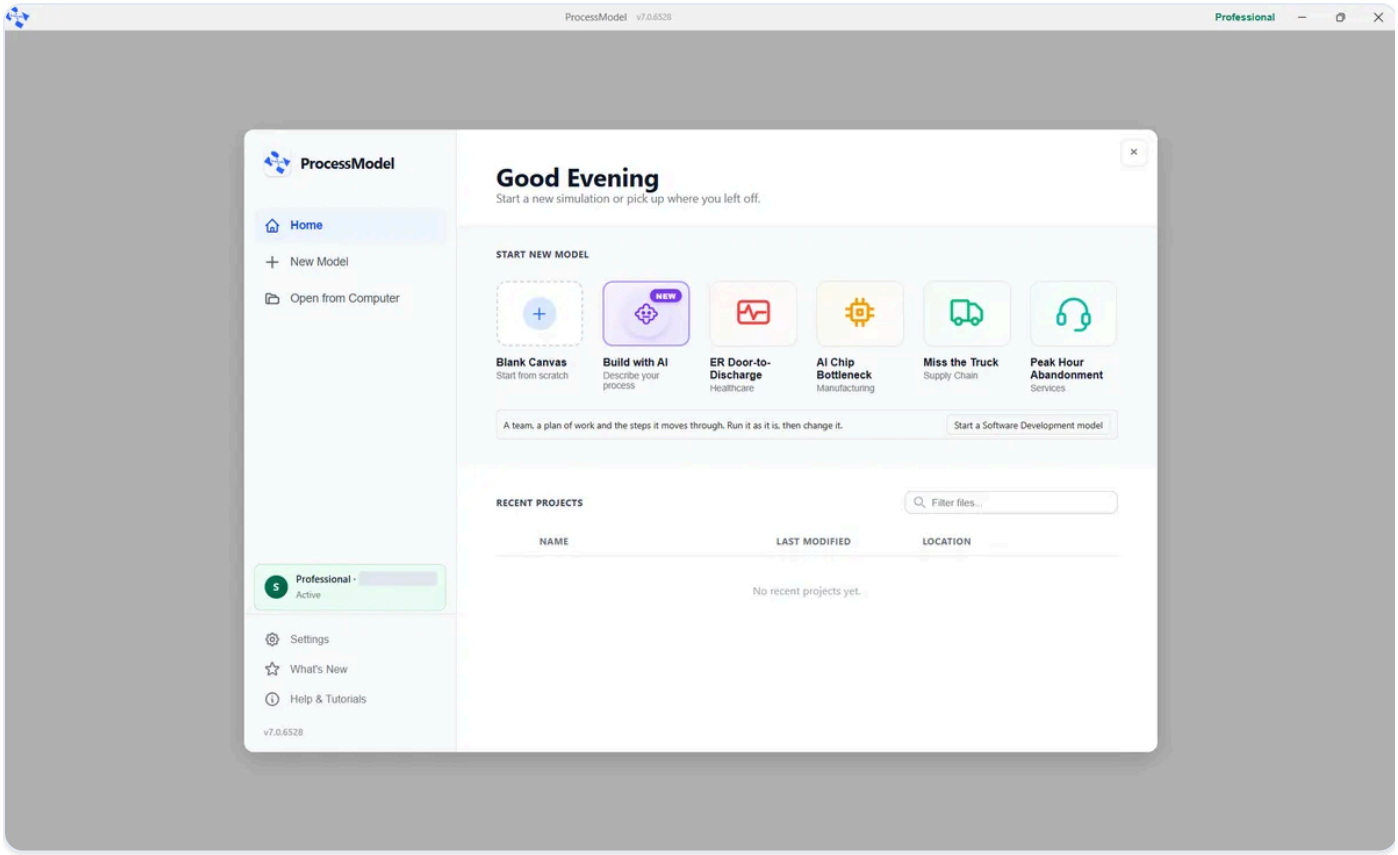

Figure 1. The Welcome screen: the left sidebar, the Start New Model tiles, and Recent Projects.

1.3 The main window

Once a model is open, you are looking at six things. Find each one before going further.

What	Where	What it does
The menus	Along the top: File, PM Soterix, Edit, View, Arrange, Simulate, Tools, Help	Everything in this guide is reached from File, View, Simulate or Tools.
The toolbar	Under the menus	Icon buttons. The two that matter are the play button, named Run Simulation, and the bar-chart button, named Output Report. Hover over a button to see its name.
The canvas	The large middle area	Where the model is drawn. Scroll to pan, and use View ▸ Fit to View if you lose the drawing.
The element toolbar	Down the left of the canvas	The general elements every model is made of. It never changes, whatever kind of model you are building.
The properties panel	Appears on the right when you click something	Where you change whatever you clicked. Click empty canvas and it goes away.

What	Where	What it does
The industry palette	Appears on the right when you ask for it	A second, optional panel holding one industry's elements. This is where the Software Development elements live. Part 2 opens it.

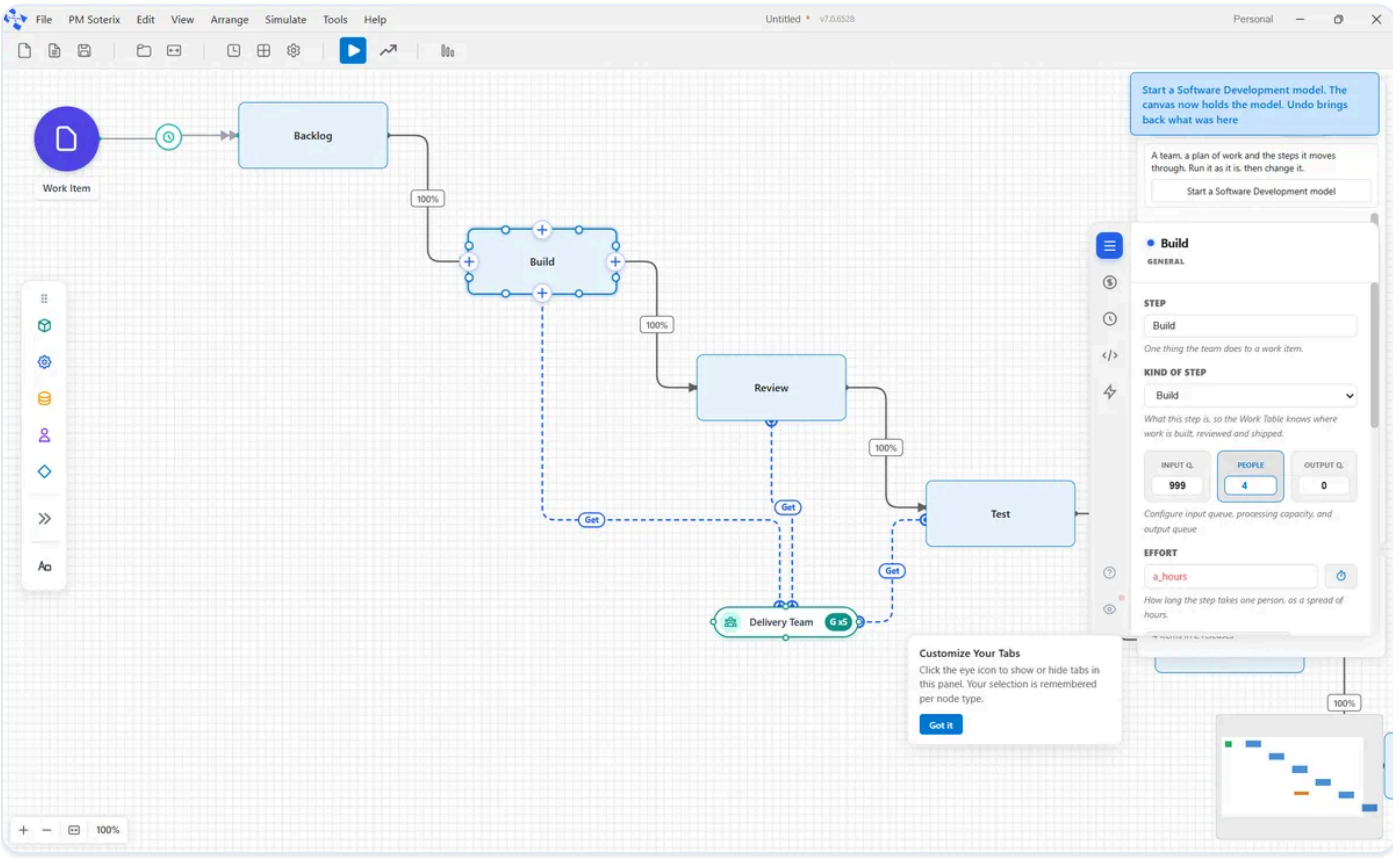

Figure 2. The main window with a model open: menus, toolbar, canvas, the element toolbar on the left, and the properties panel on the right.

1.4 Four habits

Click a thing to edit it. There is no **Edit** command and no double-click needed. Click an element on the canvas and the properties panel appears on the right. Click an arrow, and you get the arrow's properties. Click the empty canvas to hide the panel.

The tabs in that panel are icons. They have no words on them. Hover over an icon to see its name. The first is always **General**.

Change one thing at a time. That is the whole method of this guide. Change one field, run, read one number, write it down. If you make two changes at once, you cannot tell which one moved the answer.

Save early. **File** ▸ **Save As** (or Ctrl+Shift+S) asks for a **MODEL NAME** and a folder and writes one file ending in .pmd. After that, Ctrl+S saves.

> If the screen does not match this guide. Check the build number. The version sits beside the model name at the top of the window and should read v7.0.6197 or later, and **What's New** on the **Welcome screen** should open at Version 7.0.6182 or later. An older build does not have the **Software Development** elements at all.

Part 2. What a lens is

2.1 The problem a lens solves

ProcessModel is a general tool. The same six or seven elements model a hospital emergency department, a milk plant, a warehouse, and a software team because, underneath, they share the same few ideas: work arrives, it queues, somebody or something works on it, and it moves on.

That generality is the strength and the obstacle. A delivery team that opens the product sees **Entity**, **Activity**, **Queue**, and **Resource**, plus a report on throughput time and utilization. None of those are words the team uses. Meanwhile half the toolbar is for things they will never model: tanks, batches, arrivals, and bunches of routings.

2.2 What a lens is

A lens is a setting on one model that makes the product speak one industry's language. It is stored inside the model file, so a model either wears a lens or it doesn't, and two models opened side by side can differ. Turning it on changes four things.

What the lens changes	What you get with the Software Development lens
The elements offered	A Software page in the industry palette holds the ten elements a delivery team needs, each with a sentence explaining what it is and a button that builds a working model in one click.
The words on the fields	A Resource is called a Team, its Quantity is called People, an Activity's Processing Time is called Effort, and so on. The full list is in Part 6.
The report	Five tabs in the team's language: Forecast, Flow, Team, Quality and Cost. The general report tabs are hidden.
The run settings	Replications are called Runs; the run length is also shown in weeks, and two settings appear that only a plan of work needs: Plan date and Working day.

What a lens does not change is the arithmetic. A **Team** is still a resource, and a **Step** is still an activity. The same model with the lens taken off gives the same answer to the last decimal place. The lens changes what you read, never what is computed.

The **Software Development** lens also brings one thing that is genuinely new rather than renamed: the **Work Table**, a plan of work written down before the run as a list of items with sizes, dependencies, and releases. Part 4 uses it.

2.3 Opening the Software page

Open the **View** menu and click **Show Industry Palettes**. A panel appears on the right with a row of tabs across the top, one per industry.

Click the **Software** tab. The heading below the row reads **Software**.

Under the heading is a button reading **Start a Software Development model**. Below are the ten elements, each with its name and a sentence. At the very foot of the panel is one quiet row about the lens itself.

View ▸ Show Industry Palettes is a toggle, and it remembers. If the panel is already on screen, clicking the menu item closes it.

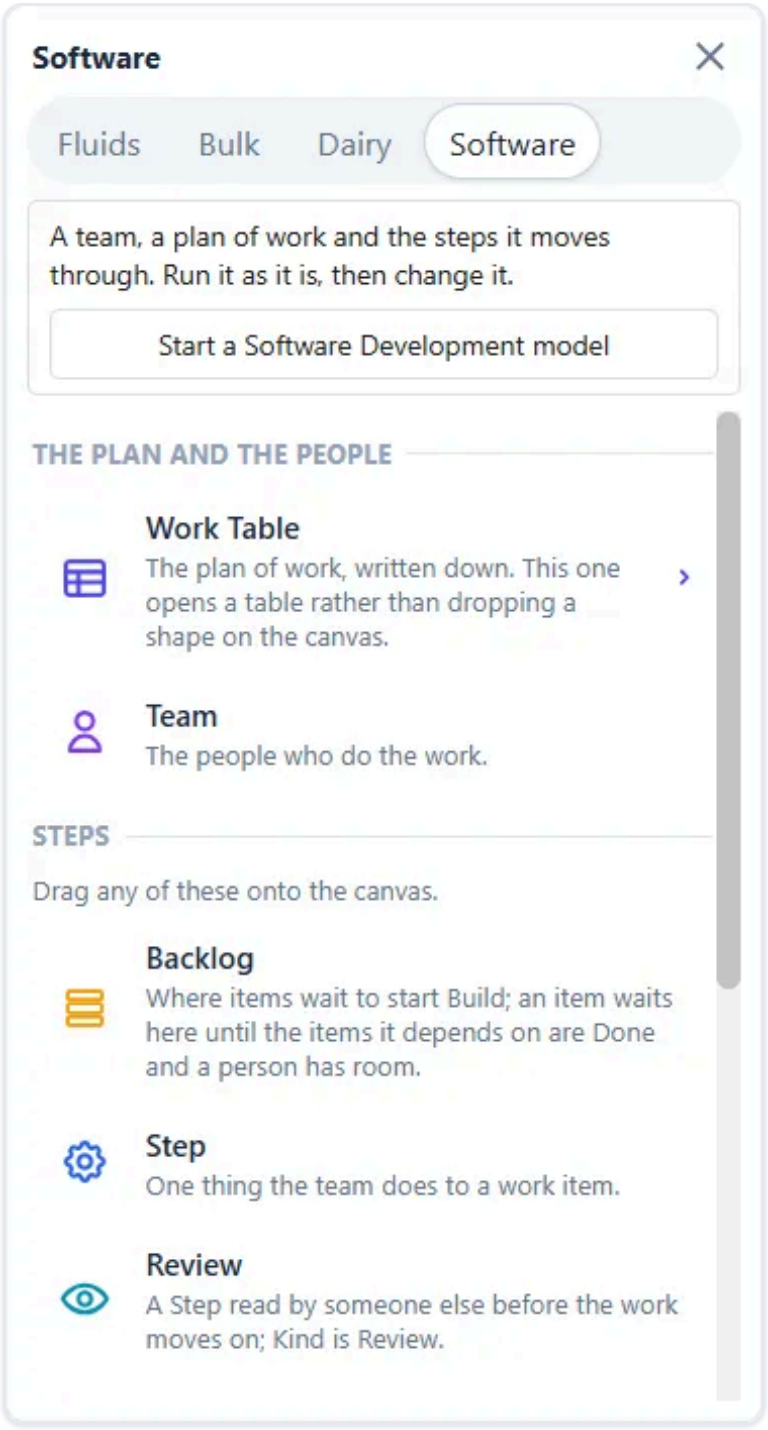

Figure 3. The industry palette with the Software tab selected: the Start a Software Development model button, the ten elements with their sentences, and the lens row at the foot.

2.4 The ten elements

These are the only elements a software model needs. Drag nine of them onto the canvas. The first one opens a window instead.

Element	What it is
Work Table	The plan of work, written down. This one opens a table rather than dropping a shape on the canvas.
Team	The people who do the work.
Backlog	Where items wait to start Build; an item waits here until the items it depends on are Done and a person has room.
Step	One thing the team does to a work item.
Review	A Step read by someone else before the work moves on; Kind is Review.
Test	A Step done by testers; Kind is Test.
Pipeline	The automated run the build system does; Kind is Pipeline.
Release	Where a release ships; a release waits here until every item in it is done.
Freeze	Shuts the Release step for a while, so nothing ships in that window.
Done	Where a work item finishes; Kind is Other.

> The left toolbar does not change. Turning the lens on adds the **Software** page on the right. The general elements down the left stay exactly as they are, and you will need one of them in Part 5, because the ten above do not include the work item itself.

2.5 Taking the lens off, and putting it back

At the foot of the **Software** page is one row about the lens. What it says depends on the model that is open.

A model that already wears the lens offers **Remove lens**. The elements stay; the words return to the general ones, and the report returns to its general tabs.

A model with elements on the canvas and no lens offers **Use the lens**.

An empty canvas shows no row at all, because there is nothing to put a lens on.

Either button changes the model, and one press of Ctrl+Z undoes it. Nothing is lost either way, because the lens only decides what you read.

Part 3. Open a finished model

The fastest way to understand what a software model answers is to read one that is already built. Two ship with the product.

3.1 Finding and opening it

Open the **File** menu and click **Demo Models**. A window opens titled **Demo Models**, with the line Ready-to-run examples by industry beneath it.

Down the left is a list headed Industry: All, Healthcare, **Manufacturing**, Supply Chain, Services, **Software Development**, Defense, Government. Click **Software Development**.

Two cards appear: Three More People and Ten More People. Each card has two buttons: **Open** and **Brief**.

On Three More People, click **Brief** first. A panel slides in with the case study: the situation, the four choices, the numbers, and what they mean. Read it. When you have finished, either close the panel or press **Open this model** at its foot.

The model opens on the canvas and the gallery closes.

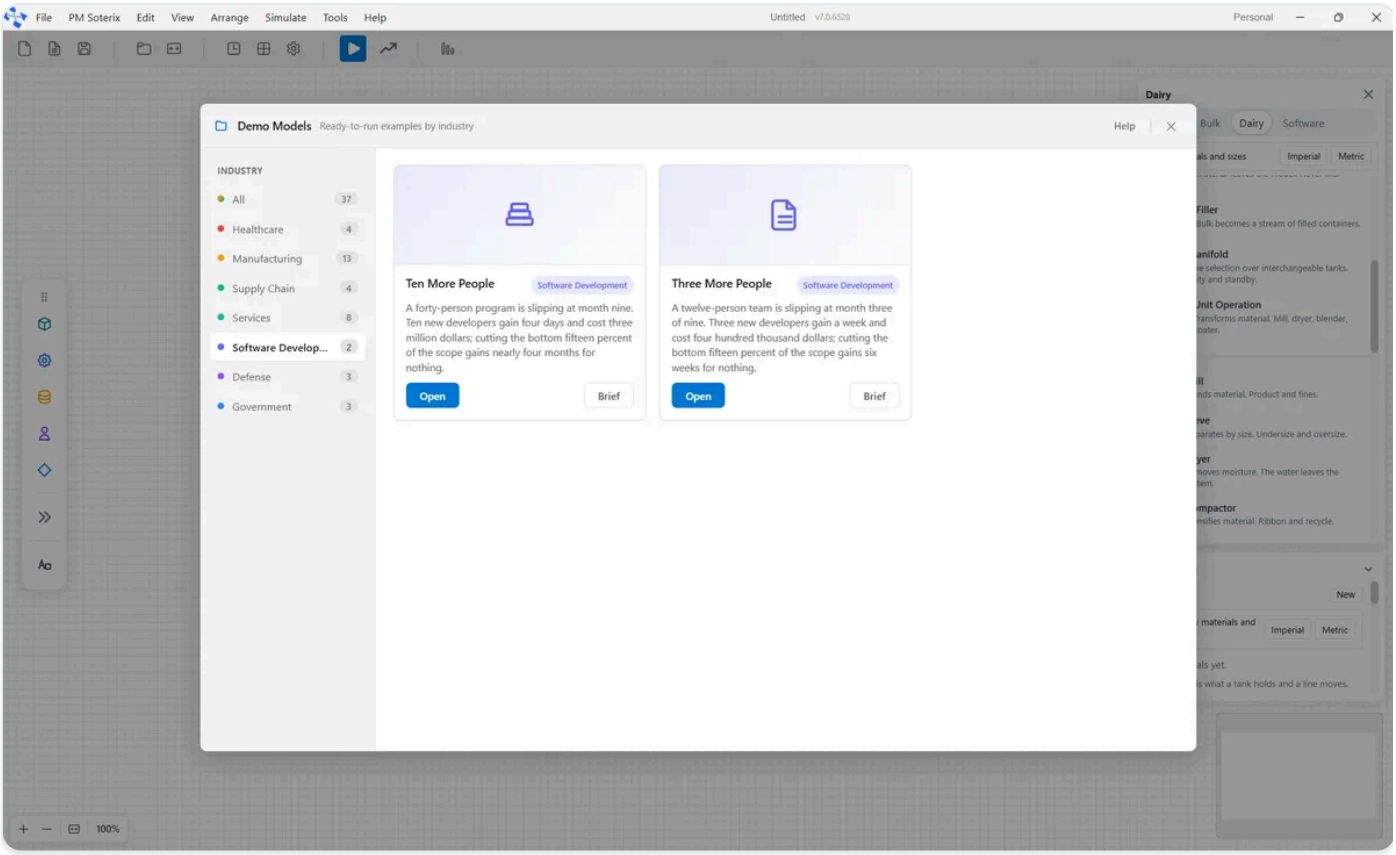

Figure 4. The Demo Models window with Software Development selected in the Industry list and the two cards showing.

> Open Three More People, not Ten More People, for your first look. Both are real programs at full size. Three More People is a twelve-person team over nine months and takes about two and a half minutes to run at its shipped settings. Ten More People is a forty-person, two-year program and takes about an hour and three quarters. Read its Brief by all means, but do not press Run on it while you are learning.

3.2 What the model says

You do not have to run Three More People to learn from it. Its Brief carries the answer, and the model on the canvas shows you what a finished software model looks like: a **Backlog**, **Build**, **Review**, **Test**, and **Release** steps, one **Delivery Team** underneath, and a plan of work with two hundred and twenty items.

The question it answers is the one every late project asks. A twelve-person team is nine months into a nine-month plan and is slipping at month three. There are four choices: do nothing, add three developers now, wish they had been added at month one, or cut the bottom fifteen percent of the scope. The plan says day 273.

Choice	Finish, most runs by	Against the plan
Do nothing	day 347	74 days late
Add three developers at month three	day 339	66 days late
Add three developers at month one	day 325	52 days late
Cut fifteen percent of the scope	day 301	28 days late

Three developers are hired at month three buy a week and cost about four hundred thousand dollars, because a new person starts at half speed, takes eighty working days to reach full speed, and takes a fifth of an experienced developer while learning. The same three hired at month one buy three weeks. Cutting the scope buys six and a half weeks and costs nothing.

That is what this kind of model is for: not a prettier plan, but a price for each choice in the room.

3.3 If you do run it

Press the play button on the toolbar, named **Run Simulation**. Expect about two and a half minutes for this model. A dialog reads **Simulation Complete** and asks Would you like to view the results? Click **Yes**, and the report opens. Part 4 explains every tab.

Part 4. The starter model and five changes

This part is the heart of the guide. You will build nothing: the product ships a small, complete software model, and you will run it and then change five single fields, watching each one move a number in the report. By the end, you will know what every tab is for.

4.1 Start it

If a model is open, save it or close it. **File** ▸ **New** gives you an empty canvas.

Open **View** ▸ **Show Industry Palettes** and click the **Software** tab, as in Part 2.

Press **Start a Software Development model**. On an empty canvas, it builds at once and asks nothing. (If the canvas is not empty, it asks first, because it replaces what is there. One press of Ctrl+Z puts your old model back.)

Six steps appear, stepping down the canvas from top left to bottom right, with a **Delivery Team** underneath and dashed lines rising from it to three of the steps.

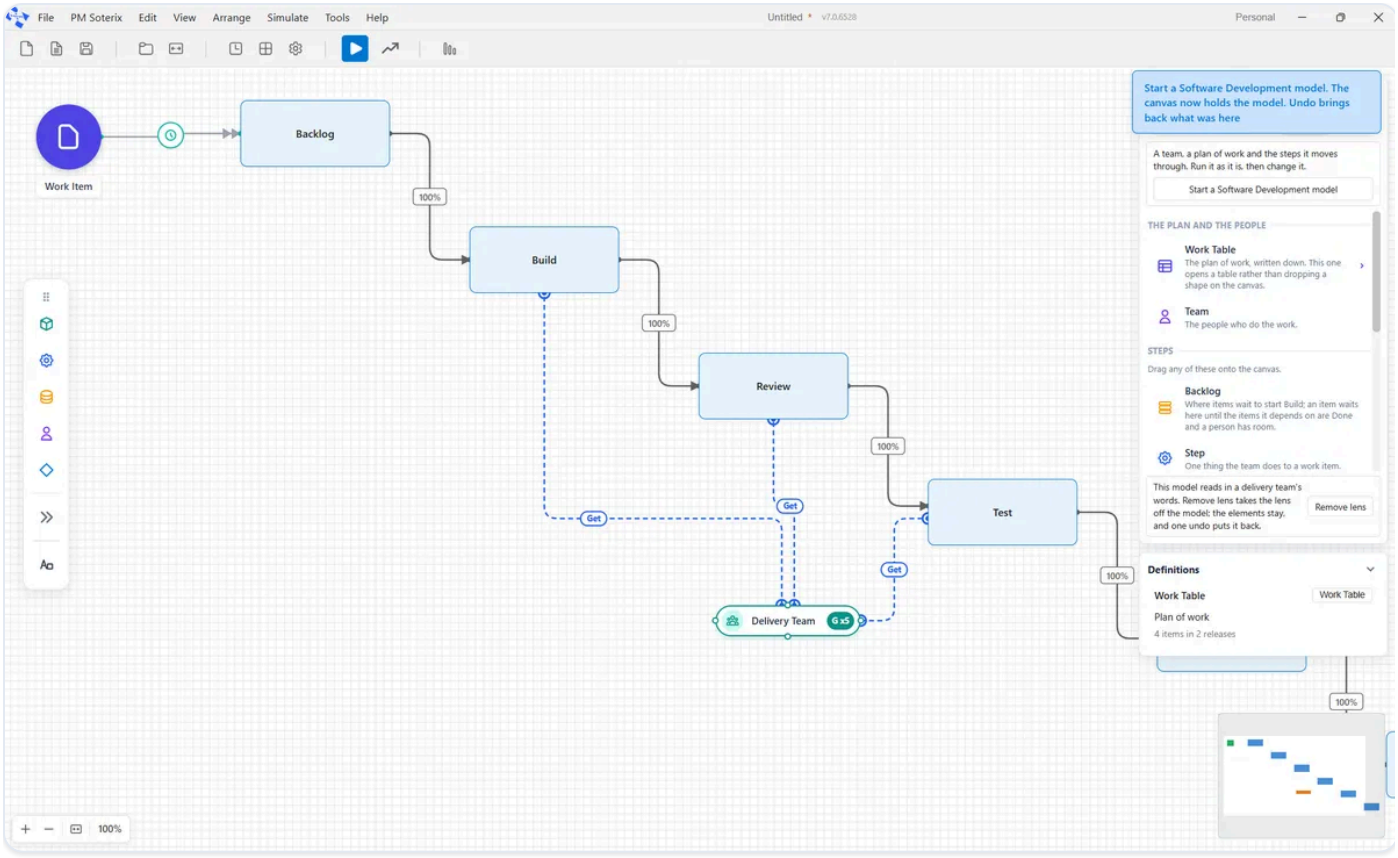

Figure 5. The starter model on the canvas: Work Item, Backlog, Build, Review, Test, Release and Done as a staircase, with the Delivery Team below.

4.2 What you are looking at

Click each element once and read its properties on the right. This is the whole model.

On the canvas	What it is	What it is set to
Work Item	The kind of work. Every row of the plan becomes one of these when the run starts.	Nothing to set.
Backlog	Where an item waits for what it depends on, and for a person with room.	Kind of step: Backlog. No effort.
Build	Where the item’s hours are spent.	Kind of step: Build. Effort is the item’s own size. Four people can be on it at once.
Review	Someone reads the work before it moves on.	Kind of step: Review. Effort: 1 hour. Two at once.
Test	A tester checks it.	Kind of step: Test. Effort: 2 hours. One at a time.
Release	A release ships when every item in it is done.	Kind of step: Release. No effort.
Done	Where an item finishes and is counted.	Kind of step: Other.

On the canvas	What it is	What it is set to
Delivery Team	The people. Four developers and one tester.	Works a weekday office day. A fifteen-minute stand-up every day takes the whole team.

The arrow from **Work Item** into **Backlog** is the arrival: it is the one place the plan of work is read. Click it and the properties panel shows **Work Table** as its kind.

The plan of work

Click the **Work Table** tile at the top of the **Software** page. A window opens, titled **Work Table**. It holds four rows and five columns.

Item	Type	Size (hours)	Depends on	Release
Sign in	Work Item	T(6,8,12)		R1
Sign out	Work Item	T(3,4,6)	Sign in	R1
Password reset	Work Item	T(8,12,20)	Sign in	R2
Sign-in error message	Work Item	T(1,2,4)		R2

T(6,8,12) means: at least six hours, most likely eight, at most twelve. Each run draws a different number from that shape. That single convention is why the answer is a range. Row order is priority: the team works down the list.

The window has no OK and no Cancel. Every keystroke is already in the model. If you make changes Ctrl+Z undoes it. Close it with the ✕ or the Escape key.

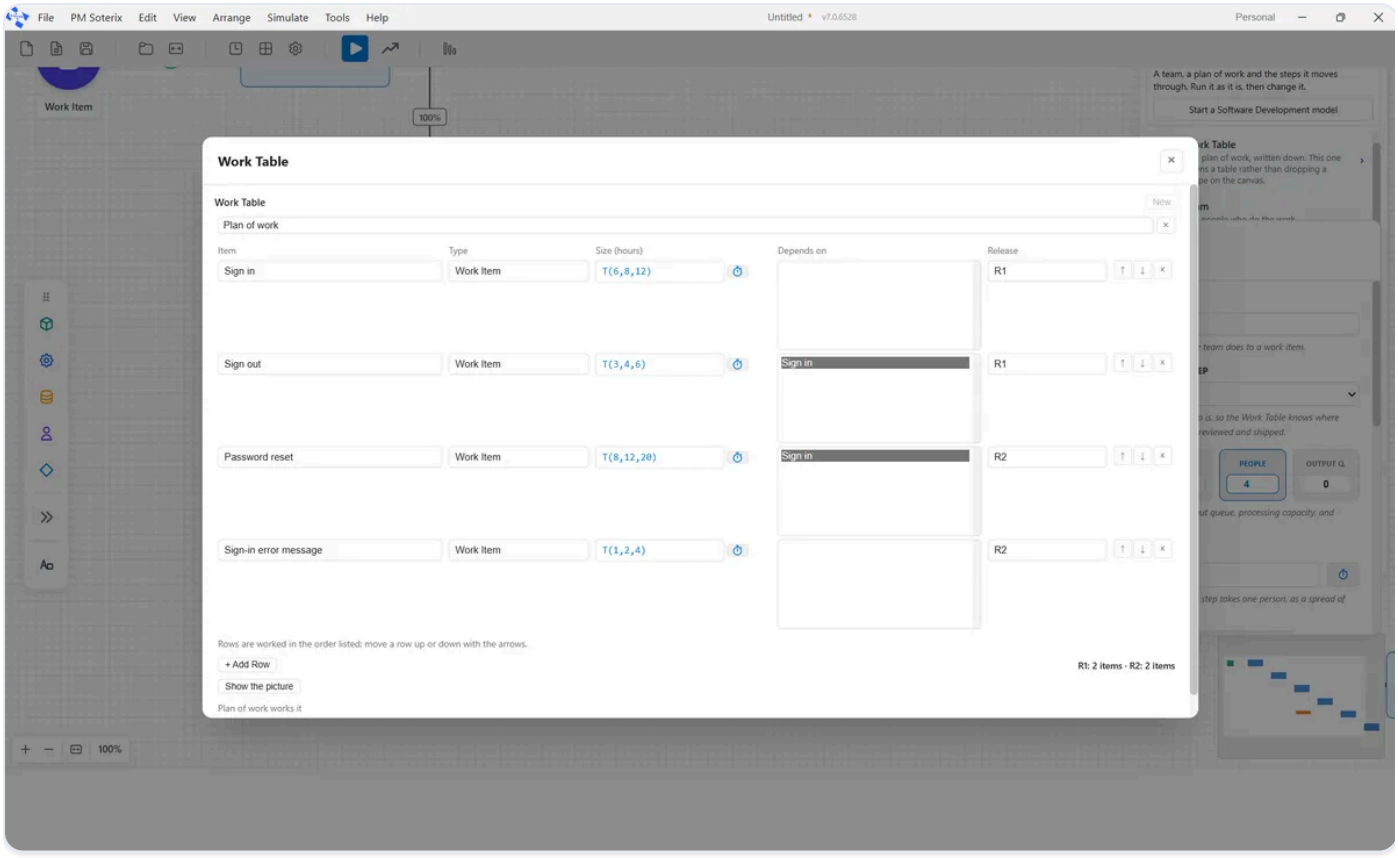

Figure 6. The Work Table window with the four starter rows.

4.3 Run it

> It arrives ready to run. This model asks for five runs, a four-hundred-day window, and no animation, so pressing Run answers in about a second. Animation is worth turning on once, later, to watch items move: it plays a run at the pace of the clock, so a four-hundred-day window would take a working day to watch. The switch is in **Simulate** ▸ **Options** under Animation.

Press the play button on the toolbar, named **Run Simulation** (or **Simulate** ▸ **Run Simulation**, or Ctrl+R). It takes about a second.

A dialog headed **Simulation Complete** names the day the plan of work finished, and asks Would you like to view the results? Click **Yes**.

The **Output Report** window opens. Across the top are five questions, numbered 1 to 5, and it opens on the second: How is it doing? Underneath that row are the lens's own tabs: **Forecast**, **Flow**, **Team** and **Quality**. A fifth, **Cost**, joins them once the team carries a pay rate, which this model does not. Those tabs, and no general ones, are what the lens shows.

You can reopen this report at any time from **Tools** ▸ **Output Report** (Ctrl+Shift+R) or the bar-chart button on the toolbar. Escape closes it.

Read the Forecast tab

It opens on **Forecast**. At the top it says Finish, across 5 runs and 5 of 5 runs finished the program. Below are three blocks.

The whole program, with three readings: Half the runs by, Most runs by (P85) and Nearly every run by. All three read day 3.

Each release. R1 reads day 2 and R2 reads day 3.

Where the runs finished, there was a small chart of three bands. Every band is labelled day 3.

Two things to take from that. The tab prints whole days, so day 3 means partway through day three. And the three readings are the same because five runs of this four-row plan all finish within a quarter of a day of each other: the spread is real, and smaller than the page can print. Change 1 is what separates them.

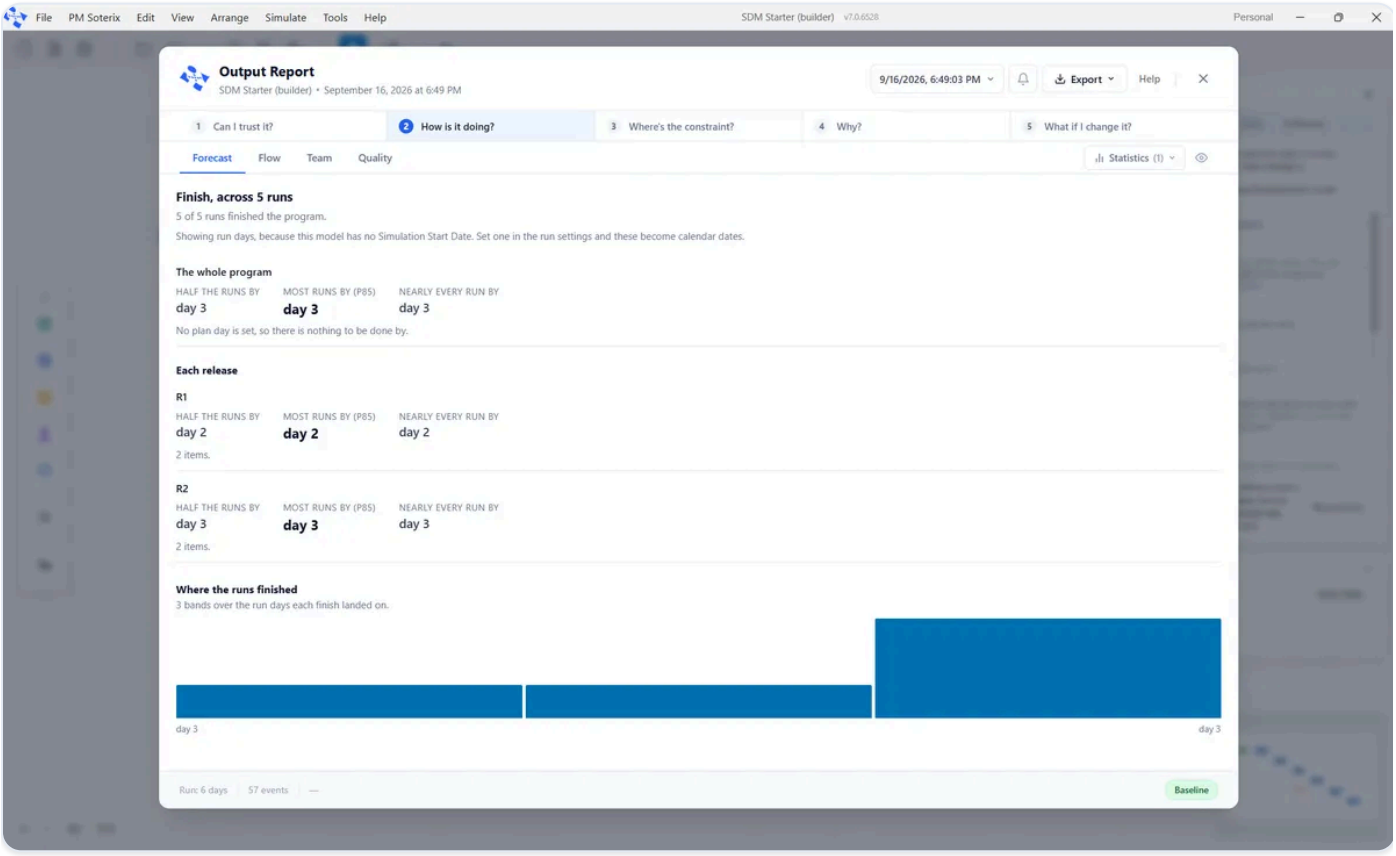

Figure 7. The Forecast tab on the first run: three readings for the whole program and for each release, and the chart of where the runs finished.

Read the Flow tab

Click **Flow**. The top is a table with one row per work item type. Below it are five sections that appear only when the model has a plan of work. The two to look at now are the last two.

Flow efficiency lists each item with its Hours of work, the Days, **Build** to **Done** it took, and an Efficiency percentage: how much of the time between starting and finishing anyone was actually working on it.

Blocked, per item, lists what waited and for how long. Sign out and Password reset each show a large Blocked (min) figure, because both wait for Sign in. The other two items show nothing.

That is the **Backlog** doing its job: an item that depends on another does not start until that other one is **Done**.

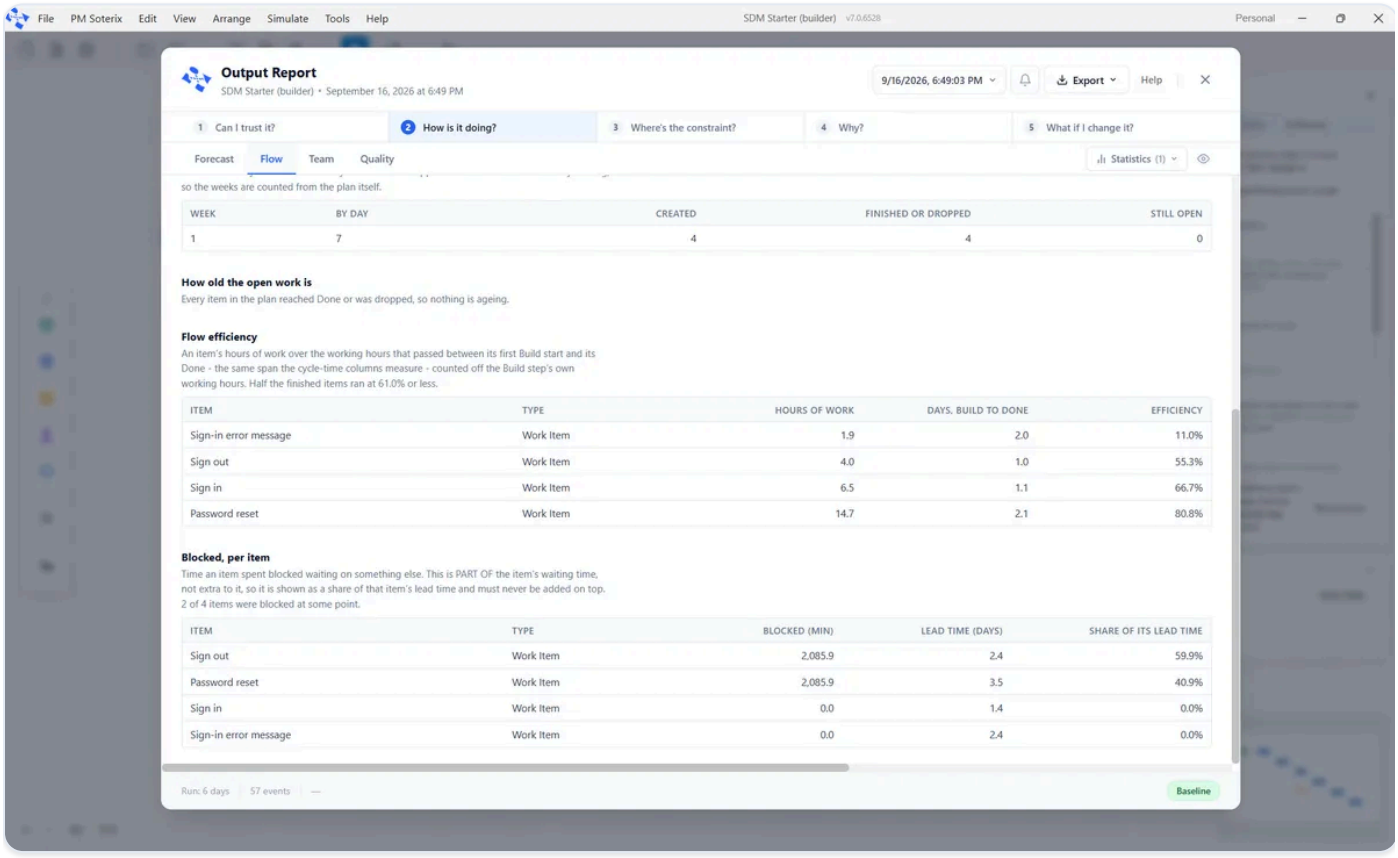

Figure 8. The Flow tab, scrolled to the Blocked, per item section.

4.4 The five changes

Each change below is one field. After each one, run again and check the named place. Keep the change unless the step says to put it back.

Change 1. Ask for thirty runs instead of one

Open **Simulate** ▸ **Options** (Ctrl+Shift+O). A dialog opens titled **Simulation Options**.

The first section is Time & Duration. The first field is **Run length**, set to 400, with Min / Hr / Day beside it and Day chosen. Leave it. The plan finishes in days; the window is wide so that a bigger plan typed into this model still finishes inside it.

Below it is **Runs**, set to 5. Change it to 30.

Press **Apply** at the bottom right. (There is no OK. Apply closes the dialog and keeps the change; Cancel throws it away.)

Press **Run Simulation** again, then Yes.

Where to see it. The **Forecast** tab. Its heading now reads Finish, across 30 runs, and the three readings for the whole program have come apart: day 3, day 4, day 4. The chart headed Where the runs finished grows a tail on the right.

What it teaches. Five runs said the plan takes about three days. Thirty runs say it sometimes takes four, and four is the number a team must promise. A handful of runs find the answer; more runs find the bad days. The middle reading is the one to give anybody who asks. From here on, leave **Runs** at 30.

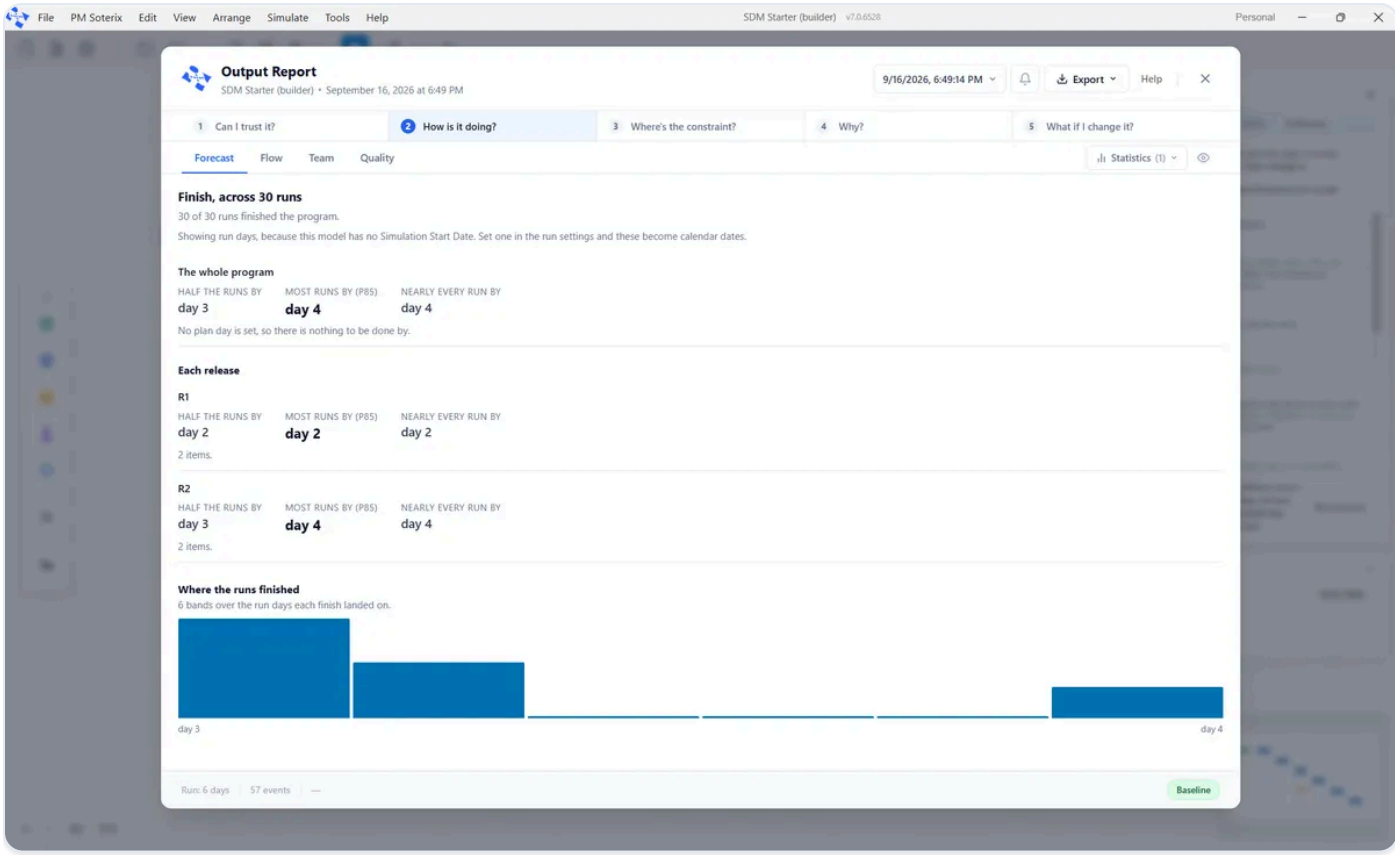

Figure 9. The Forecast tab across thirty runs: three readings per row and the Where the runs finished chart.

Change 2. Make one estimate more certain

The plan says Password reset is between 8 and 20 hours, most likely 12. Suppose the team investigates and is now sure it is about 12.

Open the **Work Table** window (the tile at the top of the **Software** page).

In the Password reset row, click the **Size (hours)** cell and replace T(8,12,20) with T(11,12,13).

Close the window with Escape and run again.

Where to see it. The **Forecast** tab, the whole program row. Most runs by (P85) moves from day 4 to day 3, and all three readings are now day 3.

What it teaches. The most likely size did not change. What changed is how badly the estimate could be wrong, and that alone moved the date a team could promise. The spread in your estimates, not the speed of your people, is what pushes out the date you can commit to.

Put it back to T(8,12,20) before the next change.

Change 3. Remove a dependency

Open the **Work Table** window again.

In the Password reset row, clear the **Depends on** cell so that it no longer names Sign in.

Close the window and run again.

Where to see it. Two places. On **Forecast**, the whole program falls to day 2 on all three readings. On **Flow**, in the Blocked, per item section, the Password reset row's Blocked (min) drops to nothing, while Sign out still waits.

What it teaches. On a small plan, the dependencies, not the people, set the finish. This one item waited about a day and a half for another item to finish, and simply not needing it anymore was worth two days of the four. Before hiring, look at the Blocked column.

Put the dependency back (choose Sign in in the **Depends on** cell) before the next change.

Change 4. Make the stand-up longer

This one needs a tab that is hidden until you ask for it, which is worth learning once.

Click the **Delivery Team** on the canvas. The properties panel appears on the right, with a column of tab icons.

Click the eye icon at the bottom of that column (its name is **Manage visible tabs**). A list of tabs appears with tick boxes. Tick **People**. It is now a tab in the row, and it stays for every team from now on.

Open the **People** tab. Below the **People** table is a table headed **Interruptions**. Its first row is greyed and named **Interruptions**; that is the team's own availability, and it carries a note pointing at the **Availability** tab, so leave it. The row beneath it is the **Stand-up**. Its **Time Between Interruptions** is 1440 and its **Interruption Time** is 15, both in minutes, with a Min button beside each (1440 minutes is one day). Change the **Stand-up**'s **Interruption Time** to 90.

Click the empty canvas to put the panel away, and run again.

Where to see it. The **Forecast** tab, the whole program row. Half the runs by slips from day 3 to day 4; Most runs by (P85) and Nearly every run by hold at day 4.

What it teaches. Meetings are hours of work like any other. A daily stand-up that grows from fifteen minutes to an hour and a half takes a real bite out of every working day, and it pushes the day the typical run finishes out by one. Push it much further and the team can no longer clear the plan inside the run window at all, and the **Forecast** stops giving a date and reads not reached. The cost of a meeting is the work that did not get done.

Put the **Stand-up**'s **Interruption Time** back to 15 before the last change.

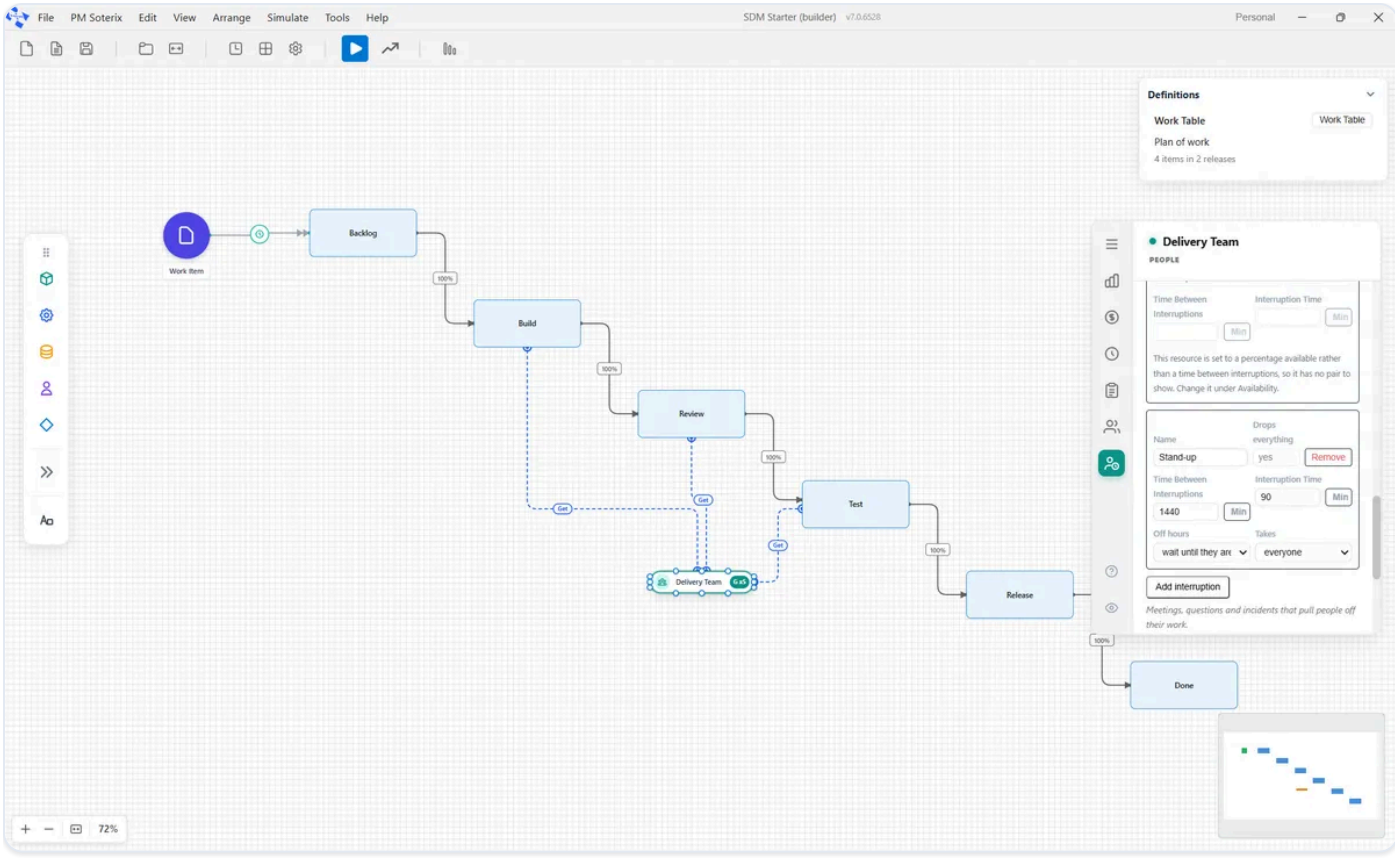

Figure 10. The Delivery Team properties, People tab, with the Interruptions table and the Stand-up row.

Change 5. Change the size of the team twice

This one has three runs, and the answer is the most useful thing in this guide.

Click the **Delivery Team** and open the **People** tab again. The **People** table at the top of it has one row for the team itself, with **How many** set to 4, and a second row for the Tester.

Set the first row's **How many** to 8. Run, and read the **Forecast**.

Now set it to 1. Run, and read the **Forecast** again.

Where to see it. The **Forecast** tab: the whole program row reads day 3, day 4, day 4 in all three cases. Doubling the team and cutting it to a quarter changed nothing.

Why did nothing move? The plan has four items, and two of them cannot start until Sign in is finished, so there is never work for more than two people at once. One developer and the tester were always enough. The team was never the constraint, and no amount of hiring would have changed the answer.

Now find the number that does bite. Put **How many** back to 4. Click the **Build** step and look at its **General** tab: there is a row of three small boxes, and the middle one is labeled **People** (hover over it and it says **People** who can work this step at once). It is set to 4. Change it to 1 and run.

Where to see it. The **Forecast** tab. Half the runs by slips from day 3 to day 4.

What it teaches. There are two different numbers called **People**, and they mean different things. On a **Team**, it is how many people exist. On a **Step**, it is how many of them may be on that step at once. Adding people to a plan that cannot use them is the most common mistake in resourcing a project, and two runs of a four-row model show it.

Put the **Build** step's **People** box back to 4.

4.5 The other three tabs

You have used **Forecast**, **Flow** and **Team**. The other two are worth one look each.

Quality is headed Checking, rework and what never shipped. Three cards count the items that went through a **Review**, **Test** or **Pipeline** step, the ones that came back for rework, and the ones that never shipped. Under them, By the kind of step groups the work by what kind of step did it. The starter model has no rework route, so the middle card reads that there is none. That is a true answer, not a blank.

Cost is the general cost report under a new name. It is not even on the tab row until a team carries an hourly rate, which the starter model does not set. Part 5 sets one, and the tab appears.

4.6 Save it

File ▸ **Save As**, type a **MODEL NAME**, choose a folder, and press **Save Model**. The file ends in .pmd and holds the model, the plan of work, the lens, and the run settings. Opening it later gives you exactly this.

Part 5. Build a model from an empty canvas

This part builds a model of a real team: five developers, two testers, and twenty stories in two releases. Nothing is inherited from the starter model. Follow it in order, because a few steps depend on the one before.

Allow an hour. Save as you go.

> ⓘ What you are about to build. A work item called Story arrives from a plan of twenty rows, waits in a **Backlog** until what it depends on is done, is built by a developer, reviewed by a developer, tested by a tester, and ships with its release. Two teams: Developers and Testers.

5.1 Start with an empty canvas and the Software page

Open **File** ▸ **New** (Ctrl+N). An empty canvas.

Open **View** ▸ **Show Industry Palettes** and click the **Software** tab.

Do not press **Start a Software Development model** this time; that builds the model for you, which is Part 4. Note that the lens row at the foot of the page is absent: an empty canvas has nothing to put a lens on. It will appear once there are elements.

The page has two groups: THE PLAN AND THE PEOPLE at the top, and STEPS below it, under the line Drag any of these onto the canvas.

5.2 Put the steps on the canvas

Drag these six tiles from the **Software** page onto the canvas, left to right with a good gap between them. A dragged tile lands where you drop it.

Drag this tile	Drop it	What it is born with
Backlog	first, on the left	Kind of step already set to Backlog
Step	second	Nothing set. You will make this one Build in a moment.
Review	third	Kind of step already set to Review
Test	fourth	Kind of step already set to Test
Release	fifth	Kind of step already set to Release
Done	last, on the right	Kind of step already set to Other

Rename the second one. Double-click its name on the canvas (or select it and press F2), type **Build**, and press Enter.

With **Build** still selected, look at the properties panel on the right, **General** tab. Find **Kind of step** and choose **Build** from the list. This is how the plan of work knows which step spends an item's hours.

If you want them tidied up, **Arrange** ▸ **Auto-Layout** lines everything up once the arrows are drawn.

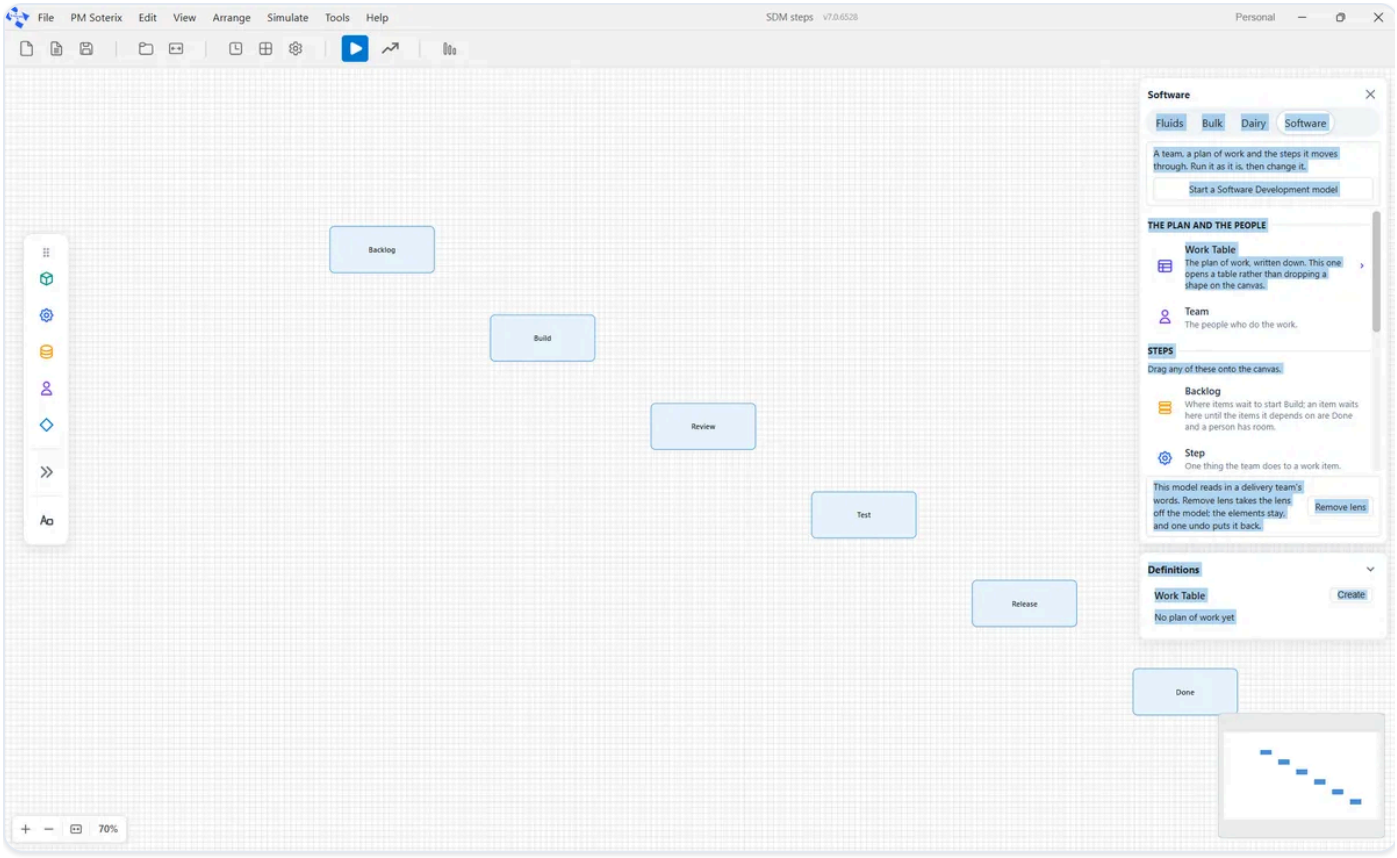

Figure 11. The six steps on the canvas after they have been dragged from the Software page.

5.3 Add the work item

The **Software** page has ten elements, and none of them is the work item itself. That is deliberate: a work item is a general element, and general elements live on the toolbar down the left of the canvas, which the lens never changes.

From the left toolbar, drag the tile named **Entity** onto the canvas, above and left of the **Backlog**. (The same thing is on the canvas right-click menu as Add **Entity**.)

Rename it Story (double-click the name, or use the Work item field on its **General** tab, which is what the lens calls the name field on this element).

> ⓘ The name matters. Every row of the plan of work names its type, and that name has to match this element letter for letter. If the element is called Story, every row's Type must read Story.

5.4 Add the two teams

Drag the **Team** tile from the **Software** page onto the canvas, below the steps. Rename it Developers.

With it selected, on the **General** tab set **People** to 5.

Open the **Shift** tab. **Working hours** reads Always Available (24/7) because a new model has no working hours yet, and a team with none works around the clock. No software team does, so make some.

Click + **New Schedule**. An editor opens. Type a **Schedule Name** such as Office hours. Under Quick Presets click **Office 9-5**: the weekly grid fills in with weekdays nine to five. Press **Create Schedule**.

Back on the **Shift** tab, check that **Downtime Rule** reads Preempt & Wait. A **Team** dragged from the **Software** page is born with it. It means a person puts the work down at the end of the day and picks it up the next morning; the other setting keeps them working past five until the item is finished, and books hours nobody worked.

Drag a second **Team** onto the canvas. Rename it Testers and set **People** to 2. On its **Shift** tab choose the Office hours schedule you just made; it is in the list now.

Leave **Speed**, **Starts on day**, **Learning curve**, and the rest of the **People** tab empty. They are for modeling people who join late or who are still learning, and they belong to a later question, not a first model.

5.5 Write the plan of work

The plan comes before the arrow that reads it, because the arrow needs something to point at.

Click the **Work Table** tile at the top of the **Software** page. It does not drag; it opens a window titled **Work Table**.

If the window is empty, press **New**. Name the plan in the box at the top, such as Q4 plan.

Press + **Add Row** and fill the five columns. Then repeat. Twenty rows is a few minutes of typing.

The columns are Item, Type, **Size (hours)**, **Depends on** and **Release**. **Depends on** is chosen from the items you have already typed, so enter the rows in the order below and each dependency will be there when you need it.

Item	Type	Size (hours)	Depends on	Release
Sign in with SSO	Story	T(8,12,20)		R1
Sign out	Story	T(2,3,5)	Sign in with SSO	R1
Password reset	Story	T(8,12,20)	Sign in with SSO	R1
User profile page	Story	T(6,10,16)	Sign in with SSO	R1

Item	Type	Size (hours)	Depends on	Release
Team roster list	Story	T(8,12,20)		R1
Invite a teammate by email	Story	T(8,14,24)	Team roster list	R1
Role permissions	Story	T(12,20,32)	Team roster list	R1
Audit log of sign-ins	Story	T(6,10,16)	Sign in with SSO	R1
Sign-in error messages	Story	T(2,3,6)	Sign in with SSO	R1
Onboarding screens	Story	T(4,6,10)	User profile page	R1
Project dashboard	Story	T(12,20,32)	Role permissions	R2
Create and edit a project	Story	T(8,14,24)	Project dashboard	R2
File upload to a project	Story	T(12,20,32)	Create and edit a project	R2
Search across projects	Story	T(8,14,24)	Project dashboard	R2
Email notifications	Story	T(6,10,16)	Invite a teammate by email	R2
Export a project to PDF	Story	T(8,12,20)	Create and edit a project	R2
Activity feed	Story	T(6,10,16)	Project dashboard	R2
Mobile layout pass	Story	T(8,12,20)		R2
Settings page	Story	T(4,8,12)	User profile page	R2
Help center links	Story	T(2,3,5)		R2

The four rules the window enforces, and what they mean.

A size is a spread, never one number. T(8,12,20) means at least 8 hours, most likely 12, at most 20. The window refuses a bare number, because one number would throw away the uncertainty the plan actually has.

Row order is priority. Use the ↑ and ↓ buttons on a row to move it. The team works down the list.

Depends on names other rows. An item waits in the **Backlog** until every row it names is **Done**.

Release is filled on every row or on none, and an item may not depend on something in a later release, because it would wait forever. Both are refused before the run, not during it.

The window has no OK and no Cancel: every keystroke is already in the model. Close it with the ✕ or Escape. Ctrl+Z undoes an edit.

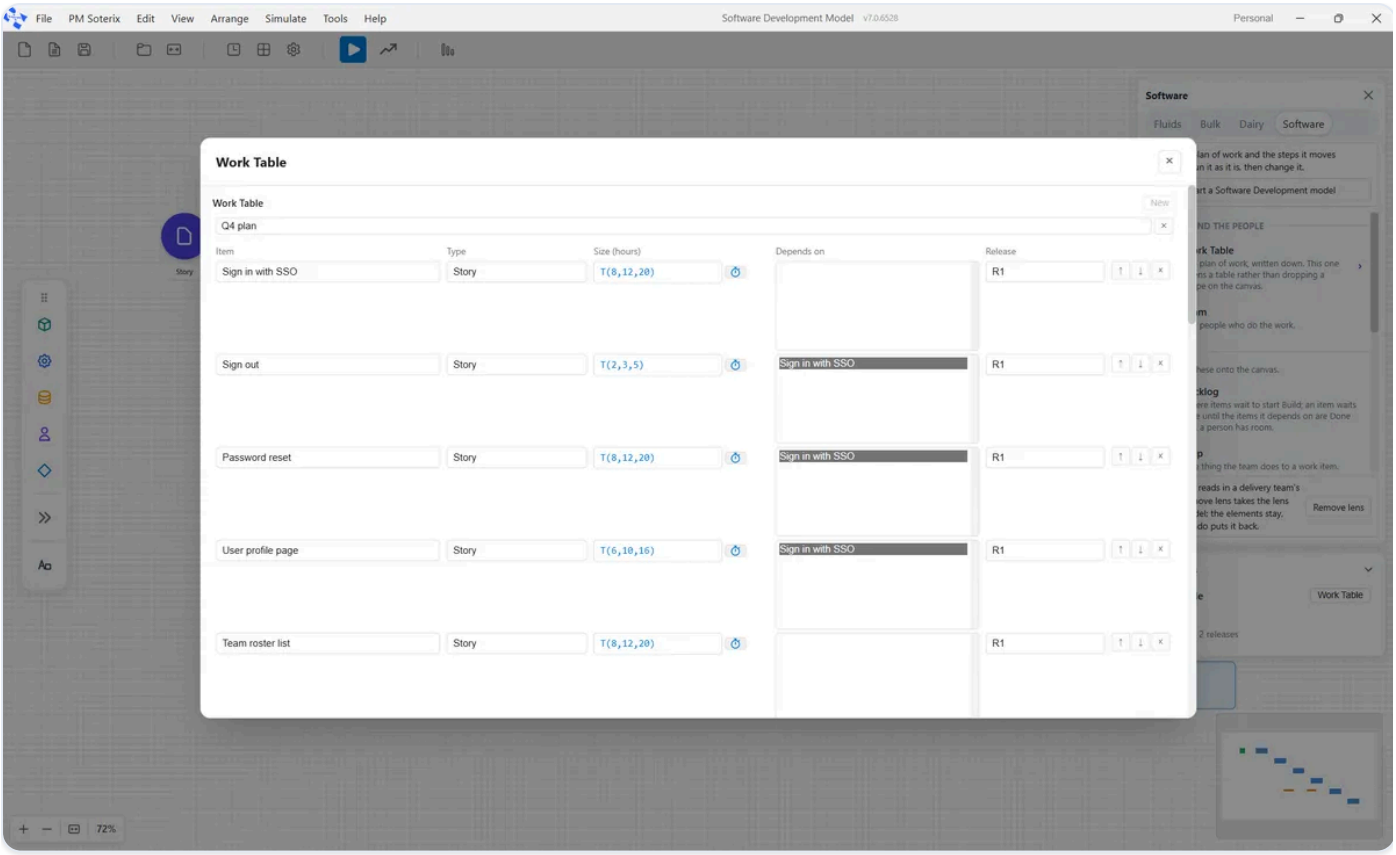

Figure 12. The Work Table window with the twenty rows typed in.

5.6 Draw the arrows

An arrow is drawn by dragging from one element onto another. Hover over an element and small dots appear around its edge; drag from a dot on the first element and drop on the second. What kind of arrow you get depends on what you joined.

The route through the steps

Drag from **Backlog** to **Build**. A plain arrow appears.

Do the same from **Build** to **Review**, **Review** to **Test**, **Test** to **Release**, and **Release** to **Done**.

Nothing needs typing on these. Each is born carrying 100 percent, which is right when a step has only one way out. If you ever draw two arrows out of one step, that is when the **Probability** field on each of them starts to matter.

The arrival

Drag from Story to **Backlog**. This one is an arrival, not a route: it is how work enters the model.

With the arrow selected, look at the top of the properties panel. There is a dropdown there showing the kind of arrival, with no label beside it; out of the box, it reads Periodic. Open it and choose **Work Table** (described as Items from the model's **Work Table**, in row order).

Three fields appear on the **General** tab. Set **Work Table** to the plan you named. Leave **Pace** on All at start, which puts every row into the **Backlog** on day one, in row order.

The other choice, N per sprint, feeds the plan in a few rows at a time and asks for **How many per sprint** and **Sprint length**. It is the right choice for a team that plans sprint by sprint, and it is worth trying later.

The people

Drag from the Developers team onto the **Build** step. A dashed line appears. Click it: the panel header reads **Get / Free**, and **Quantity** is 1. That is exactly right: the step takes one developer while it runs and gives them back afterward.

Drag from Developers to **Review** as well.

Drag from Testers to **Test**.

Backlog, **Release** and **Done** get no line. Nobody works them: the **Backlog** is where an item waits, and the other two are instants.

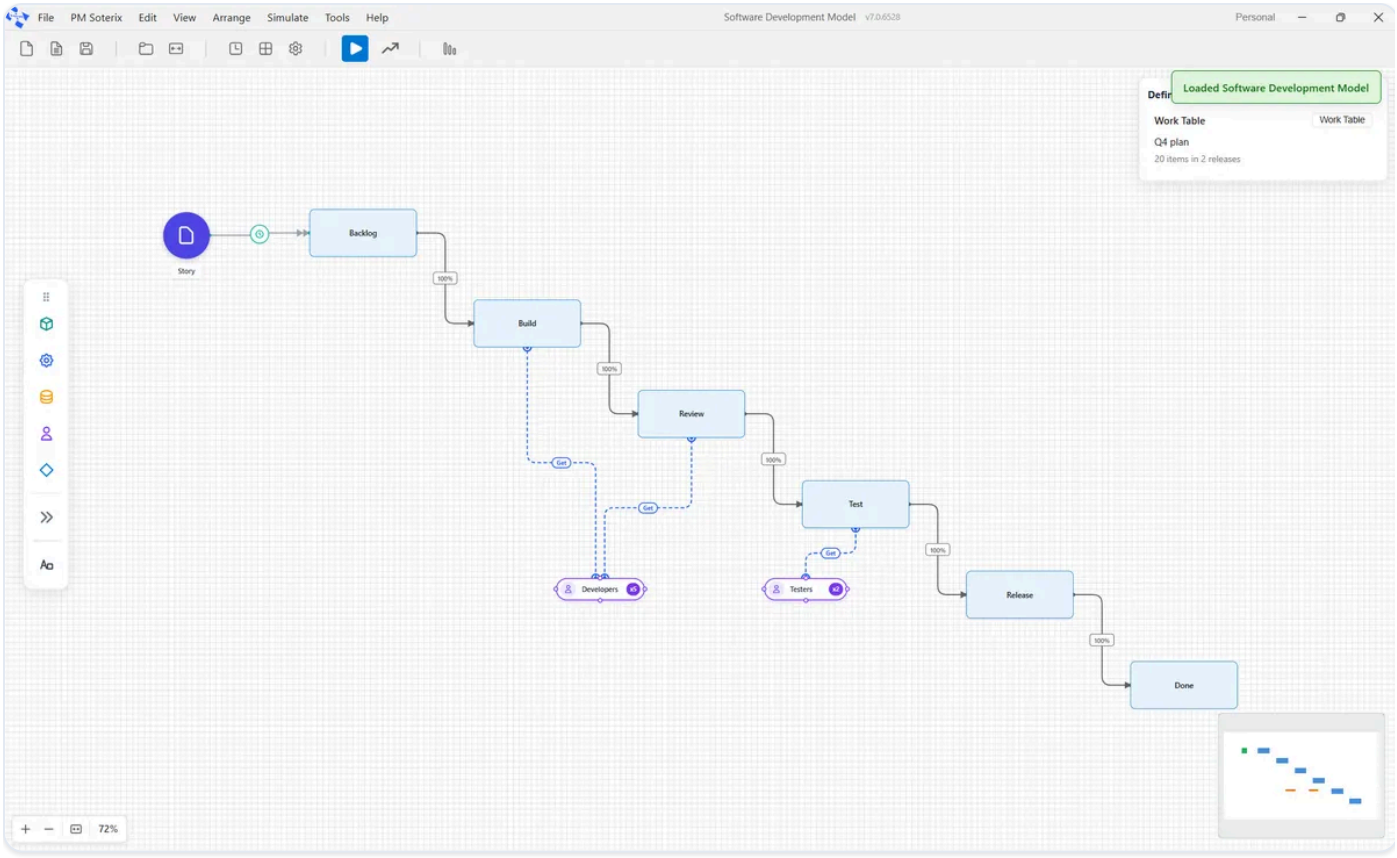

Figure 13. The finished model: Story and the six steps joined left to right, with the two teams below and their dashed lines to Build, Review and Test.

5.7 Say how long each step takes

Click each step in turn and set two things on its **General** tab: **Effort**, which is how long the step takes one person, and the middle box of the three small boxes, labelled **People**, which is how many people may be on that step at once.

Step	Effort	Time Unit	People (on the step at once)
Backlog	0	M	leave as it is
Build	a_hours * 60 (see below)	M	5
Review	T(0.5,1,2)	H	5
Test	T(2,4,8)	H	2
Release	0	M	999
Done	0	M	leave as it is

Two things about **Effort** that will catch you out. First, the unit is minutes until you change it. Under this lens the field is described as a spread of hours, but the **Time Unit** control beside it starts on M. Typing T(2,4,8) and leaving the unit alone gives you a test that takes four minutes. Click H for the **Review** and **Test** steps. Second, **Build** is written in minutes on purpose. a_hours is the size this item drew from its row of the plan, in hours. The **Build** step has to spend exactly that, and a step's time is counted in minutes, so the model says a_hours * 60 with the unit left on M. Type it exactly like that.

The **Release** step needs room for every item that could be waiting to ship, which is why it gets 999 rather than 1. If it is left small, an early release can end up stuck behind a later one, and the product will say so before the run starts.

5.8 Run settings

Open **Simulate** ▸ **Options** (Ctrl+Shift+O).

Set **Run length** to 90 and click Day beside it. A line under the field shows that in weeks. The run must be long enough for every run to finish, or the **Forecast** has nothing to report for the ones that didn't.

Set **Runs** to 30.

Set **Plan date** to 20. This is the day the plan says the work is due, counted from the start of the run; the report then tells you how often the plan was met. Twenty working days is four weeks, which is what a team would have promised for this much work.

Leave **Working day** at 8.

In the Animation section, click the switch to off. Animation plays the run at the pace of the clock and a ninety-day run would take hours. It turns on when a model arrives, so do this before every forecast.

Press **Apply**.

Software models have no warm-up setting, and that is deliberate. A warm-up exists to let a stream of random arrivals settle before the report starts counting. A plan of work is a fixed list, so a warm-up could only throw rows away. The field is hidden while the lens is on, and a model that carries one from somewhere else is refused rather than quietly miscounted.

5.9 Run it and read it

Press **Run Simulation**. Thirty runs of this model take a few seconds.

On **Simulation Complete**, click **Yes**.

Read the five tabs in this order.

Tab	What to read first
Forecast	Read the whole program, then each release. Read the middle reading, Most runs by (P85). That is the date to promise. Under it, where the runs finished, shows the spread.
Flow	Scroll to Blocked, per item. Anything with a large Blocked figure was waiting for another item, not for a person. Then Flow efficiency: the share of each item's elapsed time that anyone was actually working on it. Ten to twenty percent is normal for a real team.
Team	The utilization tiles and the Resources table, which show how busy each team was. The People over time chart, with its Effective capacity line, appears only once a team carries start dates or a learning curve, which is the model in 5.11.
Quality	Items checked, and by the kind of step. With no rework route in this model, the middle card says there is none, which is correct.
Cost	Empty until the teams carry an hourly rate. Set one on each team's Cost tab and run again if you want the money.

If a run does not finish. If the **Forecast** says not reached for some runs, the run length was too short. Raise it in **Simulate** ▸ **Options** and run again. Never shorten the plan to make the report look better; that is the one change that makes a forecast flatter and more optimistic at the same time.

> ⓘ What should this model say? Built as above and run thirty times, the whole program finishes on day 21, day 22, and day 23 for the three readings, with R1 shipping around day 8 or 9 and all twenty items done in every run. Your numbers will be close, not identical, if you typed a size differently. Against a plan date of 20, the report will tell you the plan was missed by a couple of days: two hundred and forty hours of building, five developers, and a two weeks of dependencies is a four-and-a-half week job, not a four-week one.

5.10 Save it

File ▸ **Save As** (Ctrl+Shift+S). Type a **MODEL NAME**, choose a folder, press **Save Model**. One file ending in .pmd holds the drawing, the plan of work, the teams, the lens and the run settings.

5.11 What to try next

Ask a what-if without disturbing the model. **Tools ▸ Logic Builder** (Ctrl+Shift+L), then Scenarios, then **Add new**. A scenario is a named set of changes; each one runs its own thirty runs, and the **Forecast** tab lists them side by side under Scenarios compared. That is how the demo in Part 3 compares four choices.

Add the meetings. On each team's **People** tab, add an interruption row for the stand-up, as in Part 4.

Feed the plan in sprints. Change the arrival's **Pace** to N per sprint and watch the **Flow** tab's Items finished per sprint section fill in.

Model somebody joining late. On a team's **People** tab, add a row of people with a **Starts on day** and a learning curve, and read the **People** over time chart on the **Team** tab. That is the mechanism behind both shipped demos.

Part 6. Reference

6.1 Your words and the product's words

With the lens on, most of these are what you see on screen. The right-hand column is what the same thing is called with the lens off, which is worth knowing when you read general help.

You would say	With the lens on	With the lens off
A story, a ticket, a task	Work item	Entity
The backlog, the sprint plan	Work Table	no equivalent; this is new
A story point estimate	Size (hours), as a spread	no equivalent
The team, the squad	Team, and People for its size	Resource, Quantity
A phase of work	Step, and Kind of step	Activity
How long a step takes	Effort	Processing Time
How many can work on it at once	People (on the step)	Capacity
Working hours	Working hours	Shift Schedule
Blocked	Waiting in the Backlog, reported as Blocked	Queue time
Code review	A Step whose Kind is Review	Activity
QA	A Step whose Kind is Test	Activity
CI, the build pipeline	A Step whose Kind is Pipeline	Activity

You would say	With the lens on	With the lens off
Deploy, ship	A Step whose Kind is Release	Activity
Code freeze	Freeze	A closed gate
Descoped	A Step whose Kind is Dropped	Activity
Meetings, questions, incidents	Interruptions	Interruptions
Onboarding	Learning curve	no equivalent
How many times to run it	Runs	Replications
When will it be done	The Forecast tab	no equivalent

6.2 The numbers in this guide, and where they came from

Every figure quoted for the starter model was measured on build 7.0.6528, thirty runs unless stated. If your screen differs, check that you changed only the field named.

Where	What it reads
Starter model as it arrives, five runs	Whole program day 3 in all three readings. R1 day 2, R2 day 3. All five runs finish.
Starter model, thirty runs	Whole program day 3 / day 4 / day 4. R1 day 2 in all three.
Change 2, a narrowed estimate	Whole program day 3 / day 3 / day 3.
Change 3, a dependency removed	Whole program day 2 / day 2 / day 2.
Change 4, a ninety-minute stand-up	Whole program day 4 / day 4 / day 4.
Change 5, eight people or one	Whole program day 3 / day 4 / day 4, unchanged either way.
Change 5, Build limited to one person	Whole program day 4 / day 4 / day 4.
Three More People demo, at thirty runs	Do nothing day 347. Add three at month three, day 339. Add three at month one day 325. Cut fifteen percent day 301. All at Most runs by (P85), against a plan of day 273. The shipped demo defaults to fewer runs.

The modeling assumptions in the two demos come from published research: a new person starting at half speed and reaching full speed after eighty working days, a fifth of an experienced person's time going to each learner, and a coordination cost that grows with the number of pairs on a team. The demo briefs name their sources. Replace them with your own figures before quoting them to anyone.

6.3 Mistakes that waste a run

Typing a time in the wrong unit. The **Time Unit** control starts on M. A two-hour test typed as 2 with the unit untouched is a two-minute test.

A Type in the plan that matches no element. If the plan says Story, the model needs an element named Story. The rows have nothing to become otherwise.

A size typed as a single number. The window refuses it. A plan of single numbers finishes on nearly the same day every run, and the **Forecast** then tells you nothing.

A team with no working hours. It works around the clock and every date is optimistic. Set **Working hours** on the **Shift** tab.

Leaving a team on Finish Current. A person handed a ten-hour item on Monday morning works until half past seven that night to finish it, and the team books hours nobody worked. Teams want Preempt & Wait.

Promising the first column. Half the runs by is the day you beat half the time, so promising it means being late half the time. Promise Most runs by (P85).

Adding people to a plan that cannot use them. Look at Blocked, per item on the **Flow** tab first. If the work is waiting on other work, more people change nothing.

Running one replication and believing the date. One run is one possible future.

6.4 Where everything is

To do this	Go here
Start an empty model	File ▸ New (Ctrl+N), or Blank Canvas on the Welcome screen
Open a model you saved	File ▸ Open (Ctrl+O), or Open from Computer on the Welcome screen
Open one of the shipped examples	File ▸ Demo Models, then the Industry list on the left
Show the Software elements	View ▸ Show Industry Palettes, then the Software tab
Build a working software model in one click	The Start a Software Development model button on the Software page

To do this	Go here
Open the plan of work	The Work Table tile on the Software page
Change what an element does	Click it, and use the panel on the right
Reach a hidden properties tab	The eye icon at the foot of the tab icons
Set the run length and how many runs	Simulate ▸ Options (Ctrl+Shift+O)
Run	The play button on the toolbar, or Simulate ▸ Run Simulation (Ctrl+R)
Open the report again	Tools ▸ Output Report (Ctrl+Shift+R)
Compare what-if cases	Tools ▸ Logic Builder (Ctrl+Shift+L), then Scenarios
Tidy the drawing	Arrange ▸ Auto-Layout
Save	File ▸ Save (Ctrl+S) or File ▸ Save As (Ctrl+Shift+S)

List of figures