processmodel™

SELF TEACH GUIDE

# ProcessModel 7 Self Teach Guide

Fifteen short chapters take you from your first model to the certification test. Each one is a video and a written lesson you work through in ProcessModel, a step at a time, as one phone repair shop grows into a model you can trust.

# Contents

01 Welcome and the Window

02 Your First Model

03 Make Your Times Vary

04 Make Customers Come When They Really Do

05 Send Each Job Down the Right Path

06 Give Each Line a Size and an Order

07 Give Your Technicians Real Working Hours

08 Tag Each Customer and Count the Jobs

09 Make Each Repair Wait for Its Part

10 Run the Day Ten Times, Not Once

11 Find Where Your Customers Really Wait

12 Try Two Fixes Side by Side

13 Let the Software Find the Best Setup

14 Let Your Own AI Build the Next Shop

15 Build a Shop on Your Own, Then Take the Test

CHAPTER 1 OF 15

# Chapter 1. Welcome and the Window

By the end of this chapter you can

**Find your way around ProcessModel and run your first demo model.**

Watch the video (5 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 1. Welcome and the Window**
www.youtube.com/watch?v=sJyh3LWpR6k

## Before you start

Welcome to the Self Teach Guide. It has fifteen chapters, and from Chapter 2 on you build one model, a phone repair shop, that grows a little in every chapter. Each chapter has a short video, this written lesson, a practice model, three questions to check yourself and one small task to try on your own.

This first chapter builds nothing. Its job is to make the window familiar, so that later chapters can talk about the toolbar, the palette or a shape's properties and you know at once where to look. You do that the best way there is: by opening a finished model, looking inside it, and running it.

**You need** ProcessModel installed on your computer (see System requirements) and about twenty minutes. A trial, a student license or a Professional license all work the same for this course.

**How to use this lesson.** Keep ProcessModel open beside this page and do each step as you read it. After each step, the lesson tells you what you should see. If your screen does not match, stop and look again before moving on: the small things you notice now save you time in every later chapter.

Step 1.

## Start on the Welcome screen

Start ProcessModel. It opens on the **Welcome screen**, the place where every session begins: you start a new model here, or pick up one you worked on before.

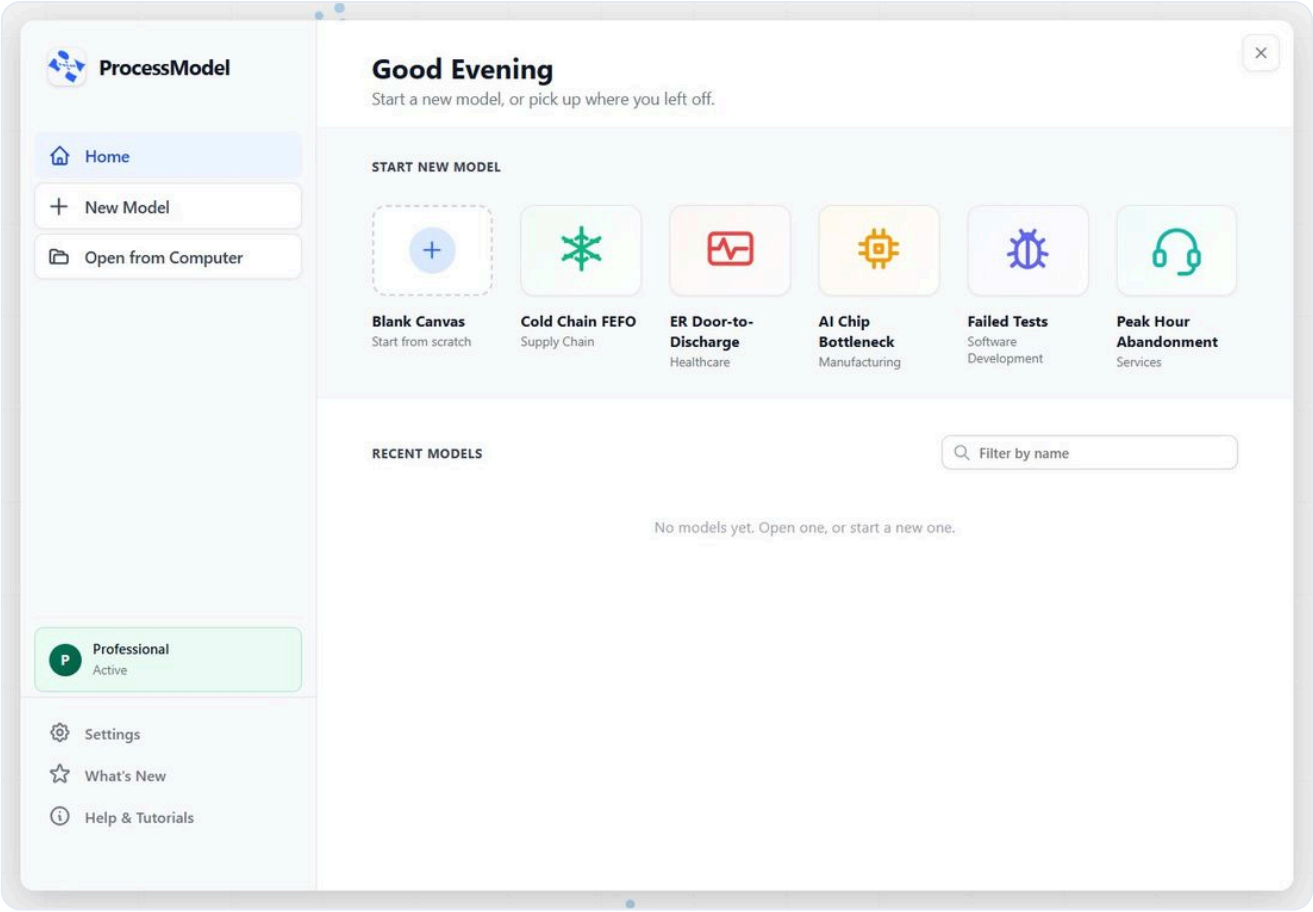



Figure 1. The Welcome screen, the first thing ProcessModel shows.

**What you should see:** a greeting that follows the time of day, a row of tiles under **Start New Model**, and a **Recent Models** list. On a first start the list is empty; it fills as you save models.

---

Step 2.

## Read the left list and your license

The list down the left side is your way into everything the Welcome screen does. Rest your eyes on each row:

- **Home** is the screen you are looking at.
- **New Model** starts an empty model.
- **Open from Computer** opens a model you saved earlier.
- The green row shows your **license**. In the picture it reads Professional; yours may read Trial or Student. It only reports: it is there so you always know which license you are working under.

- **Settings**, **What's New** and **Help & Tutorials** sit at the bottom.

Your license also shows at the top right of the window, in the title bar, so you can see it even after the Welcome screen is gone.

Now look at the tiles. **Blank Canvas** starts an empty model; you will use it in Chapter 2. The other five are finished demo models, each from a different industry. The last one, **Peak Hour Abandonment**, is a call center, and it is the model this chapter opens. You will not open it from the tile, though: you will open it from **Demo Models**, the library where all the demos live, because that is where you will find the course's practice models from Chapter 3 on.

---

Step 3.

## Open Demo Models

1. Close the Welcome screen with the **X** at its top right. You see an empty canvas, ready for a model.
2. Click **File** in the menu bar, then **Demo Models**.

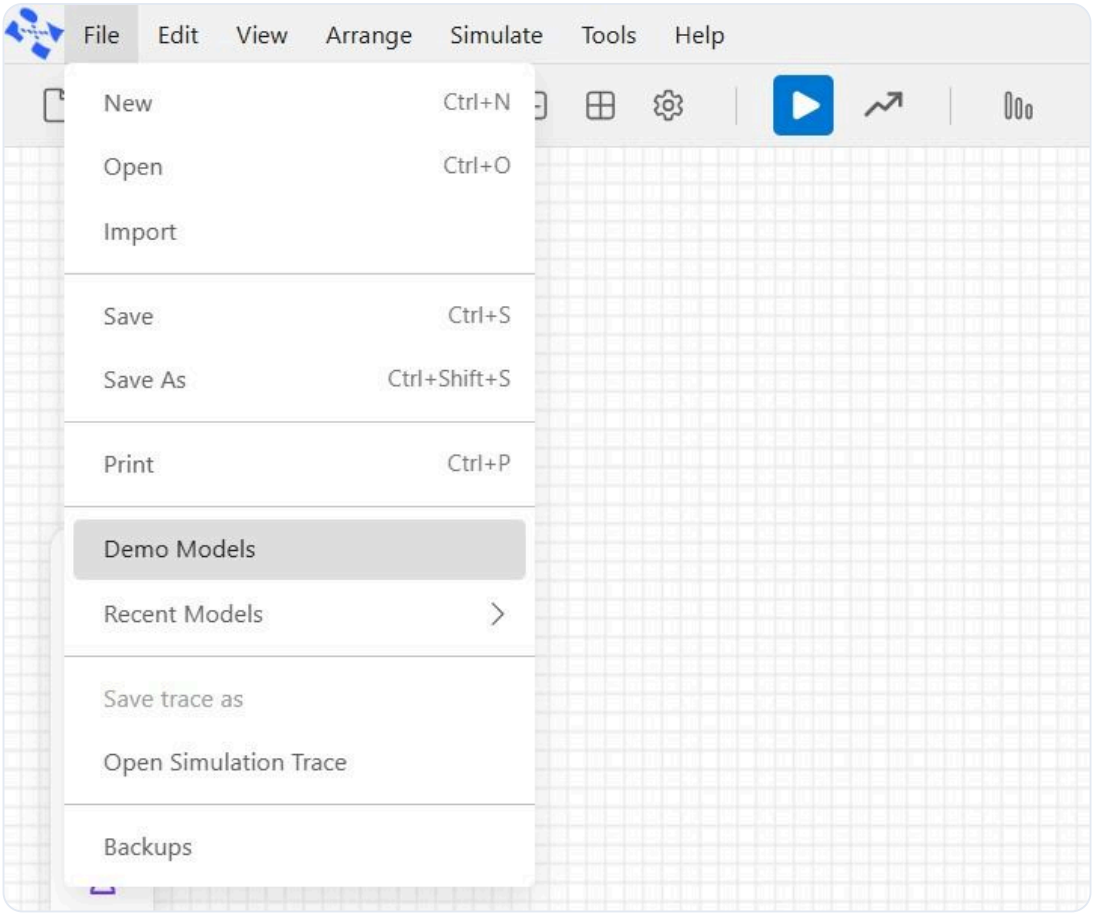



Figure 2. File ▸ Demo Models. The same library also opens from its own button on the toolbar.

There is a second way in: the toolbar under the menus has a **Demo Models** button. Rest the pointer on any toolbar button and its name shows, which is the quickest way to learn them.

> **What you should see:** the **Demo Models** window. Down its left side is a list of industries; on the right, a card for each demo.

---

Step 4.

## Read the brief, then open the model

Every demo is a finished model built around one problem. Reading the problem first is the habit that makes a model make sense: you know what it is for before you look at how it is built.

1. In the list on the left, click **Services**. The cards jump to the services demos.

2 Find the **Peak Hour Abandonment** card. Its line tells the problem in one sentence: about 3 in 10 callers hang up before an agent answers.

3 Click **Read the brief** on that card.

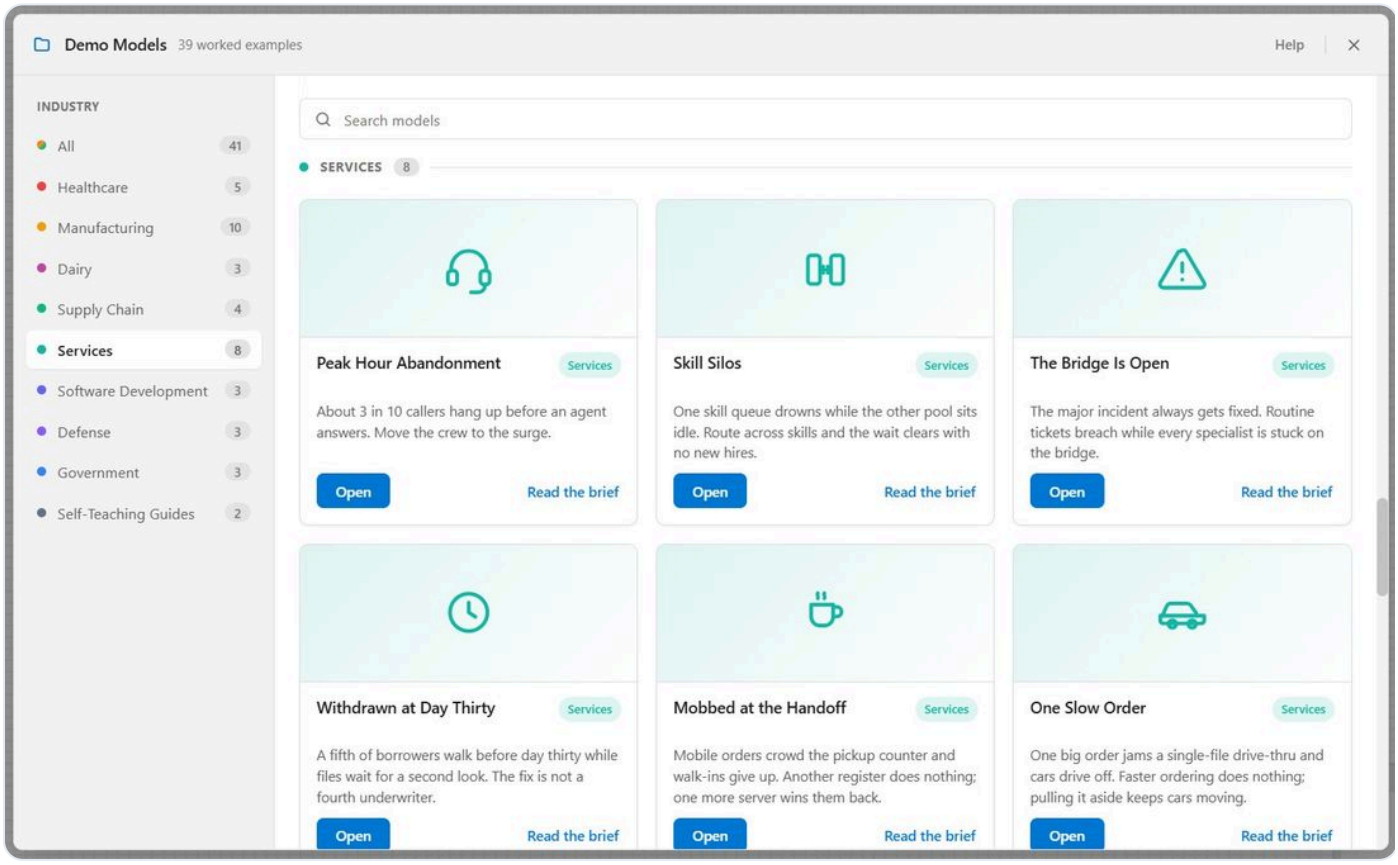



Figure 3. Demo Models, with the Services group in view.

The brief opens beside the cards. It tells you the problem, how the model is set up, and what to try. Read it now: it takes a minute, and it is the story behind everything you are about to see.

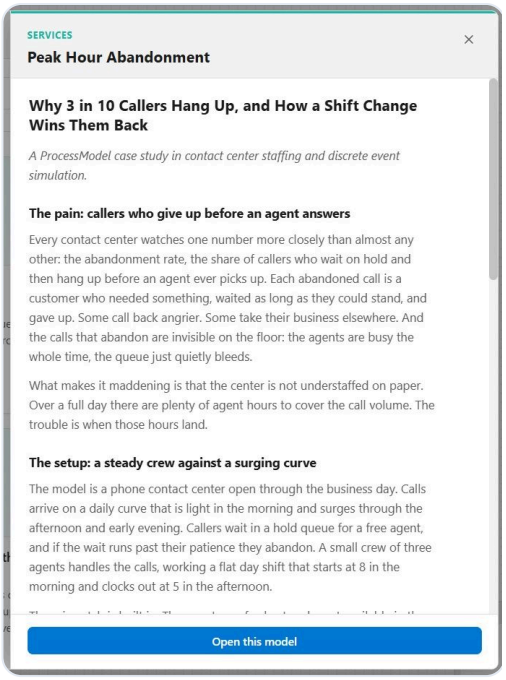



Figure 4. The brief: the problem, the setup and what to try, with Open this model at its foot.

4 Click **Open this model**.

> **What you should see:** the call center on the canvas, with its name, Peak Hour Abandonment, in the title bar. A demo always opens as a **new model**, so whatever you change, the original in Demo Models never changes. You can explore freely.

---

Step 5.

## Learn the four parts of the window

Now take in the whole window. Everything you do in ProcessModel happens in one of four places, and every later chapter names them.

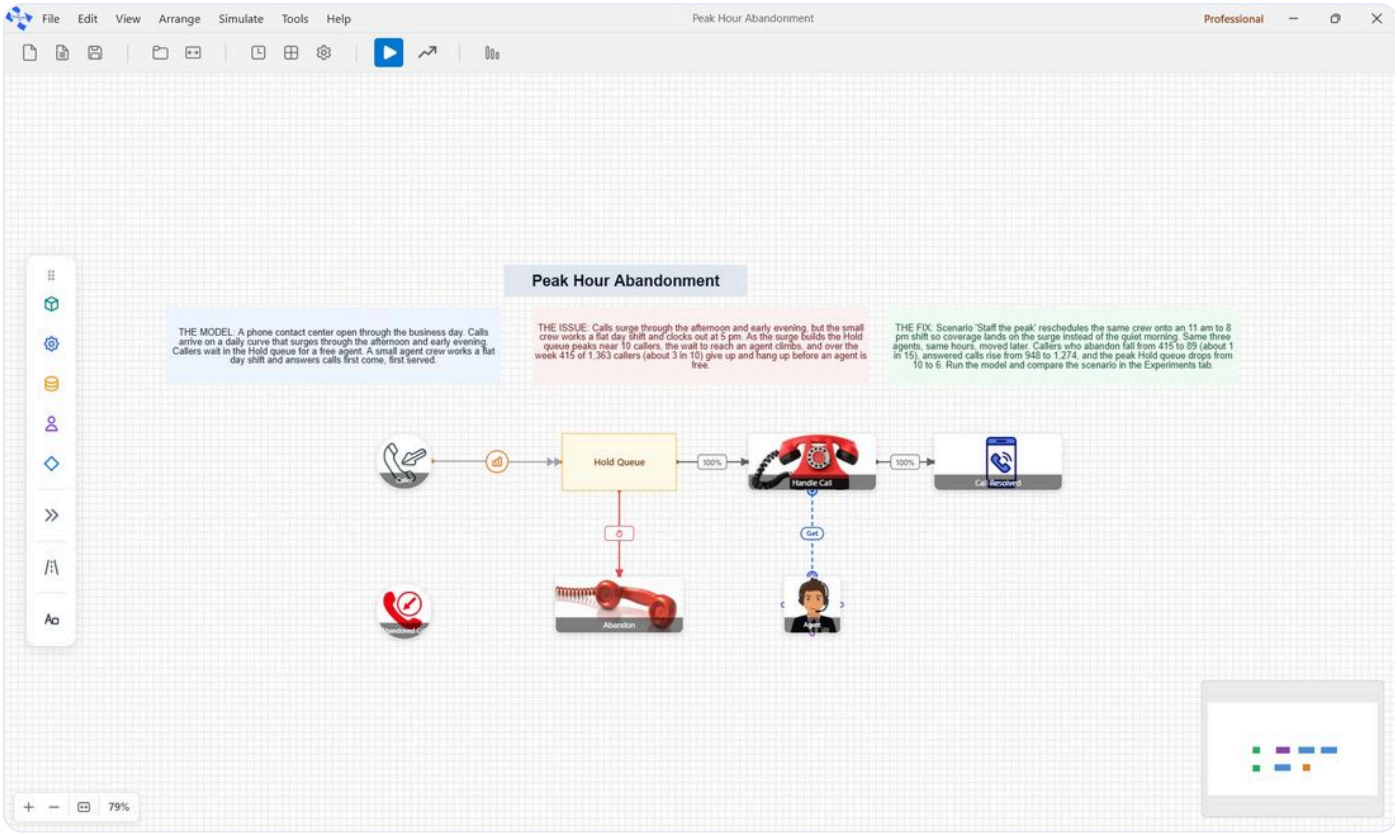



Figure 5. The window: menus, toolbar, palette and canvas.

1. **The menus** across the top: File, Edit, View, Arrange, Simulate, Tools and Help.
2. **The toolbar** under them holds the commands you use most, one click each: from New Model and Demo Models to Simulation Options, Run Simulation and the Output Report. Rest the pointer on a button to see its name.
3. **The palette** runs down the left side of the canvas. It holds the shapes you drag in to build a model: entities, activities, queues, resources and more. Its buttons name themselves the same way when you rest on them.
4. **The canvas** is the rest of the window, where the model is drawn.

Read the canvas from left to right, the way the work flows. Callers arrive, wait in the **Hold Queue** until an agent is free, and are handled at **Handle Call**. A caller who waits too long hangs up and goes to **Abandon** instead. The **Agent** under Handle Call is the people who take the calls. The three notes above the flow tell the same story as the brief.

Step 6.

## Open a shape's properties

A shape on the canvas is only a picture until you look at its **properties**: the settings that tell ProcessModel how that part of the process behaves. Every shape keeps its settings in the same place.

1 Double-click in the middle of **Handle Call**. Its properties open on the right.

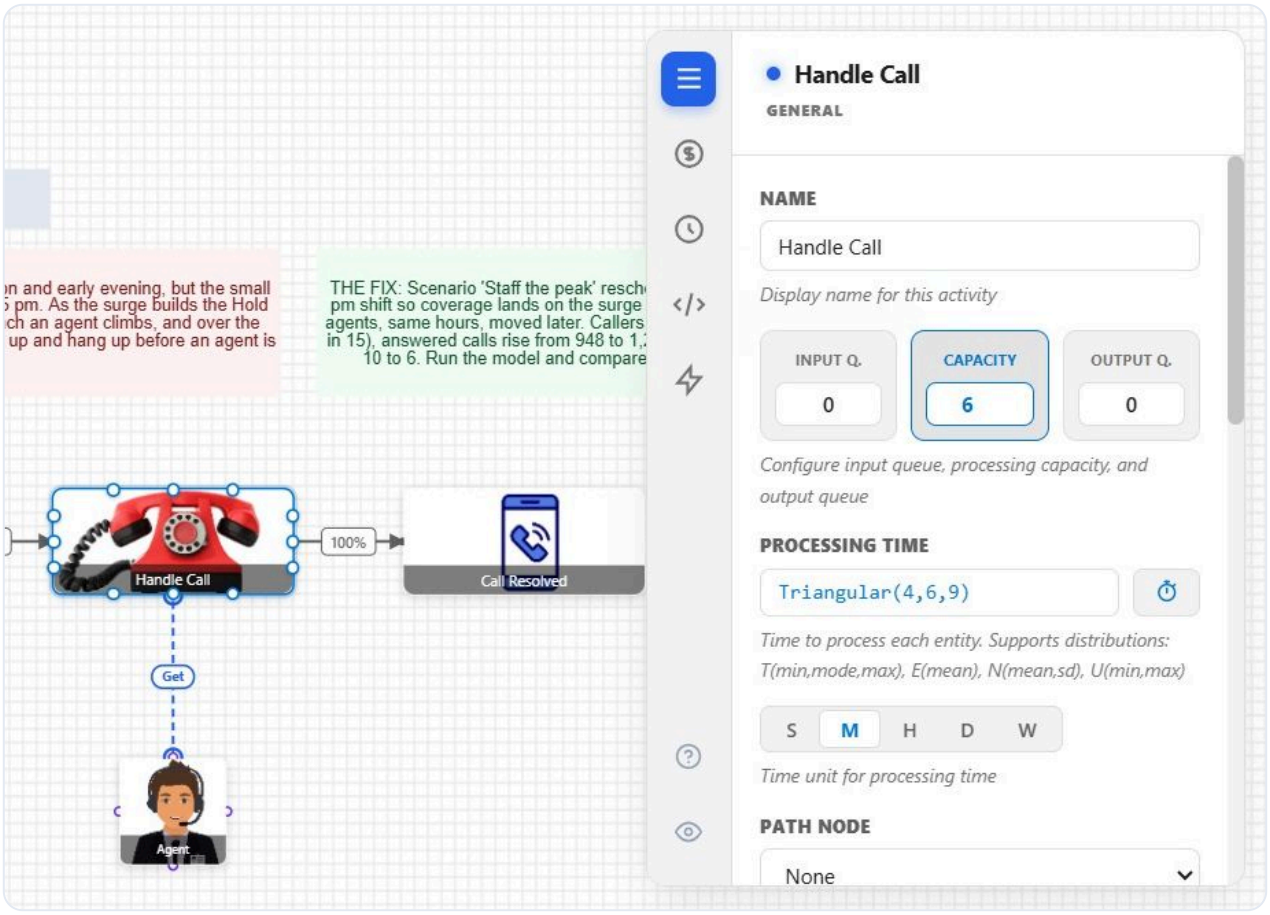



Figure 6. Handle Call's properties: its name, how many calls it takes at once, and how long each call takes.

Handle Call is an **activity**, a step where work is done. Three of its settings carry most of its meaning:

- **Name** is what the shape is called on the canvas and in the results.
- **Capacity** is how many calls it can handle at the same time.
- **Processing Time** is how long each call takes. The unit row under it says what the number means: **M** is chosen, so the time is in minutes.

2 Click an empty part of the canvas. The panel closes.

3. Now double-click **Agent**, under Handle Call. Agent is a **resource**: the people (or machines) an activity needs. Its panel shows its **Name** and its **Quantity**, how many agents there are.
4. Click the empty canvas again to close it.

**Why this matters:** from Chapter 2 on, you set these values yourself. Knowing that every shape opens the same way, with a double-click, and closes with a click on empty canvas, is most of what you need to work with any shape.

---

Step 7.

## Run the model five times

A model earns its keep when it runs: ProcessModel plays the process forward through time and counts what happens. You will run this one five times, not once. Calls come in at random, so one run is one possible week; five runs average out that chance. Chapter 10 explains why in full. For now, it is enough to know that more than one run gives an answer you can lean on.

1. On the toolbar, click **Simulation Options**. The window opens on its **Run** group, the settings of the run: how long the model runs, and how many times.
2. In **Replications**, type **5**. Read the line under it: animation is off while more than one runs. Five runs are played without drawing them, so they finish in seconds.
3. Look at the foot of the window: it names the settings that **Apply** will change, Replications and Animation. It is a good habit to read this line before you apply, because it tells you exactly what you are about to change.
4. Click **Apply**.

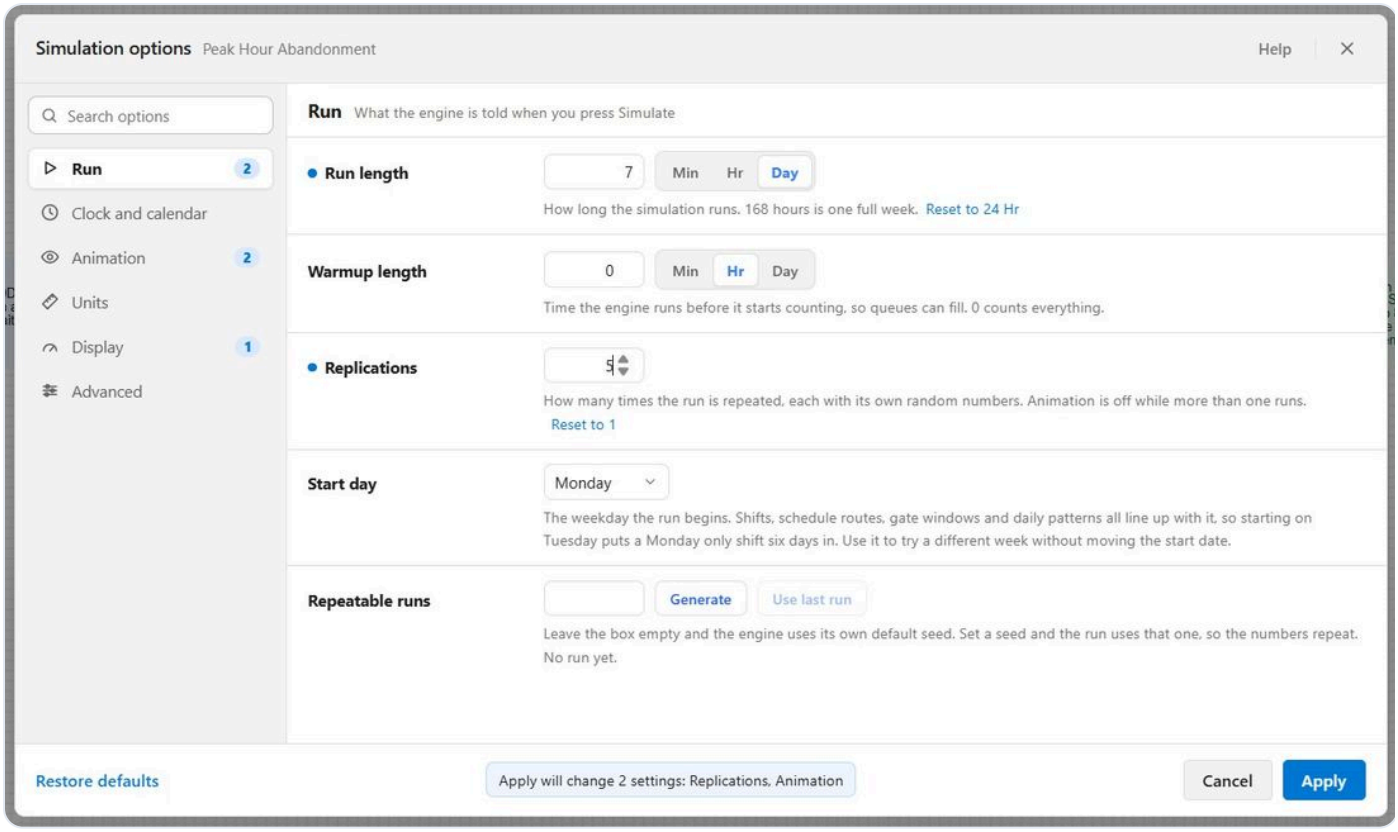



Figure 7. Simulation Options, with Replications set to 5.

5 On the toolbar, click **Run Simulation**.

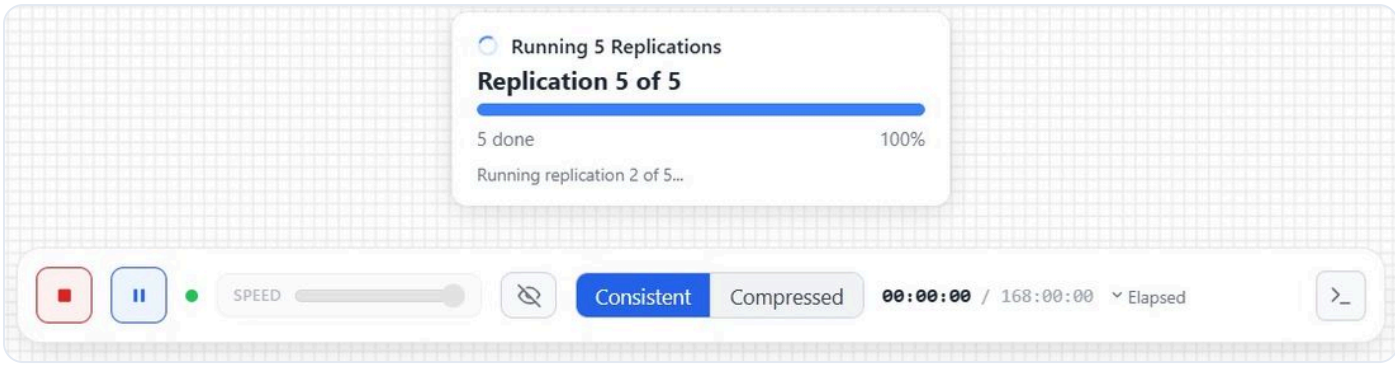



Figure 8. While the five runs play, a card counts them. The speed slider is grayed out, because nothing is being drawn.

**What you should see:** a card that counts the runs, and the run bar at the foot of the canvas with **Stop** and **Pause**. The speed slider is grayed out: with animation off, there is no speed to set. In a few seconds the runs are done and a question appears.

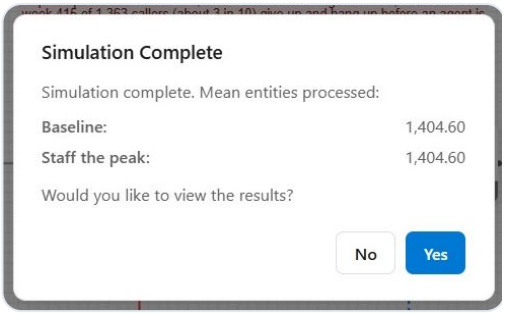



Figure 9. Simulation Complete asks whether you want to see the results.

**Simulation Complete** reports how many calls went through on average, for the model as it is (Baseline) and for a scenario the demo carries. Scenarios are how you try a change side by side with today's process; Chapter 12 teaches them.

---

Step 8.

## Glance at the results

1. Click **Yes**. The results open in the **Output Report**.

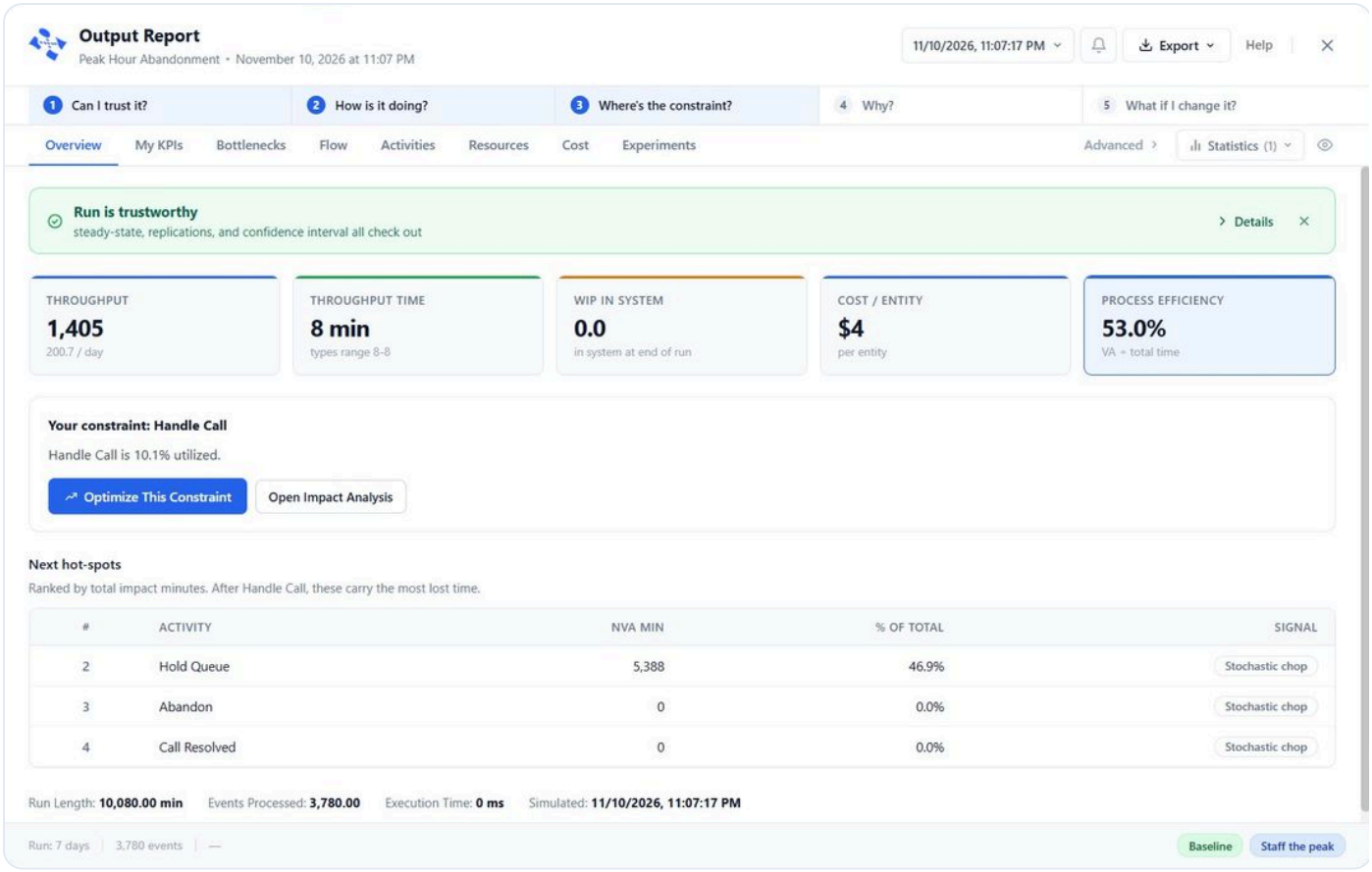



Figure 10. The Output Report opens on its five questions, with a green line that says the run is trustworthy.

The report is built around five questions, the five a manager asks about any process: **Can I trust it? How is it doing? Where is the constraint? Why? What if I change it?** The green line near the top says the run is trustworthy: the five runs agree well enough to rely on. You do not need to read the rest yet: Chapter 11 is devoted to it.

2 Close the report with the **X** at its top right.

---

Step 9.

## Know where Help is

1 Click **Help** in the menu bar.

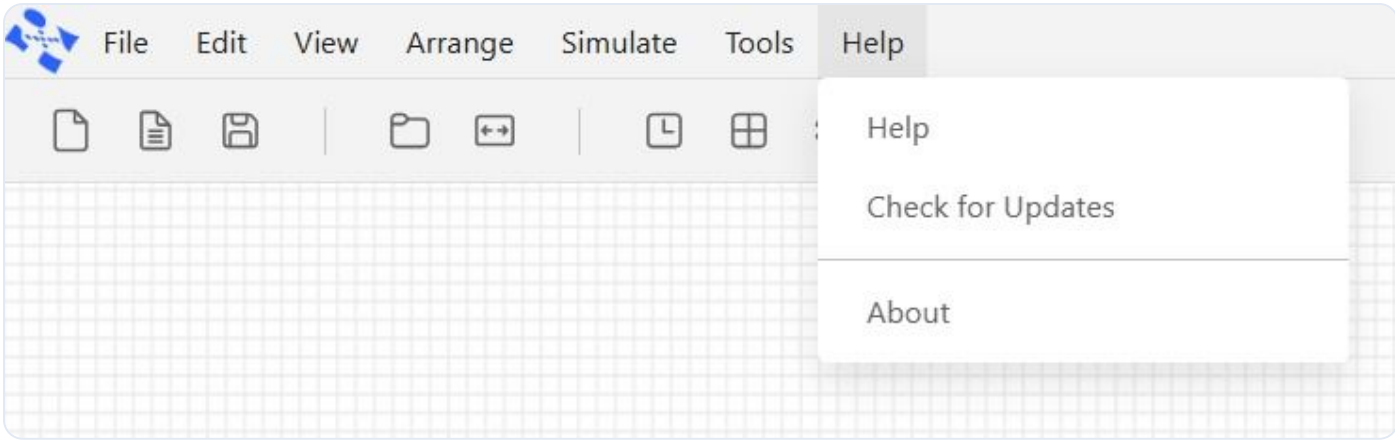



Figure 11. The Help menu.

**Help** opens the website's help, these pages, in your browser, whenever you need it. **Check for Updates** looks for a newer version of ProcessModel, and **About** shows which version you have.

2 Press **Escape** to close the menu.

When you close the demo, ProcessModel asks whether to save your changes, because you changed its run settings. Choose **Don't Save**: the demo in Demo Models stays as it was either way.

## Your practice model

This chapter's practice model is the demo you just opened: **Peak Hour Abandonment**, in **File ▸ Demo Models**, under **Services**. There is no model of the course's own for this chapter. Your own phone repair shop starts from an empty canvas in Chapter 2.

## Check yourself

QUESTION 1

**You open a demo model and change its settings. What happens to the demo in Demo Models?**

A It changes too, so the next person sees your settings

B Nothing: a demo opens as a new model, so the original never changes

C ProcessModel refuses the change until you save a copy

The answer is B. A demo always opens as a new, unsaved model, so you can explore and change it freely.

QUESTION 2

**Handle Call's Capacity reads 6. What does that mean?**

A Each call takes 6 minutes

B Handle Call can hold up to six calls at the same time

C Six agents take the calls

The answer is B. Capacity is how many a step can work on at once; each call still needs a free Agent. How long a call takes is its Processing Time, and how many agents there are is the Agent's Quantity.

QUESTION 3

**With Replications set to 5, the run plays without animation and the speed slider is grayed out. Why?**

A Something went wrong with the model

B With more than one run, animation is off, so the runs finish in seconds and there is no speed to set

C The demo locks the slider so the original cannot change

The answer is B. The line under Replications says so: animation is off while more than one runs. Several runs give an answer you can lean on; Chapter 2 shows a single run moving on screen.

## Try it yourself

Open a second demo on your own and find your way around it, without the steps above.

1. In **File** ▸ **Demo Models**, pick any industry other than Services and open a demo that interests you. Read its brief first.
2. Name its activities and resources by reading the canvas, then double-click one activity and find its **Capacity** and **Processing Time**.
3. Open **Simulation Options**, make sure **Replications** reads **1** (type it and click **Apply** if it does not), and click **Run Simulation**.

**What to expect.** With one run, the model is drawn as it runs: work flows from shape to shape, and the speed slider in the run bar is live, so you can slow the run down to watch or speed it up. When it ends, Simulation Complete asks the same question as before. Answer **No** this time. Chapter 2 shows you what to read on the canvas instead, and Chapter 10 explains why one run is only one possible outcome.

CHAPTER 2 OF 15

# Chapter 2. Your First Model

By the end of this chapter you can

**Build a model from an empty canvas, run it and save it.**

Watch the video (6 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 2. Your First Model**
www.youtube.com/watch?v=1IZjVB6V2DI

## Before you start

In this chapter you build your first model from nothing: a small **phone repair shop**. It is the model the whole course grows, so take your time with it and save it at the end. Chapter 3 starts from the shop you save here.

Every process model answers the same three questions, and you will answer each of them in turn:

1. **What flows through the process?** Here, customers, each with a phone to fix.
2. **What happens to them, step by step, and how long does each step take?**
3. **Who or what does the work?** Here, a Technician.

Here is the shop you will build. Read it now, because every number you type comes from this list.

| Part | Set to |
|---|---|
| Customers | One walks in every 10 minutes |
| Check In | 3 minutes at the counter screen, with no staff |
| Repair | 12 minutes, done by a Technician |
| Pick Up | 2 minutes at the counter screen, with no staff |
| Staff | One Technician |

| Part | Set to |
| --- | --- |
| The run | One shop day, 8 hours, run once so you can watch it |

Look at the numbers before you build anything. A customer arrives every 10 minutes, but Repair takes 12. What do you expect will happen during the day? Keep your guess in mind: the run will tell you whether you were right.

**You need** about thirty minutes. The times in this chapter are fixed on purpose, so the model is easy to follow. Chapter 3 makes them vary, as real work does.

---

Step 1.

## Start with an empty canvas

1. Start ProcessModel. On the Welcome screen, under **Start New Model**, click **Blank Canvas**.

> **What you should see:** an empty canvas, and down its left side, the palette.

2. Rest the pointer on the first shape in the palette. Its name shows beside it: **Entity**. Do the same down the column and you meet **Activity**, **Queue**, **Resource** and the others.

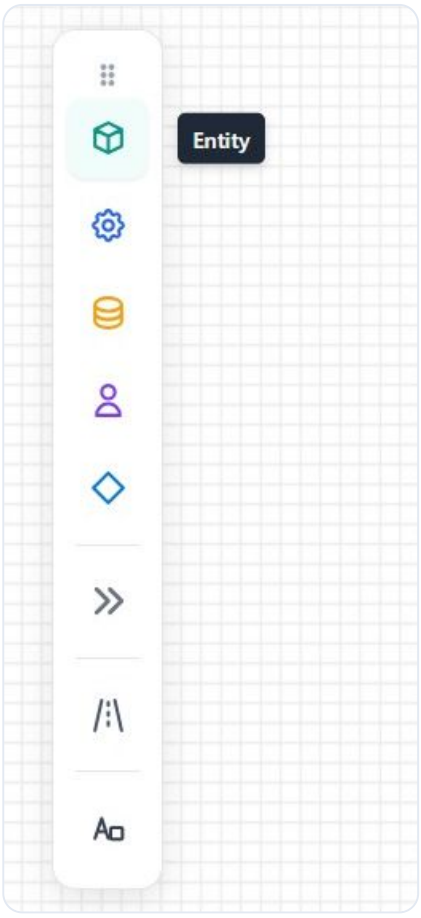



Figure 1. Rest the pointer on a palette button to see its name.

Step 2.

# Add the customer

An **entity** is whatever flows through a process: a patient, an order, a call. In this shop it is a customer with a phone to fix.

1. Drag the **Entity** button from the palette onto the canvas, near the left. Its properties open on the right.
2. Click in **Name**, select what is there, and type **Customer**.

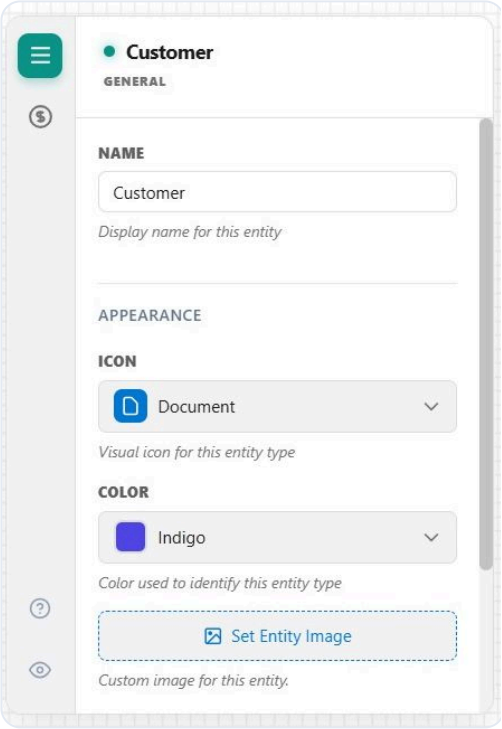



Figure 2. The entity's properties, named Customer.

**What you should see:** a round shape on the canvas labeled **Customer**.

Step 3.

## Add the three steps

Each step a customer goes through is an **activity**: a place where something is done to the entity and time passes. The shop has three: Check In, Repair and Pick Up.

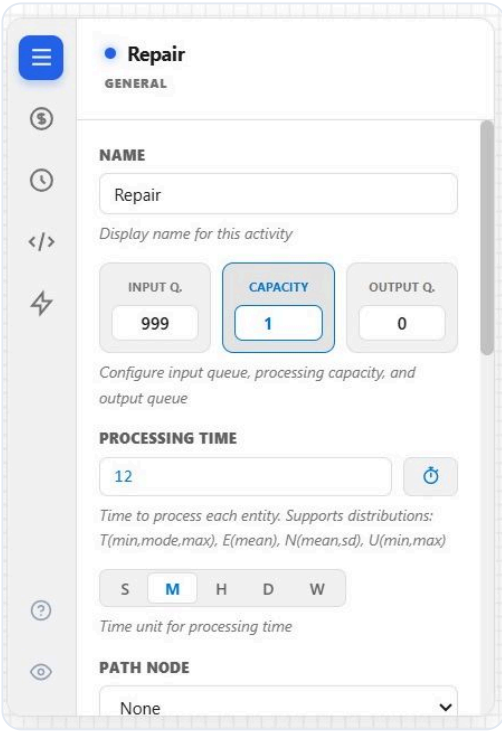



Figure 3. Repair's properties: 12 minutes, with M chosen. Capacity is 1, so one repair at a time.

1. Drag an **Activity** onto the canvas, to the right of Customer, and name it **Check In**.
2. Find **Processing Time**: how long one customer spends at this step. Type **3**.
3. Look at the unit row under it: **S, M, H, D, W**, for seconds, minutes, hours, days and weeks. **M** is already chosen, so 3 means 3 minutes. Click **M** anyway, to make it your habit to check the unit every time you type a time.
4. Click an empty part of the canvas to close the panel.
5. Drag a second Activity to the right of Check In. Name it **Repair**, type **12** in Processing Time, and check that **M** is chosen.
6. Drag a third Activity to the right of Repair. Name it **Pick Up**, with **2** minutes.

Notice **Capacity** above Processing Time. It reads 1: Repair can work on one customer at a time. You will come back to it at the end of the chapter.

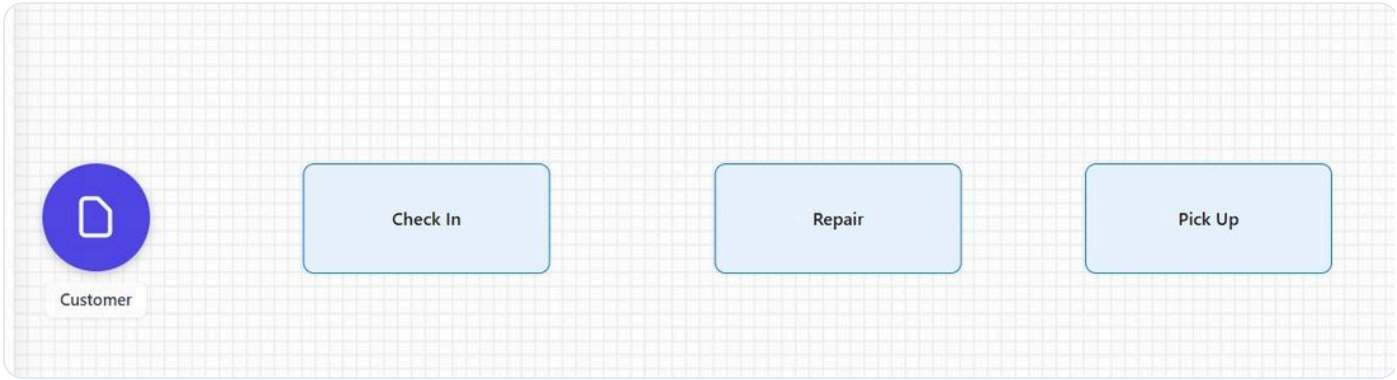



Figure 4. The customer and the three steps, not joined yet.

Step 4.

## Draw how customers arrive

Shapes on their own do nothing. The lines between them carry the customers, and the first line, from the entity to its first step, is special: it is the **arrival**, and it says how and when customers come in.

1. Rest the pointer on **Customer** until small handles show on its edge. Drag from the right-hand handle to **Check In**, and let go.

> **What you should see:** a line from Customer to Check In, with a small clock badge on it.

2. Click the line. Its properties open.
3. Open the type list at the top. It holds several kinds of arrival; Chapter 4 uses two more of them. Choose **Periodic**: customers come in at a steady interval.
4. Read the settings. **Quantity per arrival** is 1: one customer each time. **Time between arrivals** already reads **10**, and **M** is already chosen. So a customer walks in every 10 minutes. The defaults are exactly what the shop needs, so there is nothing to type.
5. Click an empty part of the canvas.

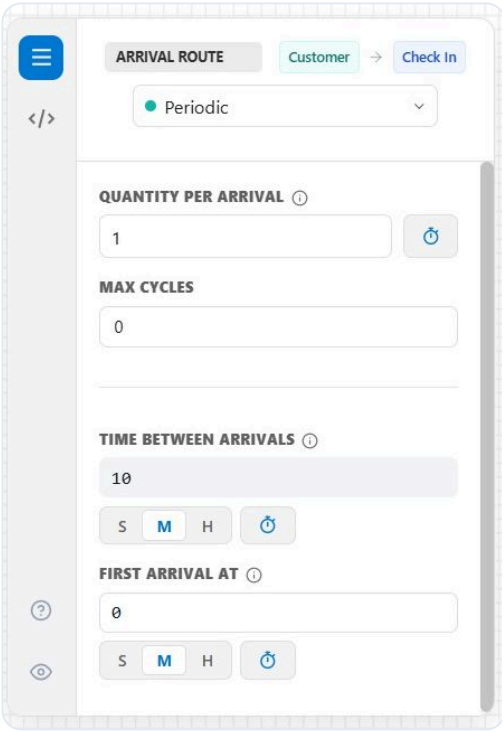



Figure 5. A Periodic arrival: one customer every 10 minutes.

Step 5.

## Draw the routes

A **route** carries a customer from one step to the next.

1. Drag from Check In's right-hand handle to **Repair**.
2. Drag from Repair's right-hand handle to **Pick Up**.

> **What you should see:** each new line carries a small label reading **100%**: every customer goes on to the next step. Chapter 5 splits a route into several paths with different shares.

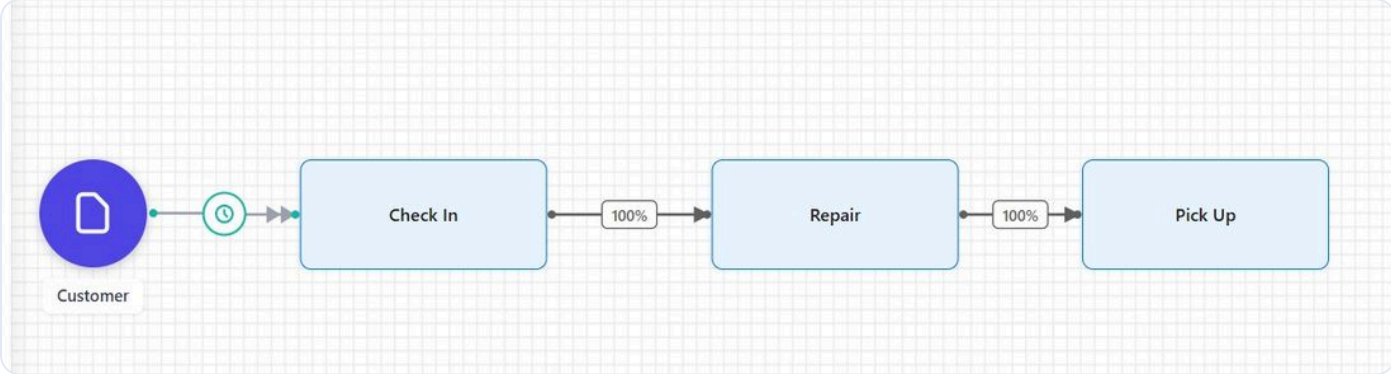



Figure 6. The arrival and two routes. Nothing is drawn after Pick Up.

Nothing is drawn after Pick Up, and nothing needs to be. A customer who finishes the last step leaves the model there and is counted as done.

Step 6.

## Add the Technician

Check In and Pick Up happen at a counter screen with no staff, so they only take time. Repair needs a person. A **resource** is whoever (or whatever) a step needs in order to work: a person, a machine, a room.

1. Drag a **Resource** onto the canvas, above Repair. Name it **Technician**. Its **Quantity** reads 1: there is one Technician. Click the empty canvas.
2. Drag from the Technician's bottom handle down to **Repair**.

> **What you should see:** a dashed line from Technician to Repair, with a small **Get** label. A dashed line means "this step needs this resource".

3. Click the dashed line and read its properties.

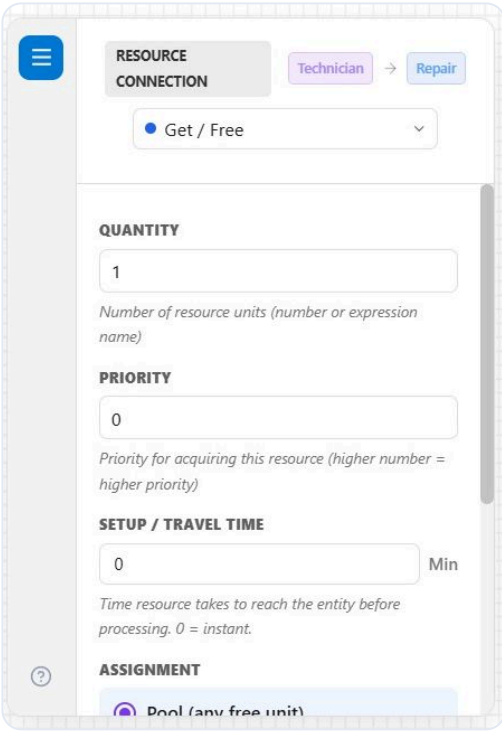



Figure 7. The Technician's connection to Repair: Get / Free, one Technician for each repair.

**Get / Free** means Repair takes the Technician for each customer and lets the Technician go when the repair is done. **Quantity** 1 means one Technician per repair. Click the empty canvas when you have read it.

The model is complete: customers arrive, go through three steps, and Repair uses the Technician.

Step 7.

## Set the run to one shop day

Before you run a model, tell it how long to run. A new model runs 24 hours; the shop is open for 8.

1. On the toolbar, click **Simulation Options**. It opens on the **Run** group.
2. **Run length** reads 24, with **Hr** chosen. Type **8**: one shop day.
3. **Replications** stays **1**, so the model runs once and you can watch it move. (In Chapter 1 you ran five; with more than one, nothing is drawn.)

4 Read the line at the foot: it names the one setting you changed, Run length. Click **Apply**.

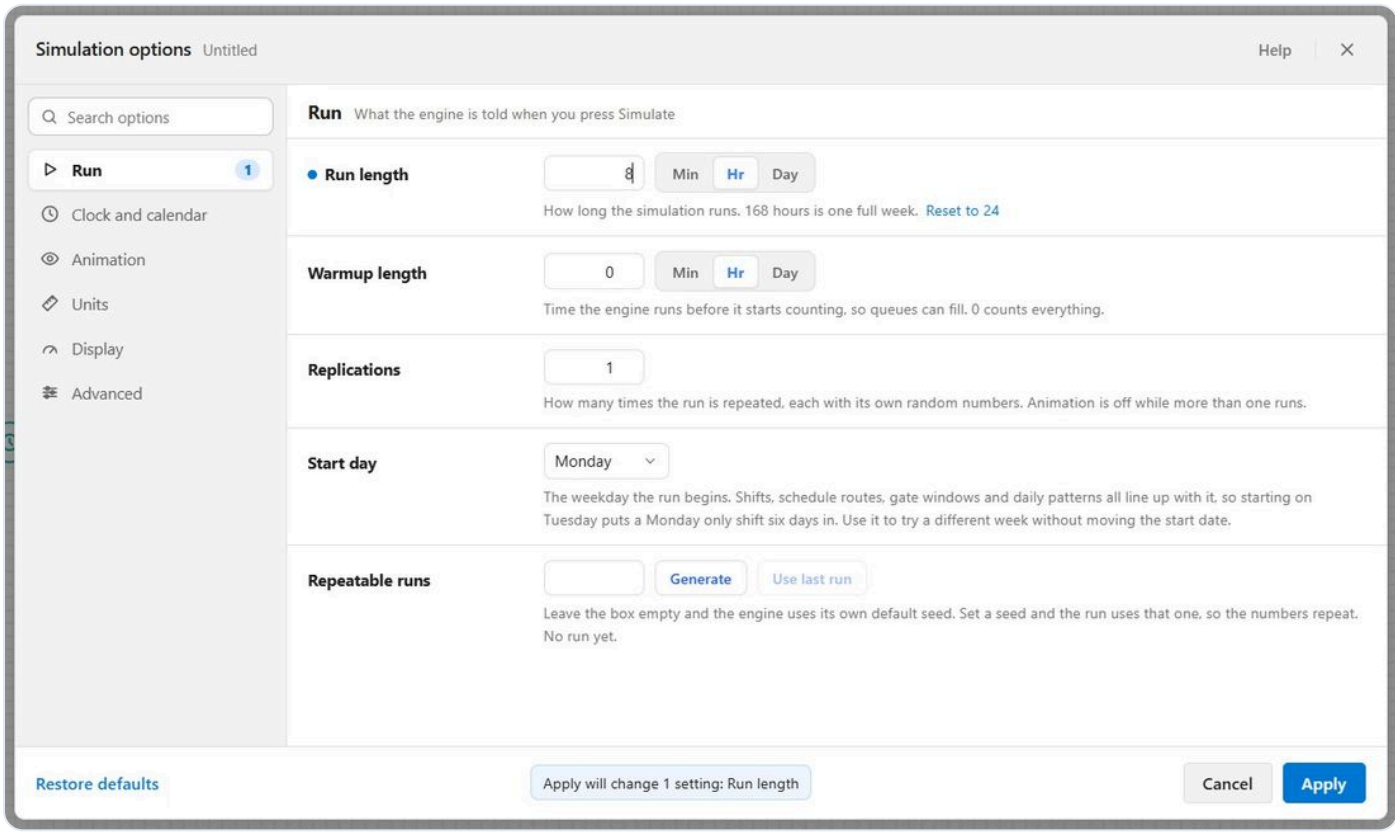



Figure 8. One shop day: Run length 8 hours, run once.

Step 8.

## Run it and watch the line build

1 On the toolbar, click **Run Simulation**.

Customers start to move from step to step. Four **live cards** appear at the top left, and they are worth learning now, because you will read them in every chapter:

- **Throughput** counts the customers who finish.
- **Avg Throughput Time** is how long a customer spends in the shop, from walking in to leaving.
- **WIP**, work in process, is how many customers are inside right now.
- **Process Efficiency** is the share of a customer's time spent being worked on, rather than waiting.

Now watch Repair. It turns red while it works. A small **blue count** on its top left corner is the number of customers waiting for the Technician.

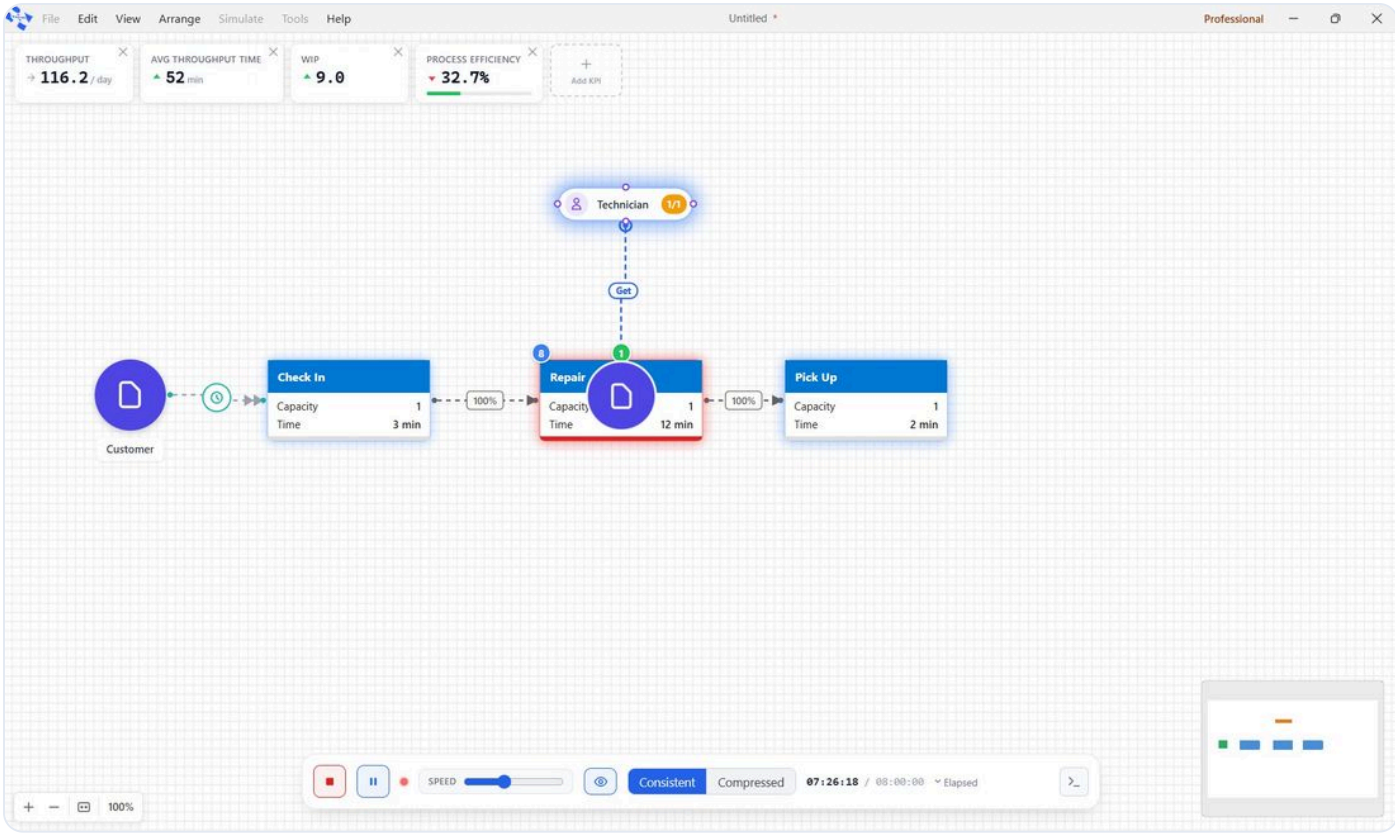



Figure 9. Late in the day: the count of customers waiting at Repair keeps climbing, and the cards climb with it.

> **What you should see:** the count on Repair's corner climbs through the day, to 8 by the end. The cards climb with it.

Step 9.

## Read the cards before you answer

When the day ends, **Simulation Complete** says how many customers went through. The cards stay on screen behind it, so read them now: they go when you answer.

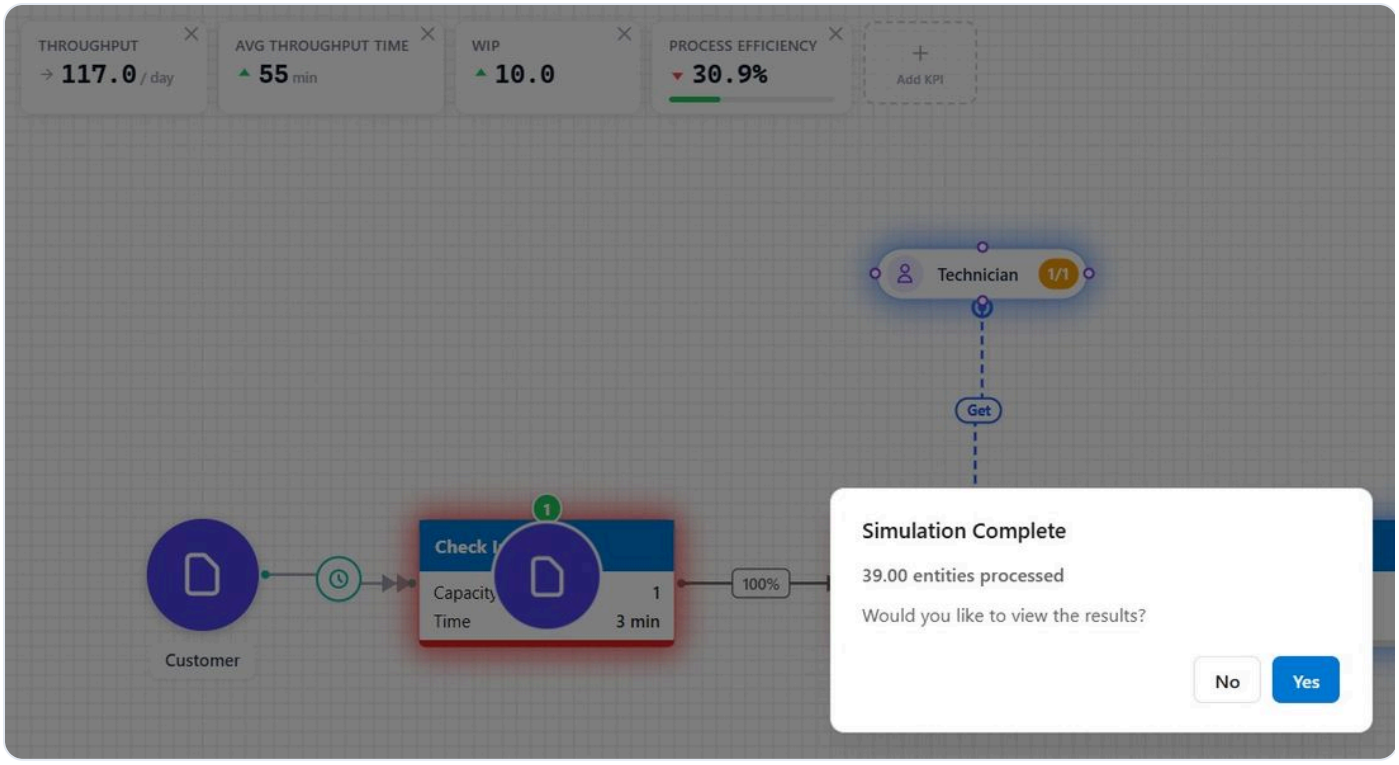



Figure 10. The first run: 39 customers, 55 minutes in the shop, 30.9% efficiency.

- **39** customers went through.
- **Avg Throughput Time, 55 minutes**, though the three steps take only 3 + 12 + 2 = 17. The other 38 minutes, on average, were spent waiting.
- **WIP, 10**: ten customers are still inside when the day ends.
- **Process Efficiency, 30.9%**: less than a third of a customer's time is spent being worked on.

Here is why, and it is the most important idea in this chapter. Repair needs 12 minutes, but a new customer comes every 10. The Technician can never catch up, so the line grows all day long. Nothing is broken in the model: it shows you exactly what that shop would do.

Click **No**. For now the cards tell you what you need; Chapters 10 and 11 teach the Output Report.

---

Step 10.

## Fix it with a second Technician

A model's real value is trying a change before you make it in the real shop. Give Repair a second Technician.

1. Double-click the person icon on the **Technician** (a double-click on its name renames it instead) and set **Quantity** to **2**. The badge on the shape now reads x2.

2. Double-click **Repair** and set **Capacity** to **2**, so two repairs can happen at the same time. Both changes are needed: two Technicians are no help if Repair can only take one customer at a time.

3. Click **Run Simulation** again.

> **What you should see:** this time no count builds on Repair's corner. Two Technicians finish two customers every 12 minutes, more than the one who walks in every 10, so each customer is taken as soon as they arrive.

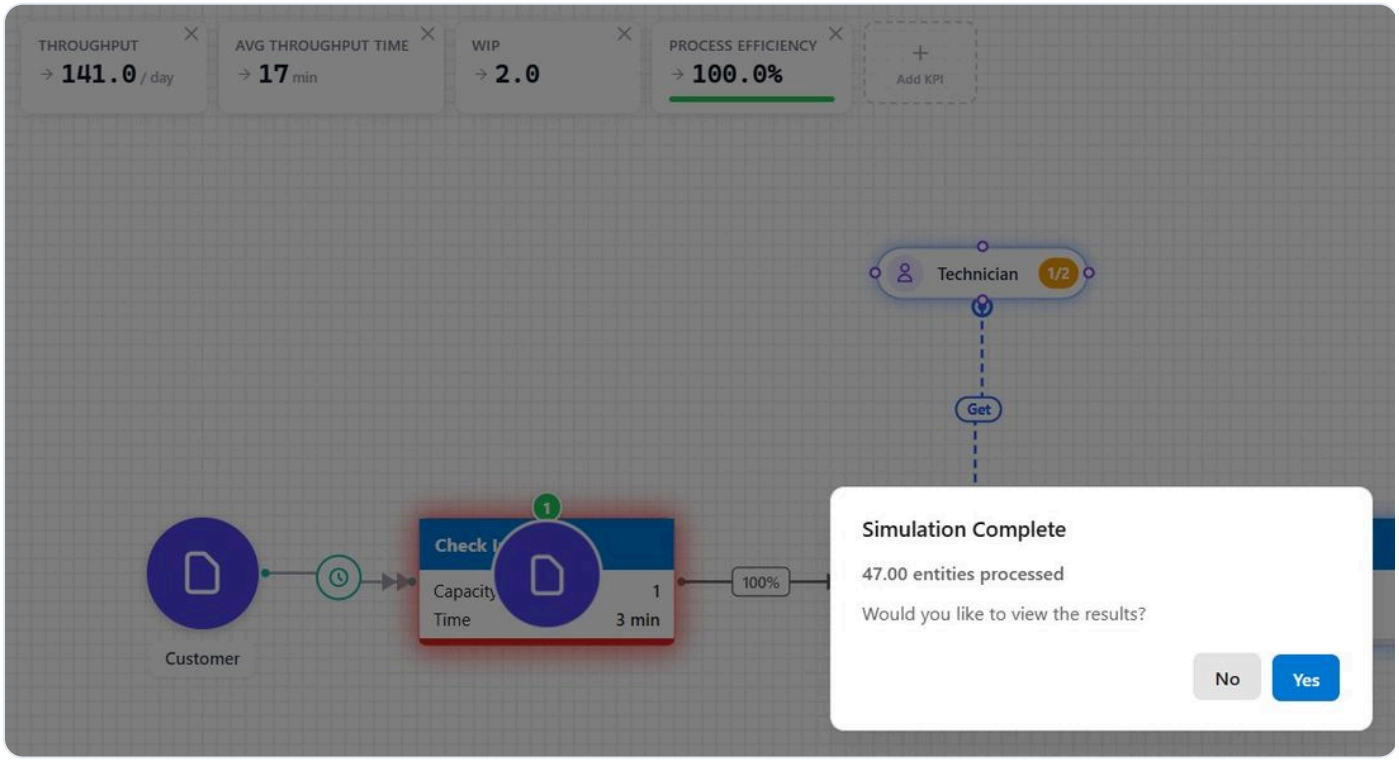



Figure 11. With two Technicians: 47 customers, 17 minutes, 100% efficiency.

Read the cards: **47** customers, **17 minutes** in the shop (just the time of the three steps), **WIP 2**, **Process Efficiency 100%**. Nobody waits. Click **No**.

---

Step 11.

## Save it

Chapter 3 starts from this model, so save it where you will find it again.

1. Click **File**, then **Save As**.

2. In **Model name**, type **Phone Repair Shop**.
3. Pick a folder you will find again, and check the path shown under **Saving to**.
4. Click **Save Model**.

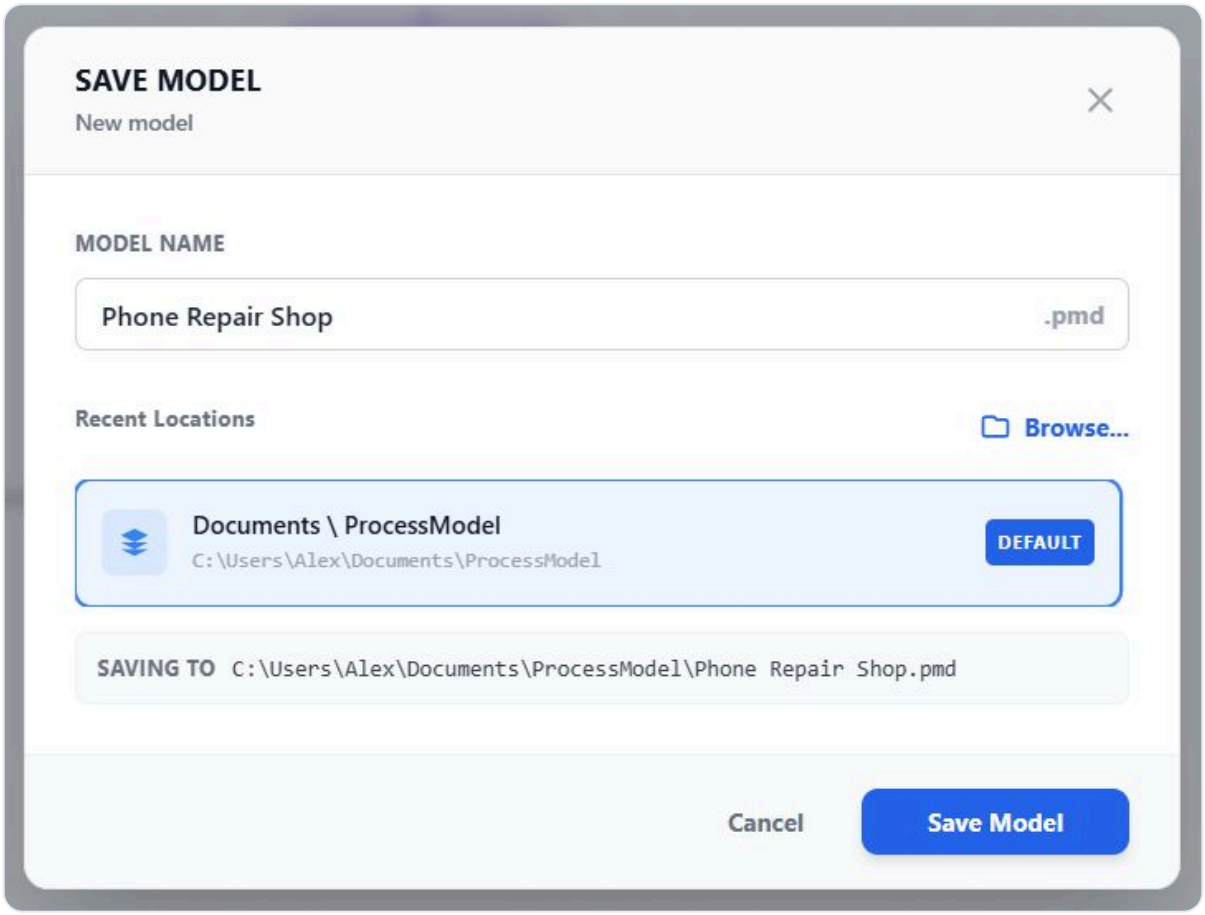



Figure 12. Save As: the shop's name, a folder you will find again, and the full path under Saving to.

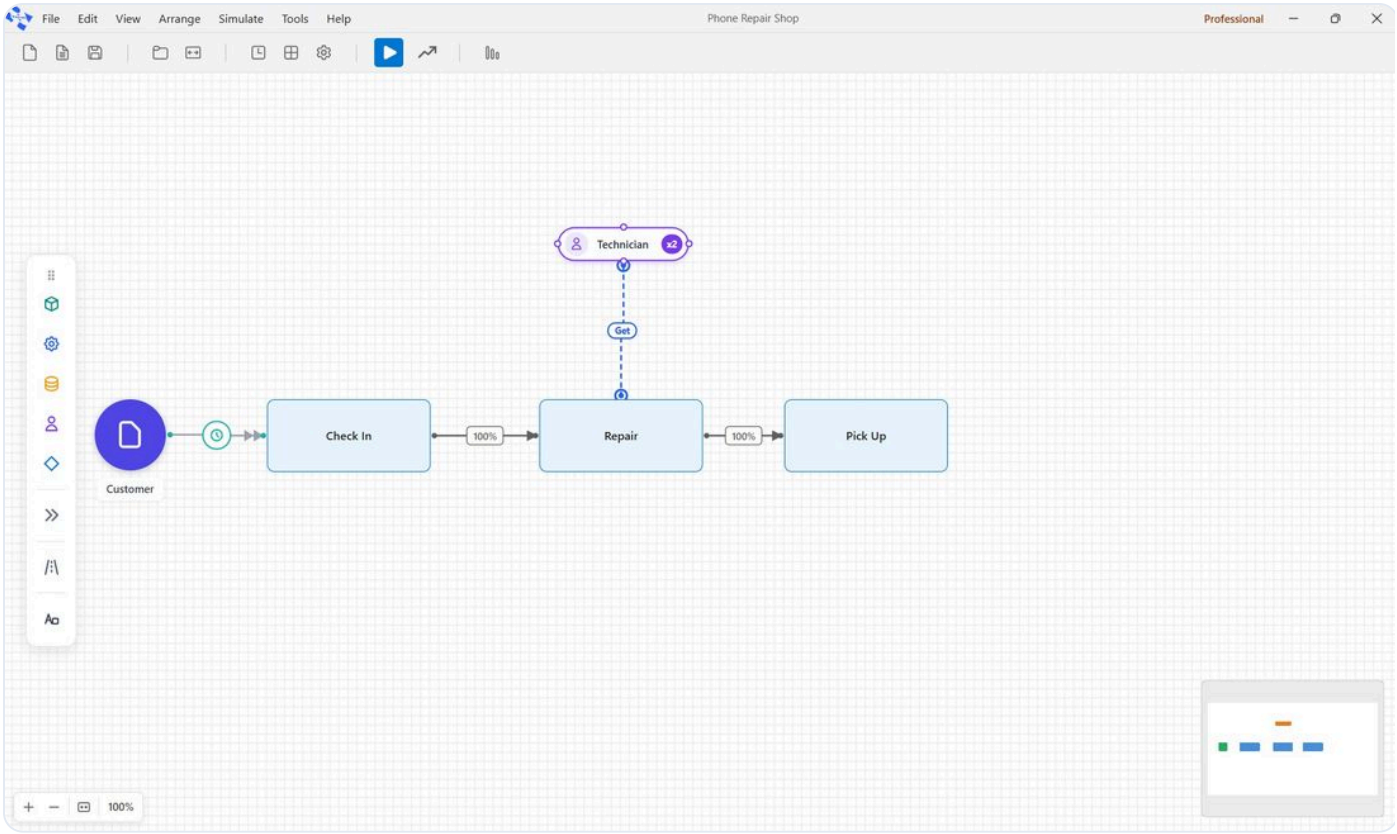



Figure 13. The finished shop, saved. The title bar shows its name.

> **What you should see:** the title bar now reads **Phone Repair Shop**.

## Your practice model

You build this one yourself, and that is the point: there is no shortcut to the feel of building. When you are done, compare your shop with the course's own, the shop as this chapter ends, with two Technicians.

START FROM

**An empty canvas**

Blank Canvas on the Welcome screen: you build the shop yourself, step by step.

COMPARE WITH

**Chapter 2. Your First Model**

The shop as this chapter leaves it. Open it when you are done, and check your settings against it.

The model to compare with is in **File** ▸ **Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**In the first run, the three steps add up to 17 minutes, yet customers spent 55 minutes in the shop on average. Where did the other time go?**

- **A** Into waiting for the Technician, because Repair takes longer than the time between customers
- B Into walking between the steps
- C Into Check In and Pick Up, which have no staff

The answer is A. A customer comes every 10 minutes but a repair takes 12, so a line forms at Repair and grows all day. The routes take no time in this model.

QUESTION 2

**You set the Technician's Quantity to 2 but leave Repair's Capacity at 1. What happens?**

- A The line clears, because there are two Technicians
- **B** The line still builds, because Repair still works on one customer at a time
- C ProcessModel will not run the model

The answer is B. Capacity limits how many customers a step works on at once. The second Technician sits idle until Repair can take a second customer.

QUESTION 3

**Why did this chapter run the day once, with Replications at 1?**

A Because one run is always enough

B So the model is drawn as it runs and you can watch the line build

C Because Simulation Options allows only one run for a new model

The answer is B. With more than one run, animation is off. With fixed times every run gives the same day anyway; once times vary, in Chapter 3 on, one run is only one possible day.

## Try it yourself

Before you close the shop, find the smallest fix on your own.

1. Set the Technician back to **1** and Repair's Capacity back to **1**. Then make Repair faster instead: set its Processing Time to **9** minutes.
2. Run it again and read the cards before you answer.

**What to expect.** With a repair of 9 minutes, shorter than the 10 between customers, the Technician keeps up: no line builds on Repair's corner, and Avg Throughput Time settles at the time of the three steps, 3 + 9 + 2 = 14 minutes. The lesson: a line forms whenever work arrives faster than a step can finish it, and it clears when the step can keep up, whether by more people or by faster work. Set Repair back to 12 minutes, two Technicians and Capacity 2, and save, so your shop matches the one Chapter 3 starts from.

CHAPTER 3 OF 15

# Chapter 3. Make Your Times Vary

By the end of this chapter you can

**Make work times vary, from your own data or by hand.**

Watch the video (5 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 3. Make Your Times Vary**
www.youtube.com/watch?v=VkDuilCqvNE

## Before you start

In Chapter 2 every time in the shop was fixed: a customer exactly every 10 minutes, a repair exactly 12. Real work is never like that. One repair is a quick screen swap, the next a stubborn fault; customers come in clumps and gaps. This chapter teaches you to give the model times that **vary**, the way real times do.

Why does it matter? Because averages hide waiting. With fixed times, two Technicians kept up with no line at all. With times that vary, the same shop, with the same averages, sometimes gets two long repairs back to back while customers keep coming, and then people wait. A model with fixed times would tell you the shop is fine when it is not.

A time that varies is written as a **distribution**: a rule that says which times are likely and which are rare. You will set one three ways, the three you will use in real projects:

| Part | Chapter 2 | Chapter 3 |
|---|---|---|
| Repair | 12 minutes | **Fitted to data**: 40 repair times the shop recorded, about 12 minutes on average |
| Check In | 3 minutes | **Built by hand**: at least 2, mostly 3, at most 5 minutes |
| Customers | One every 10 minutes exactly | **At random**, 10 minutes apart on average |
| Pick Up and staff | 2 minutes, two Technicians, Capacity 2 | Unchanged |

**You need** your Phone Repair Shop from Chapter 2 (or the practice model below) and about thirty minutes.

Step 1.

## Open your shop

1. Start ProcessModel. The Welcome screen lists your shop under **Recent Models**.
2. Click **Phone Repair Shop** to open it.

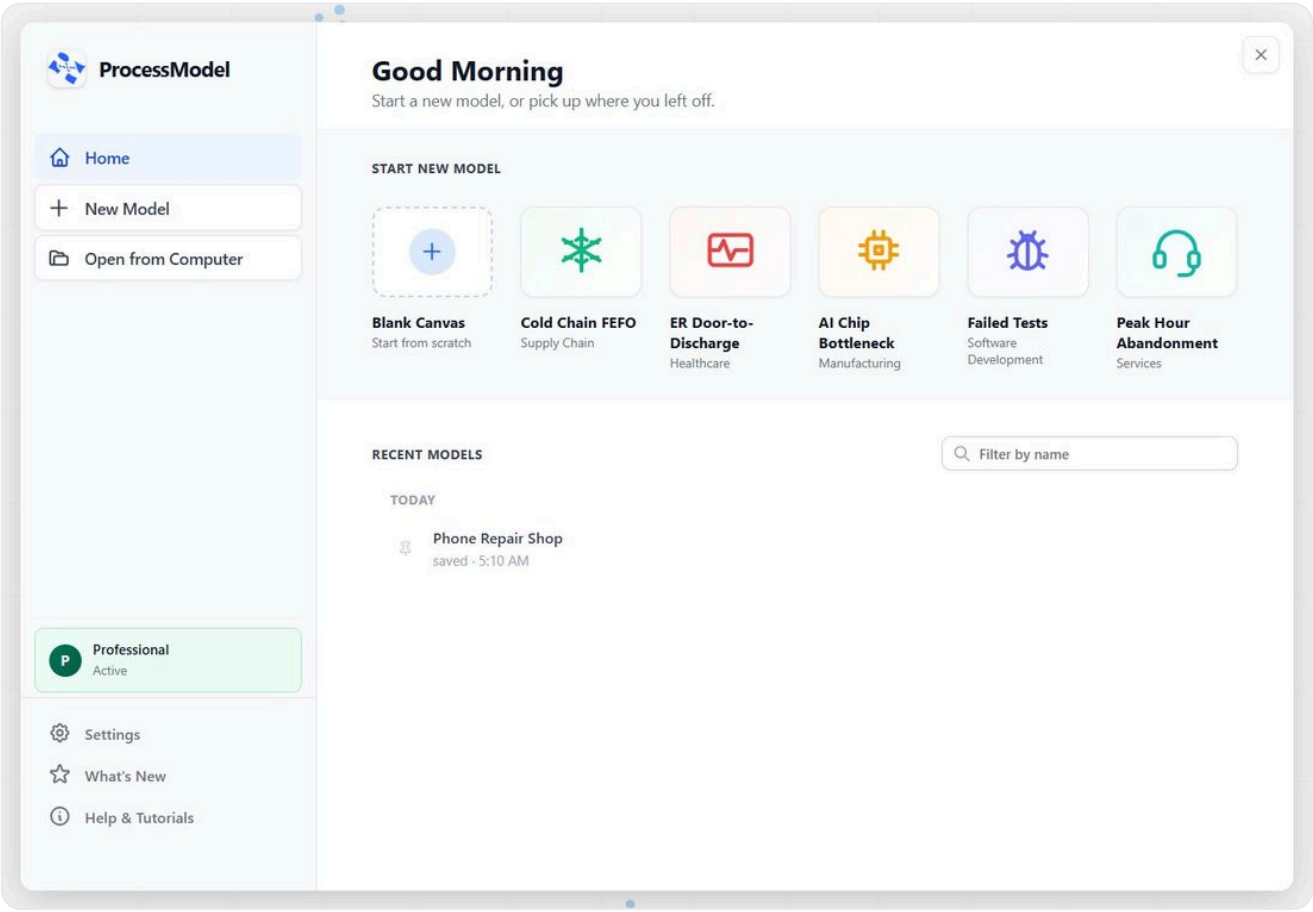



Figure 1. Your saved shop waits under Recent Models.

**What you should see:** the shop just as you left it, with two Technicians and Repair's Capacity at 2.

Step 2.

## Fit Repair’s time to real data

When you have records of how long a job really took, let the data choose the distribution. That is the most honest way, because it describes your work, not a guess about it.

The shop recorded 40 repair times, in minutes. They are made up for this course, not real customer data, but they behave like real records: most repairs take about 10 to 13 minutes, a few are quick, and two run close to 25.

▶ **The 40 repair times, one per line, to copy**

1. Click **Repair**. Its **Processing Time** reads 12.
2. Click the button beside the Processing Time box. **Time Expression** opens, on its **Fit Data to Distribution** tab.
3. Click in **Sample Data** and paste (or type) the 40 times, one per line. **Count**, under Data Summary, reads 40.
4. Click **Analyze**.

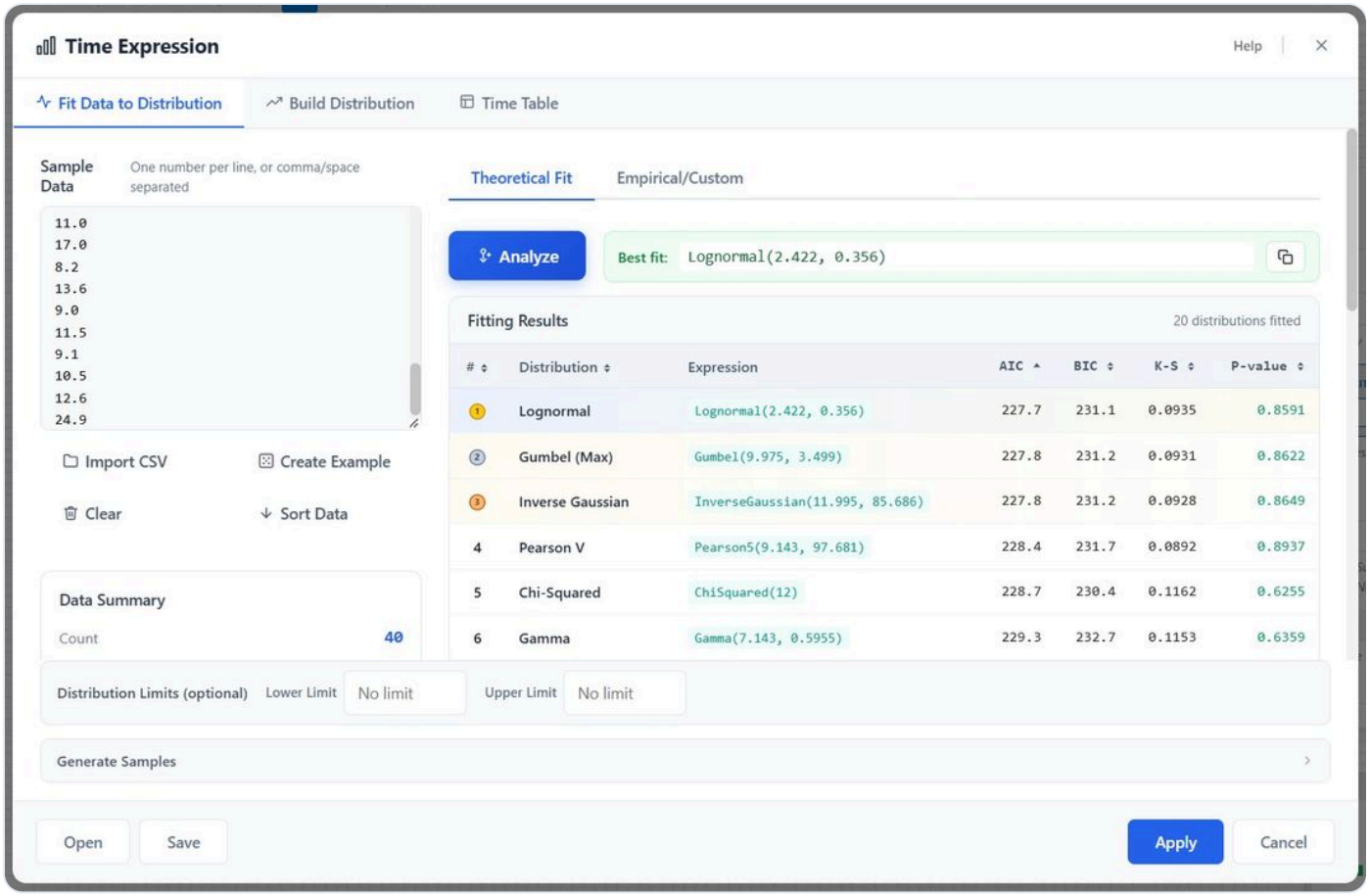



Figure 2. Twenty distributions fitted and ranked, best first. The best fit is shown above the list.

**What you should see:** twenty distributions fitted to your data and ranked, best first. The best, **Lognormal(2.422, 0.356)**, is shown above the list.

Two things are worth understanding here. First, the top three (Lognormal, Gumbel and Inverse Gaussian) score almost the same, so any of them describes these repairs well; there is no need to agonize over the winner. Second, Lognormal's two numbers are on a log scale, not in minutes, so do not read 2.422 as a time. What matters is the shape: most repairs near the middle, with a long tail of slow ones.

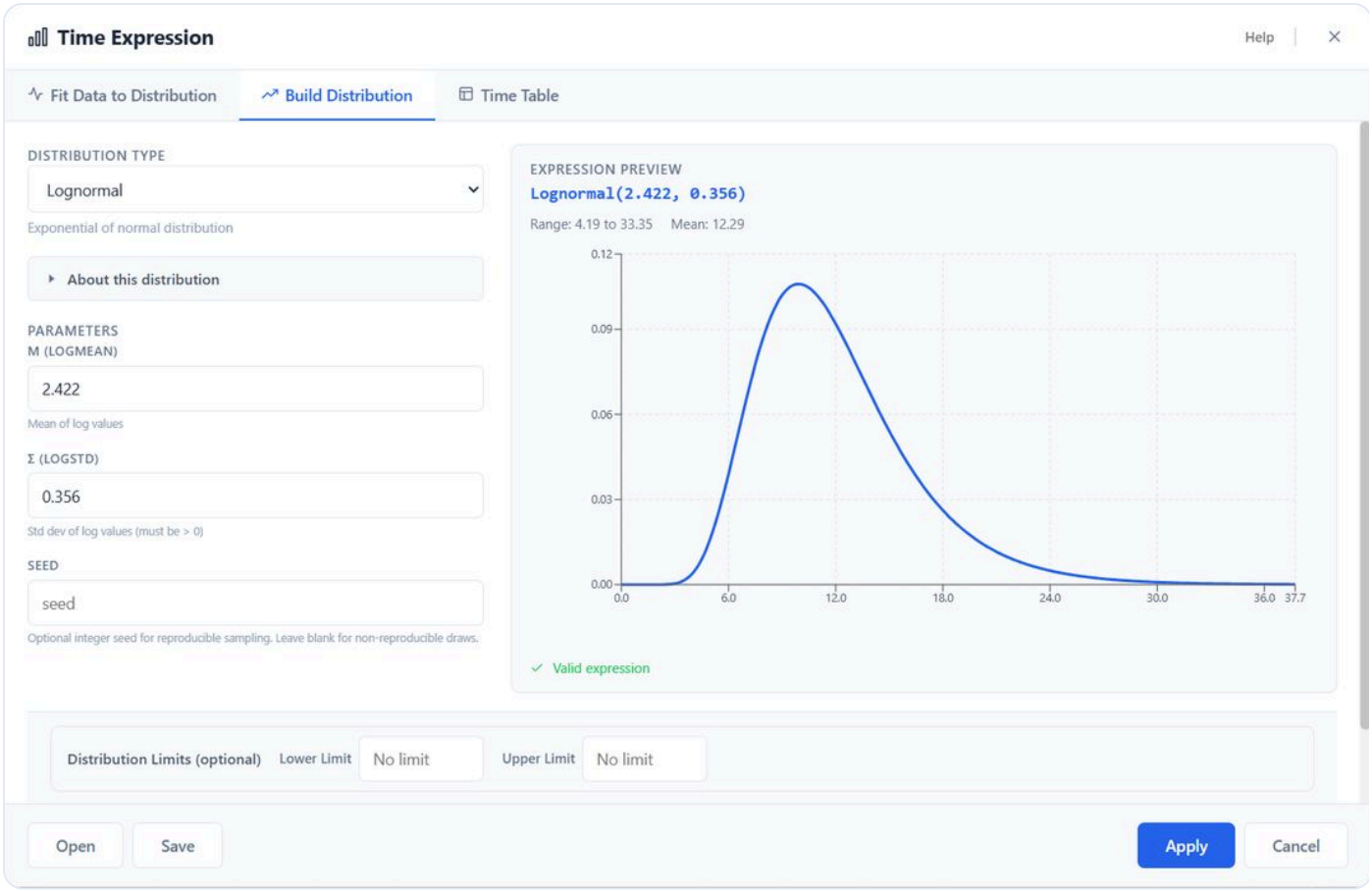



Figure 3. The fitted shape, on the Build Distribution tab: most repairs near the middle, a long tail of slow ones, a mean of about 12 minutes.

5 Click **Apply**. A question asks whether to save the data with the model. Click **No**: the fitted distribution is all the model needs.

> **What you should see:** Processing Time now reads **Lognormal(2.422, 0.356)**, and **M** is still chosen. The numbers in a distribution take the unit of the field, so the times it draws are in minutes.

---

Step 3.

## Build Check In's time by hand

Often you have no records, but the people who do the work know it well. Ask them three questions: what is the least it ever takes, what does it usually take, and what is the most? Those three answers make a **triangular** distribution.

For Check In, the staff say: at least 2 minutes, mostly 3, at most 5.

1. Click **Check In**. Its Processing Time reads 3.
2. Open Time Expression with the button beside the box, and click the **Build Distribution** tab.
3. **Distribution type** is already **Triangular**, and **A (min)**, its least value, is already 2.
4. Type **3** in **M (mode)**, the most likely time, and **5** in **B (max)**, the most.

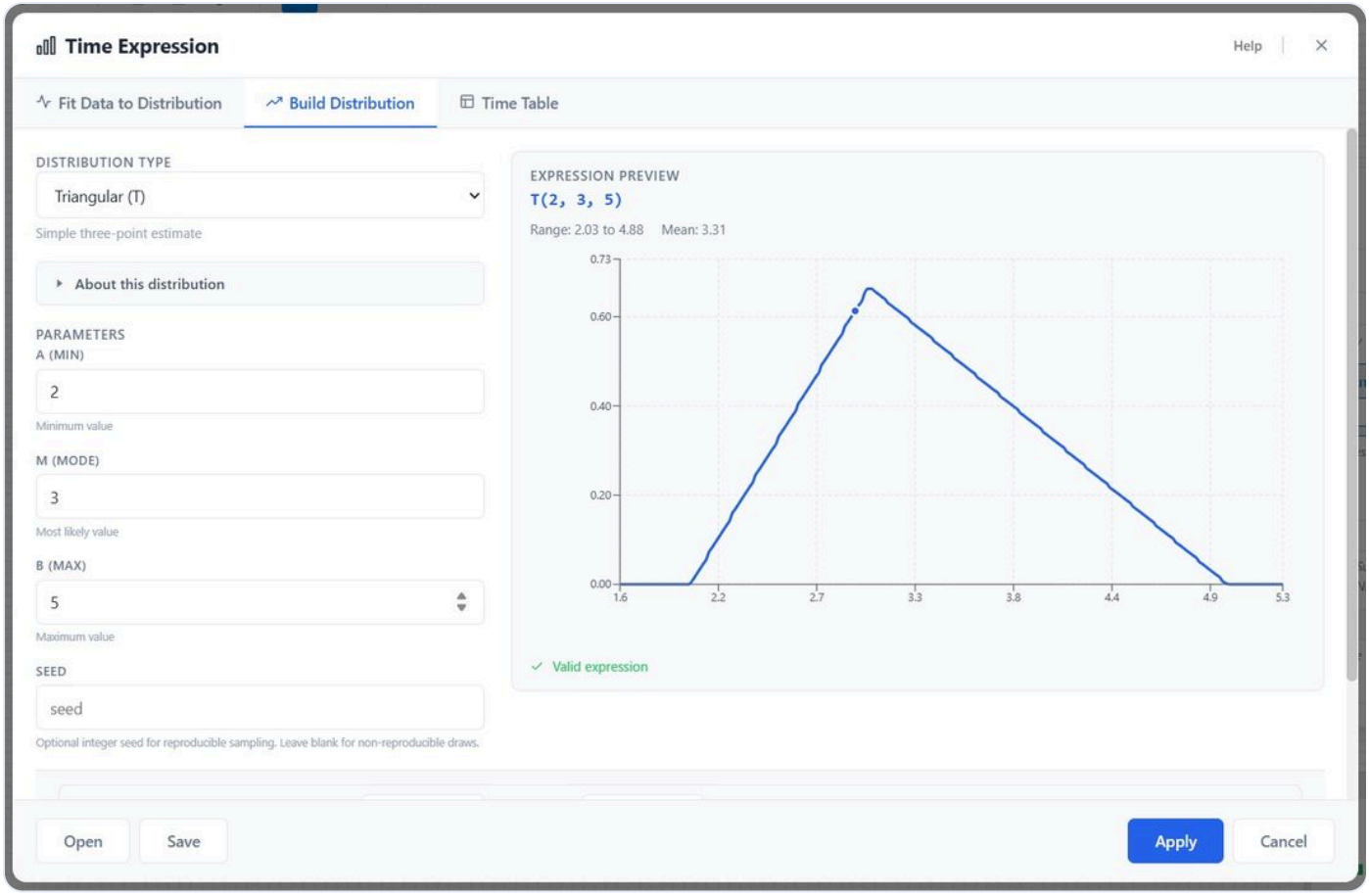



Figure 4. A triangular distribution from three answers: least 2, most likely 3, most 5.

**What you should see:** the preview writes it as **T(2, 3, 5)**, T for triangular, and draws its shape: most check-ins take close to 3 minutes, none less than 2 or more than 5.

5. Click **Apply**. Processing Time now holds T(2, 3, 5), still in minutes.

Step 4.

## Make customers arrive at random

Customers do not walk in on a timer. They come at random: one every 10 minutes **on average**, but sometimes two in a minute and sometimes nobody for half an hour. Short gaps are common and long gaps rare. The **exponential** distribution is made for gaps like these, and it needs only one number, the average gap.

1. Click the arrival line, from Customer to Check In. **Time between arrivals** reads 10.
2. Its Time Expression button sits under the box, on the unit row. Click it, then click **Build Distribution**.
3. In **Distribution type**, choose **Exponential**.
4. Type **10** in **M (mean)**, the average gap in minutes.
5. Click **Apply**.

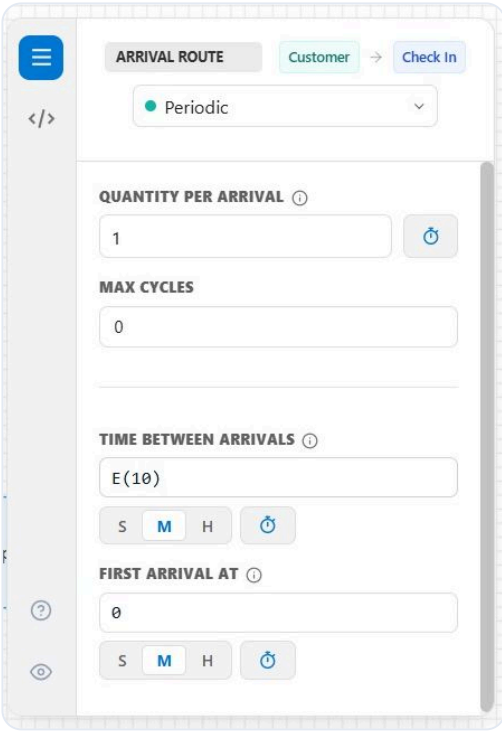



Figure 5. Time between arrivals now reads E(10): random gaps, 10 minutes on average.

> **What you should see:** Time between arrivals reads **E(10)**, E for exponential, with a mean of 10, still in minutes.

Look at what you have done: you changed **no average**. Customers still come every 10 minutes on average, Check In still averages about 3, Repair about 12. Only the variation is new. Now see what it does.

Step 5.

## Run one day and watch

The run settings are the ones you set in Chapter 2: 8 hours, run once.

1. Click **Run Simulation**.
2. Watch the top left corner of **Repair**. When both Technicians are busy and another customer is ready, that customer waits, and the blue count shows there.

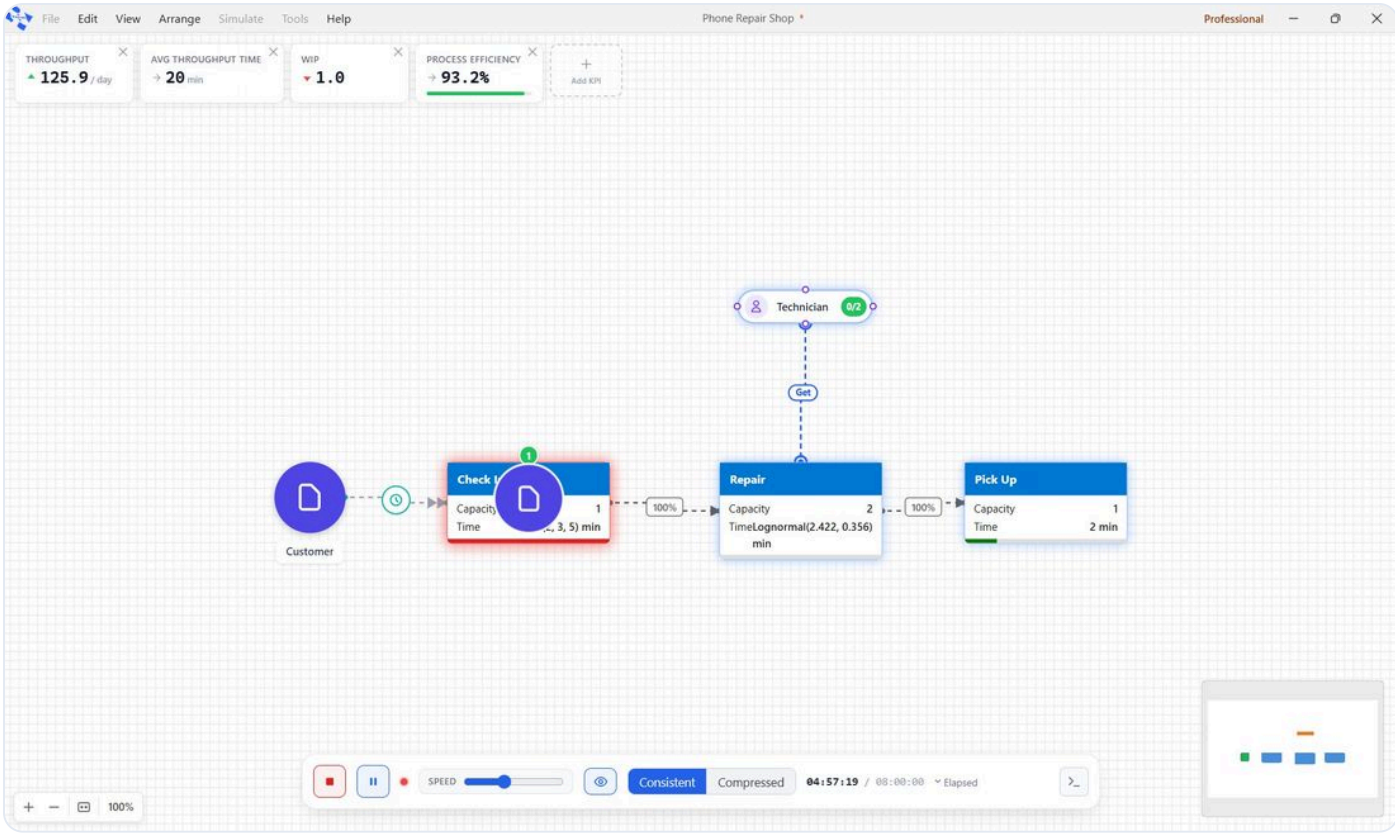



Figure 6. One day with times that vary. Each shape now shows its time as a distribution.

> **What you should see:** the count on Repair's corner comes and goes, one or two at a time, and clears again. In Chapter 2, with fixed times, it never appeared at all.

Step 6.

## Read the cards

When the day ends, read the cards before you answer **Simulation Complete**.

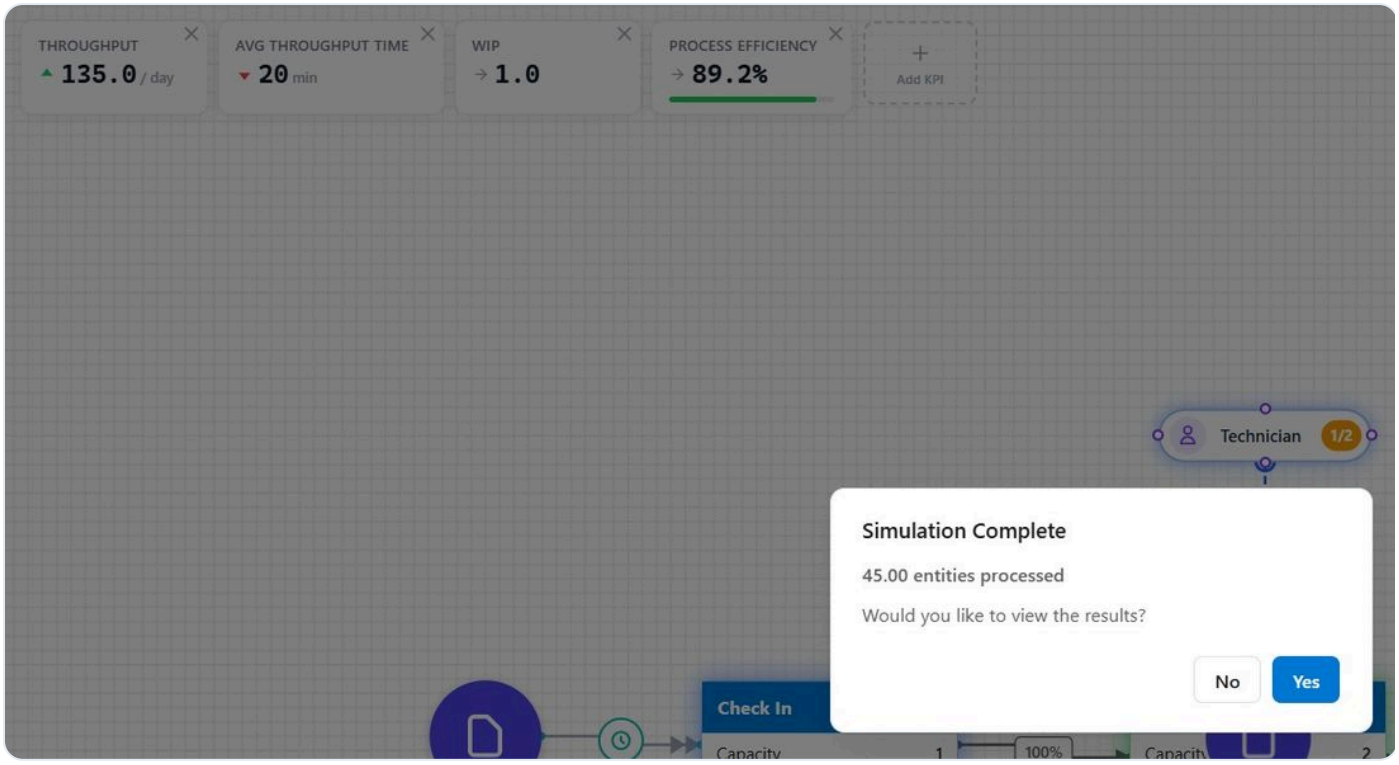



Figure 7. The day with times that vary: 45 customers, 20 minutes in the shop, 89.2% efficiency.

- **45** customers went through. Throughput reads **135 per day**.
- **Avg Throughput Time, 20 minutes**, up from 17 in Chapter 2. Customers now wait at times.
- **WIP, 1**, where Chapter 2 ended on 2. WIP counts who is inside at this moment, so it moves all day; at the end of a run it tells you little.
- **Process Efficiency, 89.2%**, down from 100%. The lost share is time spent waiting.

The shop, its staff and every average are the same as in Chapter 2. Variation alone added the wait. That is why real models need real variation.

One caution: this is **one run**, so it is **one possible day**. A different day, with different random customers and repairs, would give somewhat different numbers. Chapter 10 runs many days and reads them together. Click **No**.

---

Step 7.

## Save it

1. Click **File**, then **Save**.

**What you should see:** a short note that the model was saved, and the star (*) that marked unsaved changes leaves the title bar. The shop keeps its name, Phone Repair Shop. Chapter 4 starts from it.

## Your practice model

START FROM

**Chapter 2. Your First Model**

The shop as Chapter 2 left it. Use your own saved Phone Repair Shop instead if you built it.

COMPARE WITH

**Chapter 3. Make Your Times Vary**

The shop as this chapter leaves it. Open it when you are done, and check your settings against it.

Both are in **File** ▸ **Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**You changed no average in this chapter, yet Avg Throughput Time went from 17 to 20 minutes. Why?**

A. The fitted repair time is longer than 12 minutes

B. With times that vary, long repairs and close arrivals sometimes bunch up, and customers wait

C. Check In now takes 5 minutes every time

The answer is B. The averages are about the same, but variation makes busy moments when work arrives faster than the Technicians can finish it. Those moments make the wait.

QUESTION 2

**A clinic has no records of how long a check-up takes, but the nurses say: never under 10 minutes, usually 15, at most 30. What do you use?**

A Fit Data to Distribution, with 15 typed in

B A Triangular distribution built by hand: T(10, 15, 30)

C An Exponential distribution with a mean of 15

The answer is B. Least, most likely and most are exactly what a triangular distribution takes. Fit Data needs real records, and Exponential is for random gaps between arrivals.

QUESTION 3

**The best fit reads Lognormal(2.422, 0.356). What do its two numbers mean?**

A The shortest and the longest repair, in minutes

B They are on a log scale, not in minutes: they set the distribution's shape

C The repair takes between 2.422 and 0.356 hours

The answer is B. Lognormal's parameters describe the logarithm of the time. The times it draws are still in the field's unit, minutes, and average about 12.

## Try it yourself

The top three fits scored almost the same. Does it matter which one you pick? Find out.

1. Click **Repair**, open Time Expression from its Processing Time, and paste the 40 times into **Sample Data** again. Click **Analyze**.
2. Click the second row, **Gumbel (Max)**, to choose it instead of the best fit, and click **Apply** (answer **No** to saving the data).
3. Run the day again and read the cards before you answer Simulation Complete.

> **What to expect.** Avg Throughput Time stays close to this chapter's 20 minutes, but not every card does. In one run of the practice model, Avg Throughput Time went from 20 to 23 minutes, while the count went from 45 customers to 63 and Throughput from 135 to 189 per day. One run is one possible day, so no single card settles it: read all four before you decide a change made no difference. Put Repair back to **Lognormal(2.422, 0.356)** the same way, or simply close without saving, before Chapter 4.

CHAPTER 4 OF 15

# Chapter 4. Make Customers Come When They Really Do

By the end of this chapter you can

**Make customers arrive the way they really do, rushes and schedules included.**

Watch the video (6 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 4. Make Customers Come When They Really Do**
www.youtube.com/watch?v=aLlBaCYPcmY

## Before you start

In Chapter 3, customers came at random, but evenly: as many at 3 in the morning as at noon. No real shop works like that. Customers come when their day allows: few at opening, a rush at lunch, a quiet evening. And some come in groups, at times you know in advance.

**When** work arrives matters as much as **how much** arrives. A shop that handles 48 customers a day with ease can still have a long line at lunch, if a third of them come in the same hour. Averages over the day hide that; a model with the right arrival pattern shows it. This chapter teaches the two arrival types that shape the day:

- A **Daily Pattern** says how many customers come in a day and how they spread over its hours.
- A **Scheduled** arrival brings a set number at exact times, like an appointment.

Here is what changes in the shop:

| Part | Chapter 3 | Chapter 4 |
|---|---|---|
| Walk-in customers | At random, 10 minutes apart, day and night | A daily pattern: 48 on Monday, from 7 in the morning to 8 at night, busiest at noon |
| An office group | None | Six customers from one office, together, at 10 every Monday morning |

| Part | Chapter 3 | Chapter 4 |
|---|---|---|
| Run length | 8 hours (midnight to 8 in the morning) | 24 hours, one whole Monday |
| Steps and staff | As Chapter 3 left them | Unchanged |

**You need** your Phone Repair Shop from Chapter 3 (or the practice model below) and about thirty minutes.

---

Step 1.

## Open your shop

1 On the Welcome screen, under **Recent Models**, click **Phone Repair Shop**.

**What you should see:** the shop as Chapter 3 left it, with its arrival reading E(10).

Step 2.

# Change the walk-ins to a daily pattern

1. Click the arrival line from **Customer** to **Check In**. Its type is **Periodic**.
2. Open the **Type** list at the top of the panel. It lists six ways for customers to arrive.

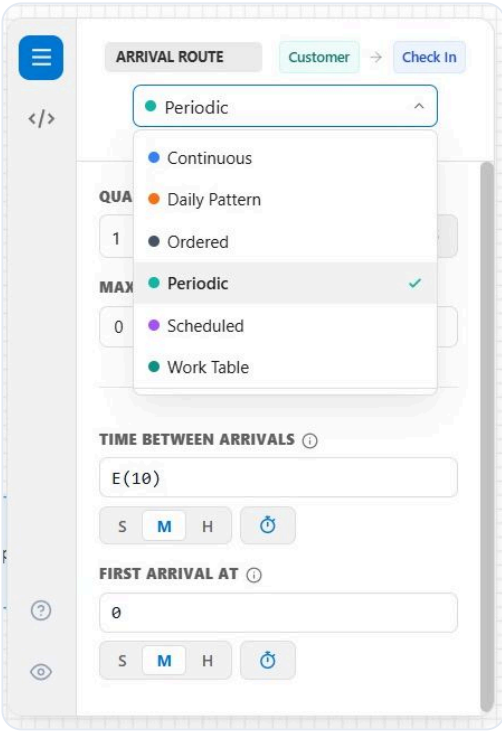



Figure 1. The six arrival types. Periodic is the one you have used so far.

3. Choose **Daily Pattern**. The panel reads **No days configured**: a pattern needs at least one day.
4. Click **Open Pattern Builder**. The builder starts with no days.
5. Click **Add Day** and choose **Monday**.
6. In **Day Quantity**, type **48**: 48 walk-ins come on a Monday.
7. Click the **Business hours** preset.

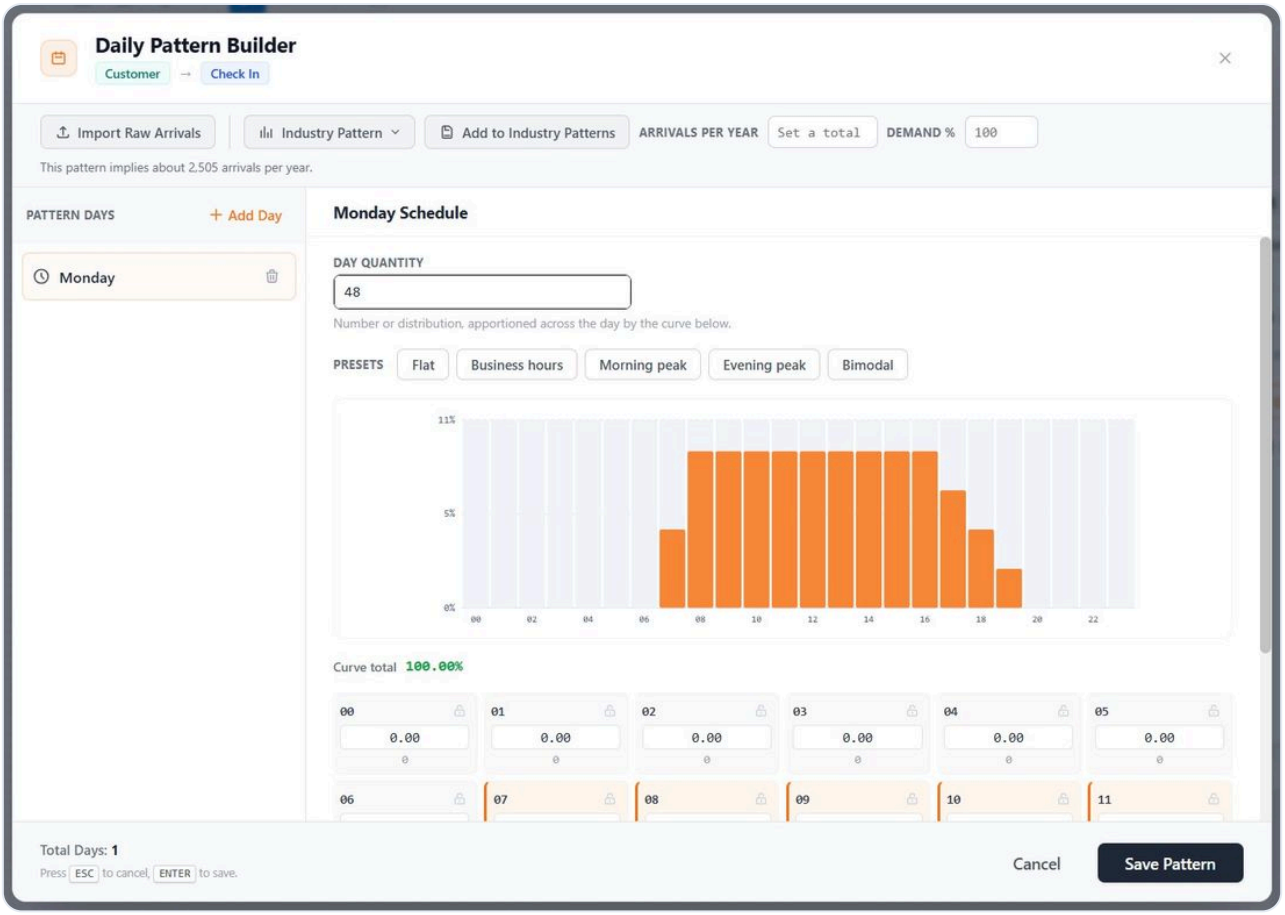



Figure 2. Monday with the Business hours preset: customers spread over the working day.

> **What you should see:** a bar chart of the day's 24 hours. Each bar is the share of the day's customers who come in that hour. With Business hours, customers come from 7 in the morning to 8 at night, and the shares add up to 100%.

Now give the day its lunchtime rush.

8. Click the box for **12**, the hour from noon to 1, and type **30**: 30 percent of the day's customers.
9. Click outside the box to set it.

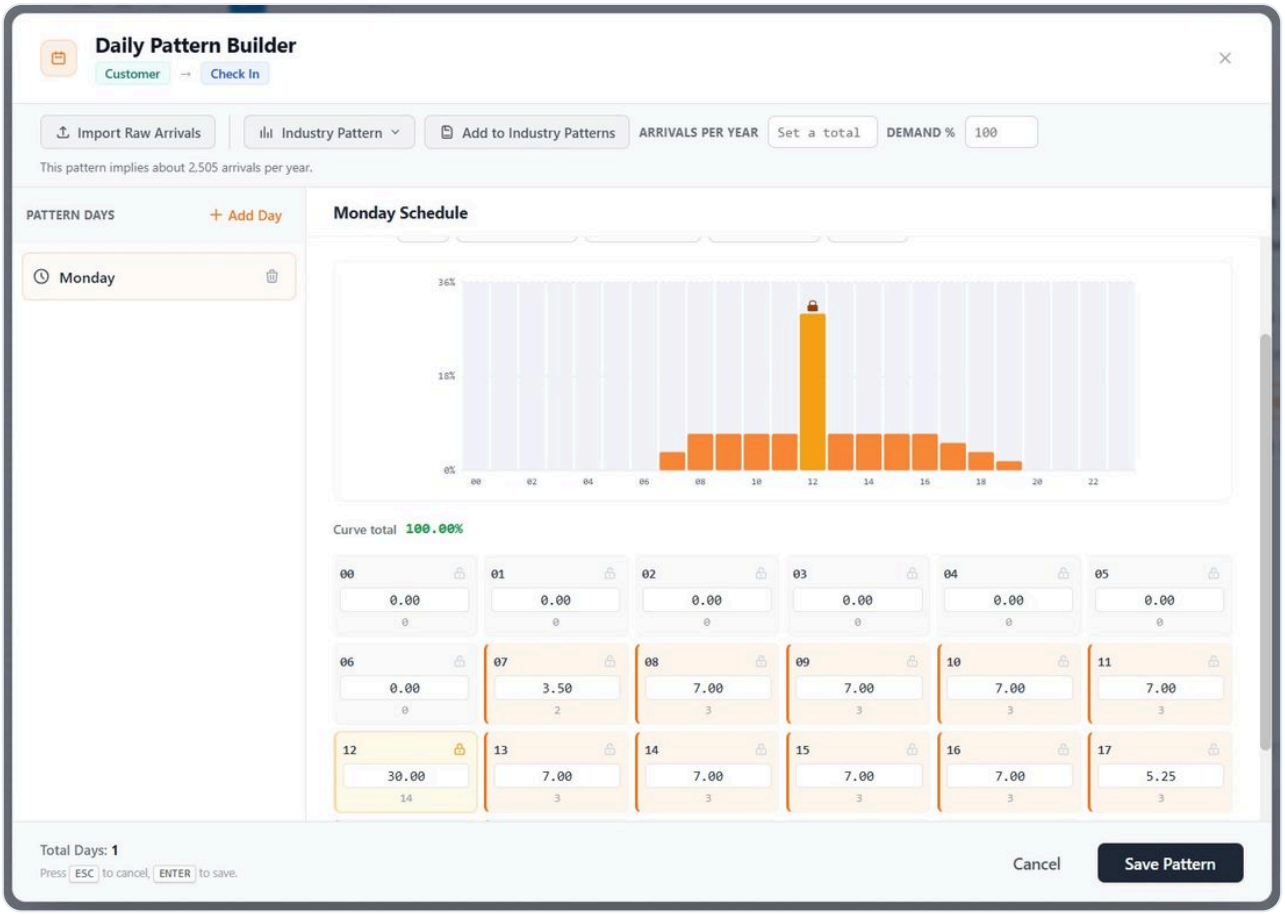



Figure 3. Noon set to 30% and locked; the other hours share the rest, so the curve still totals 100%.

> **What you should see:** the noon bar towers over the rest. A lock marks noon, and the other hours shrink to share the remaining 70%, so the curve still totals **100%**. That is 30% of 48, about **14 customers between noon and 1**, against about 3 in each other busy hour.

10 Click **Save Pattern**. The panel reads **1 day configured**.

A pattern is a shape, not a timetable. Each Monday, the day's 48 customers are shared out over the hours by the curve, and each hour's customers arrive at random moments inside that hour.

---

Step 3.

## Add the office group on a schedule

A business client sends six of its staff, each with a phone, every Monday at 10 in the morning. You know exactly when they come and how many, so they are a **Scheduled** arrival.

1. Draw a second line from **Customer** to **Check In**, just under the first.
2. Click it. It starts as **Periodic** too. Open the **Type** list and choose **Scheduled**, for arrivals at exact times.
3. Click **Open Schedule Builder**. It is empty. Click **Add Row**.
4. **Week** 1 and **Day** Monday are already set. Click the hour of **Time** and type **10**, for 10 in the morning.
5. In **Quantity**, type **6**: six customers in one arrival, a group.

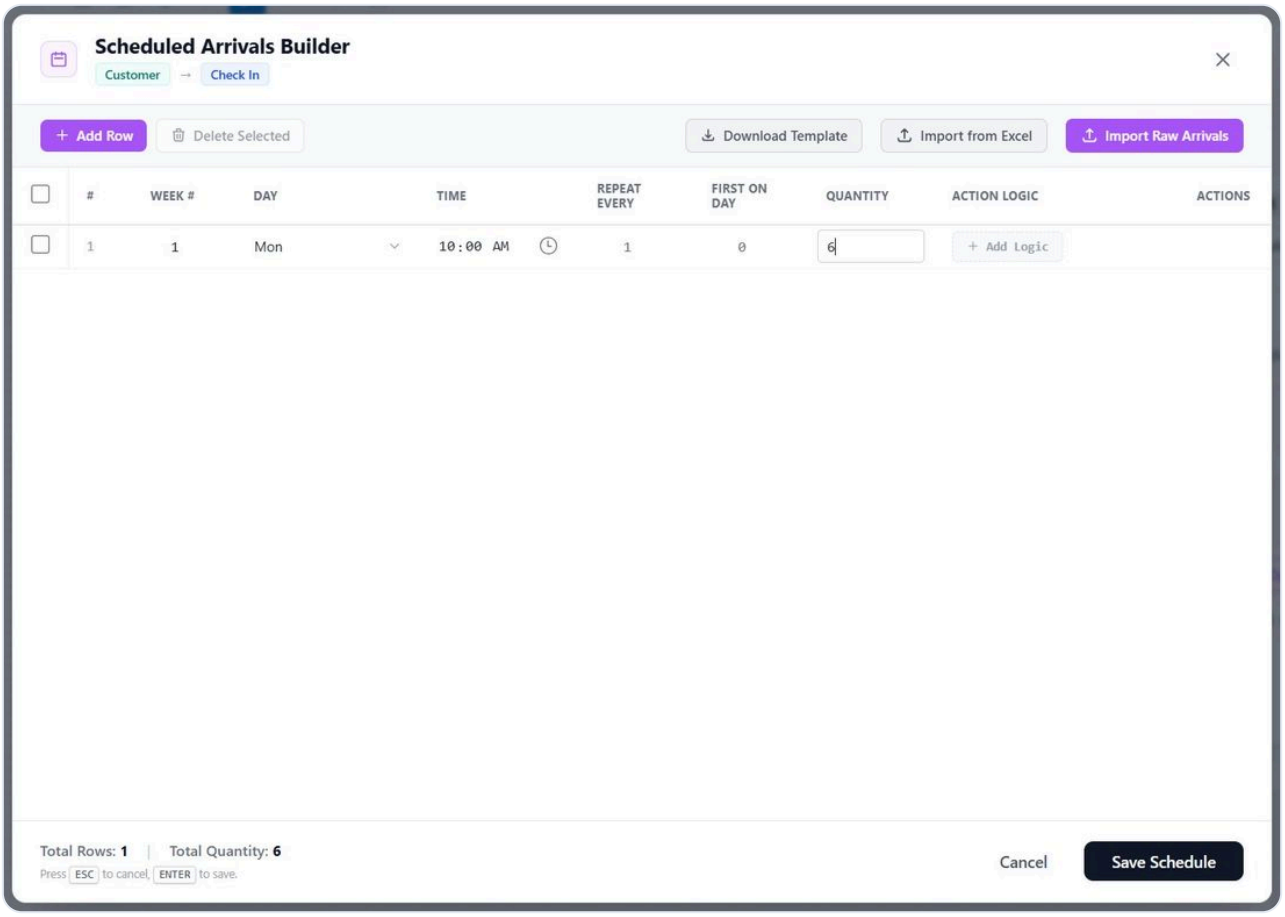



Figure 4. One row: six customers at 10:00 AM on Monday of week 1.

6. Check the foot: **Total Quantity 6**. Click **Save Schedule**.

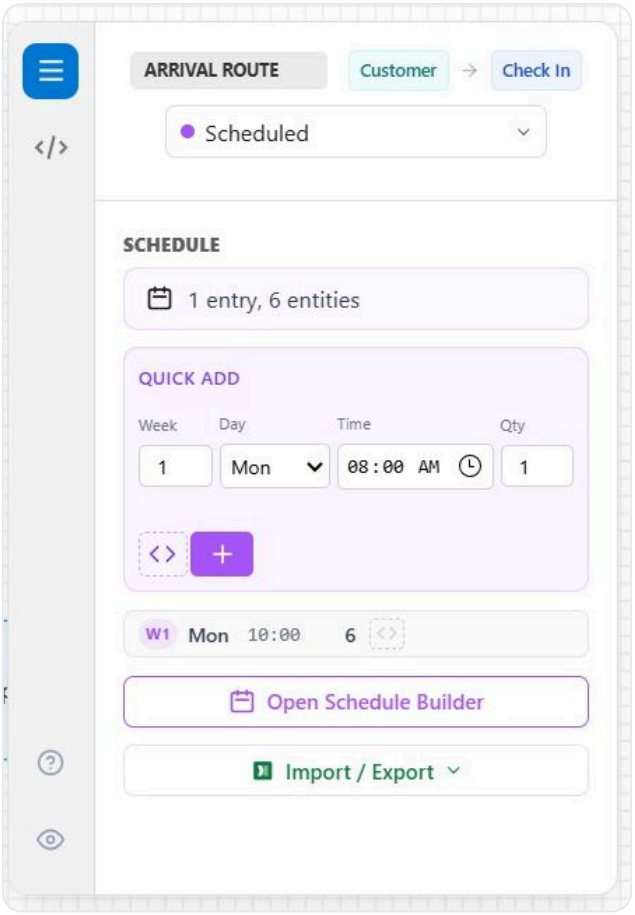



Figure 5. The scheduled arrival: one entry, six customers.

**What you should see:** the panel reads **1 entry, 6 entities**. Each arrival line now carries its own badge, so you can tell the pattern from the schedule at a glance.

Step 4.

## Run a whole day

Every run starts at **midnight on Monday**, so the pattern's hours sit on the run's clock. Your run is still 8 hours long, so it would end at 8 in the morning, before the rush. Run the whole day instead.

1. Click **Simulation Options**.
2. Type **24** in **Run length**; **Hr** is already chosen.
3. Read the foot (it names one change, Run length) and click **Apply**.

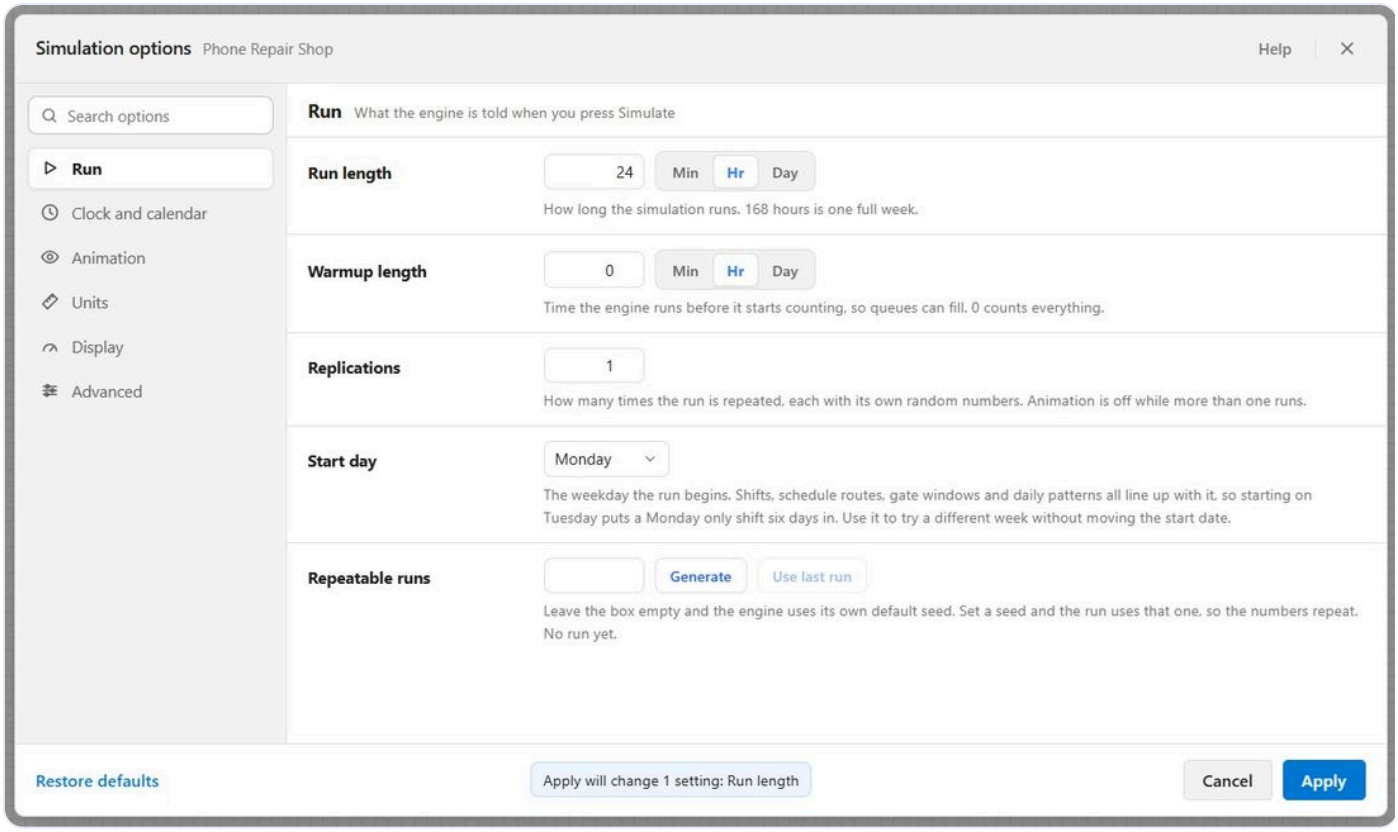



Figure 6. Run length 24 hours: one whole Monday, from midnight to midnight.

Step 5.

## Watch where the lines form

1. Click **Run Simulation**, and watch the clock in the run bar and the counts on the corners of **Check In** and **Repair**.

Here is what happens, hour by hour. Nobody comes before 7, so the night passes with nothing to see. From 7, walk-ins start to arrive, a few each hour.

At **10**, the six office customers arrive together. Check In takes one at a time, so five wait there. As they finish, the line moves on to Repair.

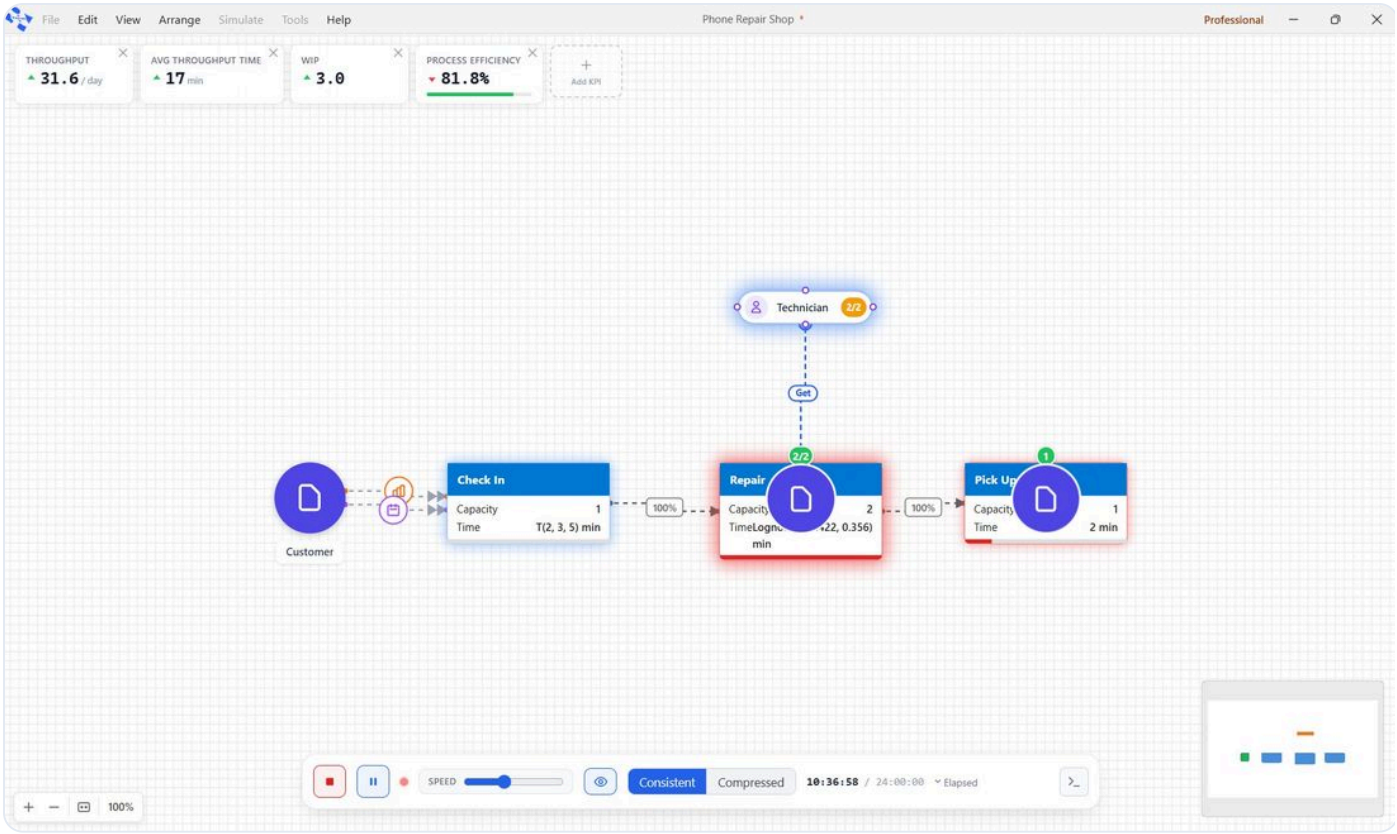



Figure 7. Just after 10: the office group works its way through Check In and on to Repair, where both Technicians are busy.

At **noon**, the rush begins. This time the line builds at **Repair**, up to five near 1 o'clock, and then drains away as the rush passes.

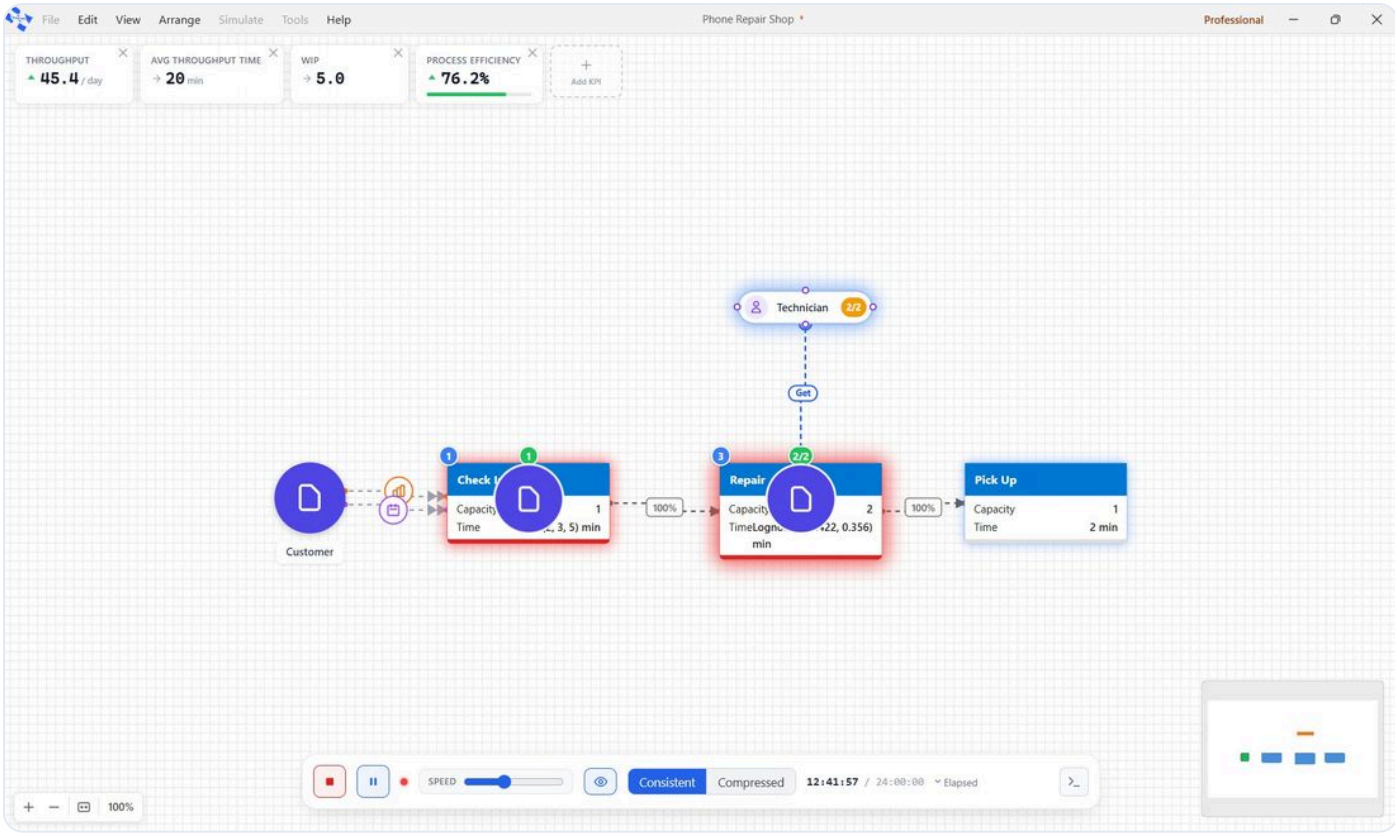



Figure 8. The lunchtime rush: both Technicians busy and customers waiting at Repair.

The afternoon is calmer, with a short wait now and then. In the evening fewer come, and the last arrive before 8. Notice that the two Technicians are still on all night: in this model they work around the clock, until Chapter 7 gives them shifts.

Step 6.

## Read the cards

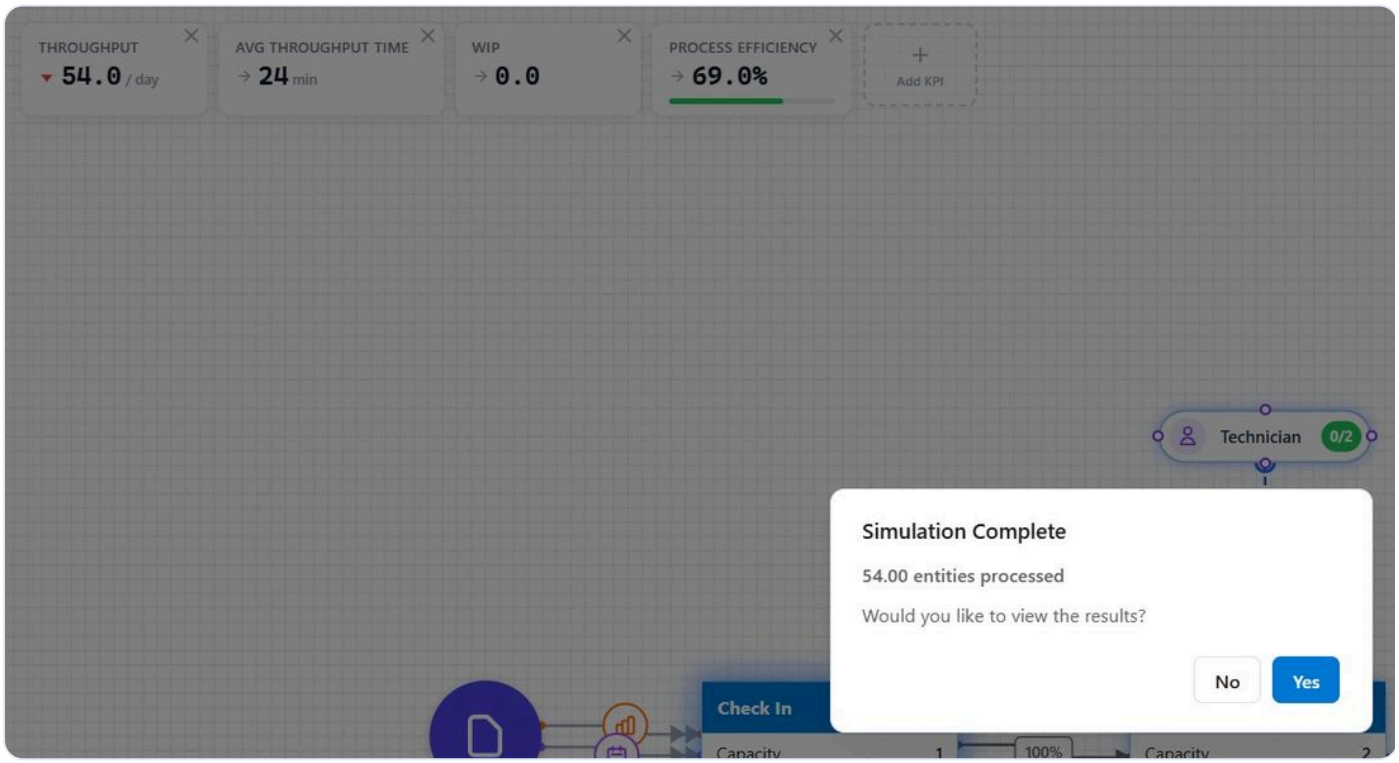



Figure 9. One whole Monday: 54 customers, 24 minutes in the shop, 69% efficiency.

- **54** went through: the 48 walk-ins and the 6 from the office.
- **Avg Throughput Time, 24 minutes**, up from 20 in Chapter 3.
- **WIP, 0**: at midnight the shop is empty.
- **Process Efficiency, 69%**, down from 89.2%.

Compare with Chapter 3. Only a few more customers came (54 against 45), with the same staff and the same times. But they came **bunched together**, so lines formed at set hours, and the average wait grew. The day's total was never the problem; its shape was.

As before, this is one run: one possible Monday. Chapter 10 runs many. Click **No**.

Step 7.

## Save it

1. Click **File**, then **Save**. The star leaves the title bar, and the shop keeps its name. Chapter 5 starts from it.

## Your practice model

START FROM

**Chapter 3. Make Your Times Vary**

The shop as Chapter 3 left it. Use your own saved Phone Repair Shop instead if you built it.

COMPARE WITH

**Chapter 4. Make Customers Come When They Really Do**

The shop as this chapter leaves it. Open it when you are done, and check your settings against it.

Both are in **File** ▸ **Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**You set noon to 30% in the Daily Pattern Builder. What happens to the other hours?**

A. Nothing: the day now totals 130%

B. They share the remaining 70%, so the curve still totals 100%

C. They are set to zero

The answer is B. Noon locks at 30% and the other hours rebalance. The curve shares out the day's quantity, so it always totals 100%.

QUESTION 2

**Your run is 8 hours long and every run starts at midnight on Monday. What would the lunchtime rush look like in the run?**

A It would be there, but squeezed into the 8 hours

B It would not appear at all: the run would end at 8 in the morning

C It would move to the first hour of the run

The answer is B. The pattern lives on the clock. An 8-hour run from midnight stops at 8 in the morning, hours before noon, which is why the run was set to 24 hours.

QUESTION 3

**Six staff from one office bring their phones at 10 every Monday. Why a Scheduled arrival rather than adding them to the daily pattern?**

A Because they come together at a known time, and a pattern spreads customers at random inside each hour

B Because a daily pattern cannot hold more than 48 customers

C Because Scheduled arrivals skip Check In

The answer is A. A schedule brings a set number at an exact time, as a group. A pattern spreads its customers at random moments through each hour.

## Try it yourself

What if the office group came at the worst possible moment?

1. Open the office group's arrival, click **Open Schedule Builder**, and change the time from 10 to **12**, so the six arrive in the middle of the lunchtime rush. Click **Save Schedule**.
2. Run the day and read the cards before you answer Simulation Complete.

> **What to expect.** The same 54 customers go through, but expect a longer line at Repair around noon and a higher Avg Throughput Time than 24 minutes. In one run of the practice model, the count on Repair's corner reached 7 at about 1:30, and Avg Throughput Time was 32 minutes: the group now lands on top of the busiest hour instead of in a quiet one. Nothing about the amount of work changed, only its timing. That is the case for asking a client to come at 10, not at noon. Set the time back to **10** and save before Chapter 5.

CHAPTER 5 OF 15

# Chapter 5. Send Each Job Down the Right Path

By the end of this chapter you can

**Send each job down the right path, by percentage or by condition.**

Watch the video (7 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 5. Send Each Job Down the Right Path**
www.youtube.com/watch?v=DnUxziMQYYk

## Before you start

Until now every customer took the same path: Check In, Repair, Pick Up. In a real shop, not every phone needs the full repair. Many are fixed in a few minutes at the bench; only the stubborn ones go on to a full repair. And some customers are different from the start: an office's phones under warranty go straight to Repair.

Deciding where each job goes next is called **routing**, and ProcessModel gives you two ways to do it. This chapter teaches both, and when to use which:

- **By percentage**: a share of the jobs goes each way, by chance. Use it when you know how often something happens, but not for which job. "About 70% are fixed quickly."
- **By condition**: a rule looks at each job and sends it the right way. Use it when you can tell from the job itself. "Office phones go straight to Repair."

Here is what changes in the shop:

| Part | Chapter 4 | Chapter 5 |
|---|---|---|
| Quick Fix | None | A new step after Check In: 3 to 10 minutes, mostly 5, done by a Technician, two at a time |
| Walk-ins after Check In | All go to Repair | All try Quick Fix first: 70% are fixed and go to Pick Up, 30% go on to Repair |

| Part | Chapter 4 | Chapter 5 |
|---|---|---|
| The office group | Six Customers at 10 on Monday | Their own kind of customer, **Office Customer**, sent by a condition straight to Repair |
| Steps, staff and run | As Chapter 4 left them | Unchanged; the two Technicians now share Quick Fix and Repair |

The 70 and 30, the Quick Fix time and the name Office Customer are made up for this course.

**You need** your Phone Repair Shop from Chapter 4 (or the practice model below) and about forty minutes. This is the longest chapter so far; take it one step at a time.

---

Step 1.

## Open your shop

1. On the Welcome screen, under **Recent Models**, click **Phone Repair Shop**.

**What you should see:** the shop as Chapter 4 left it, with its two arrival lines.

Step 2.

# Add the Quick Fix step

1. Drag an **Activity** from the palette and drop it right above **Check In**, level with the Technician, so the lines you draw next run straight.
2. Name it **Quick Fix**.
3. Click in **Processing Time** and type **T(3, 5, 10)**: a triangular time from 3 to 10 minutes, mostly 5. You built the same kind by hand in Chapter 3; typing it directly is quicker once you know the form, T(least, most likely, most). **M** is already chosen.
4. Click in **Capacity** and type **2**, so both Technicians can do quick fixes at once. Click the canvas to keep it.

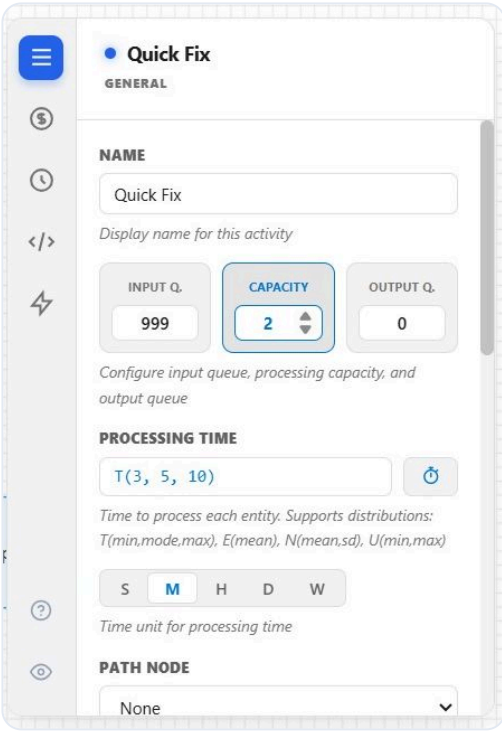



Figure 1. Quick Fix: 3 to 10 minutes, mostly 5, two at a time.

5. Draw a line from the **Technician** to **Quick Fix**, and click it. It is **Get / Free**, as the line to Repair is: Quick Fix takes a Technician for each job and lets them go after. The same two Technicians now do both jobs.

Step 3.

## Split Quick Fix's routes by percentage

1. Draw a line from **Quick Fix** to **Pick Up**. It reads **100%**: so far, every job leaving Quick Fix goes to Pick Up.
2. Draw a second line, from **Quick Fix** to **Repair**. Now both read **50%**.

Why 50 each? Because **Percent routes leaving one step always total 100%**. Every job that leaves Quick Fix must go somewhere, so the shares must add up to all of them. When you add a route, ProcessModel shares the total between them.

3. Click the new line, to Repair, and scroll down the panel to **Sibling Routes**: every route that leaves Quick Fix, with its share.
4. In **Pick Up**'s box, type **70**.

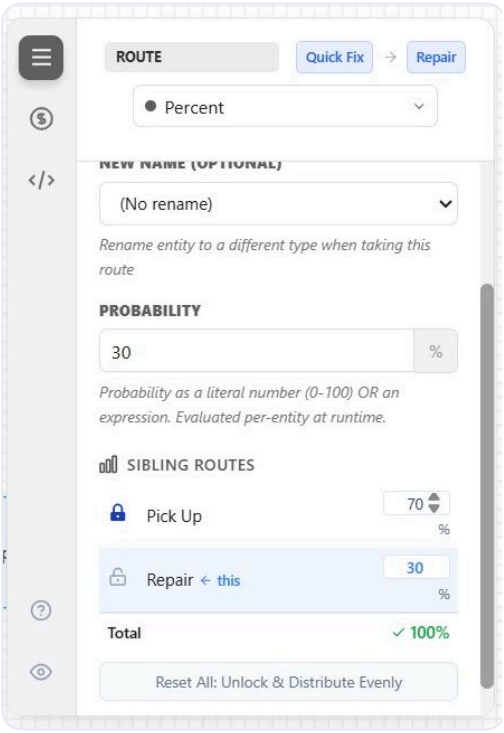



Figure 2. Sibling Routes: Pick Up set to 70, and Repair becomes 30 by itself. Total, 100%.

**What you should see:** Repair becomes **30** by itself, and the total stays at **100%**. You set one share; ProcessModel balances the other.

5. Click the canvas.

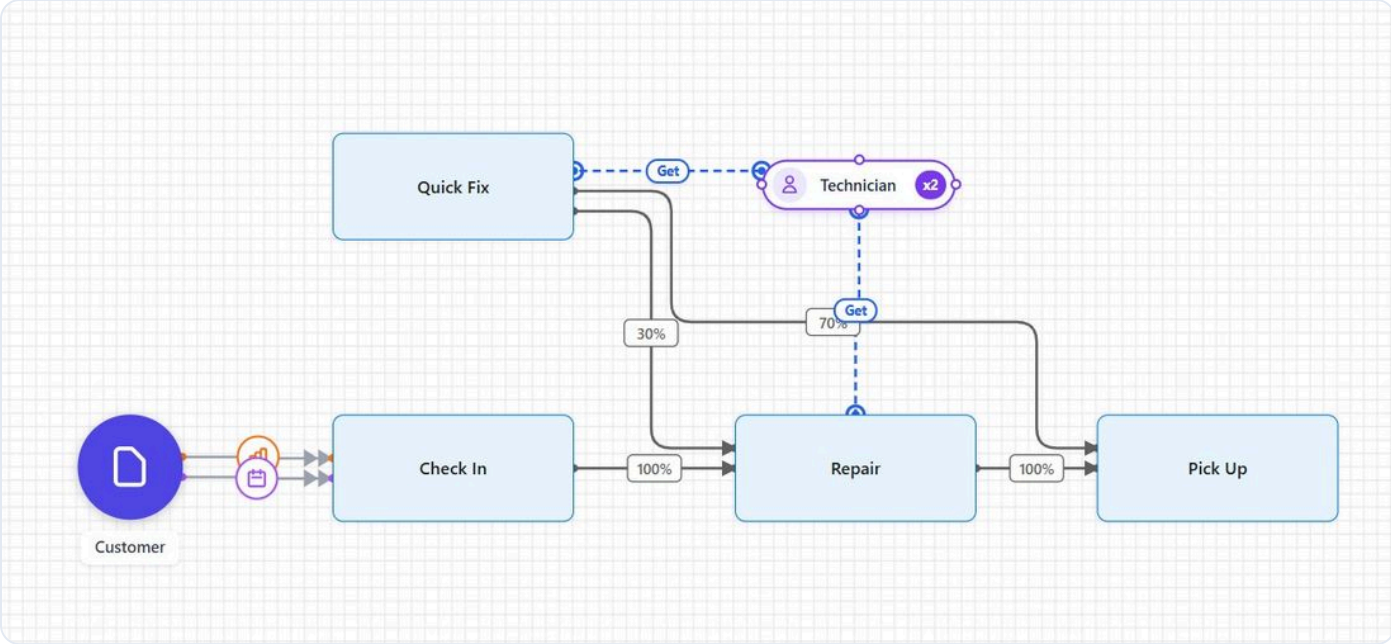



Figure 3. The lines out of Quick Fix read 70% and 30%.

Each job leaving Quick Fix now takes its chance: 7 in 10 are fixed and go to Pick Up, 3 in 10 need the full repair.

Step 4.

## Make the office group its own kind of customer

To treat the office's phones differently, the model must be able to tell them apart. So far they are Customers, like everyone else. Give them their own entity.

1. Drag an **Entity** onto the canvas, below **Customer**, and name it **Office Customer**.
2. Give it its own color, so it stands out in the run: click **Color** and choose **Orange** from the palette in the panel. Office Customer turns orange.
3. Click the office group's arrival line, the lower one from Customer. It reads **Scheduled**: 1 entry, 6 entities, at 10 on Monday.
4. Drag the end of that line off **Customer** and drop it on one of **Office Customer**'s connection points, the small handles on its edge.

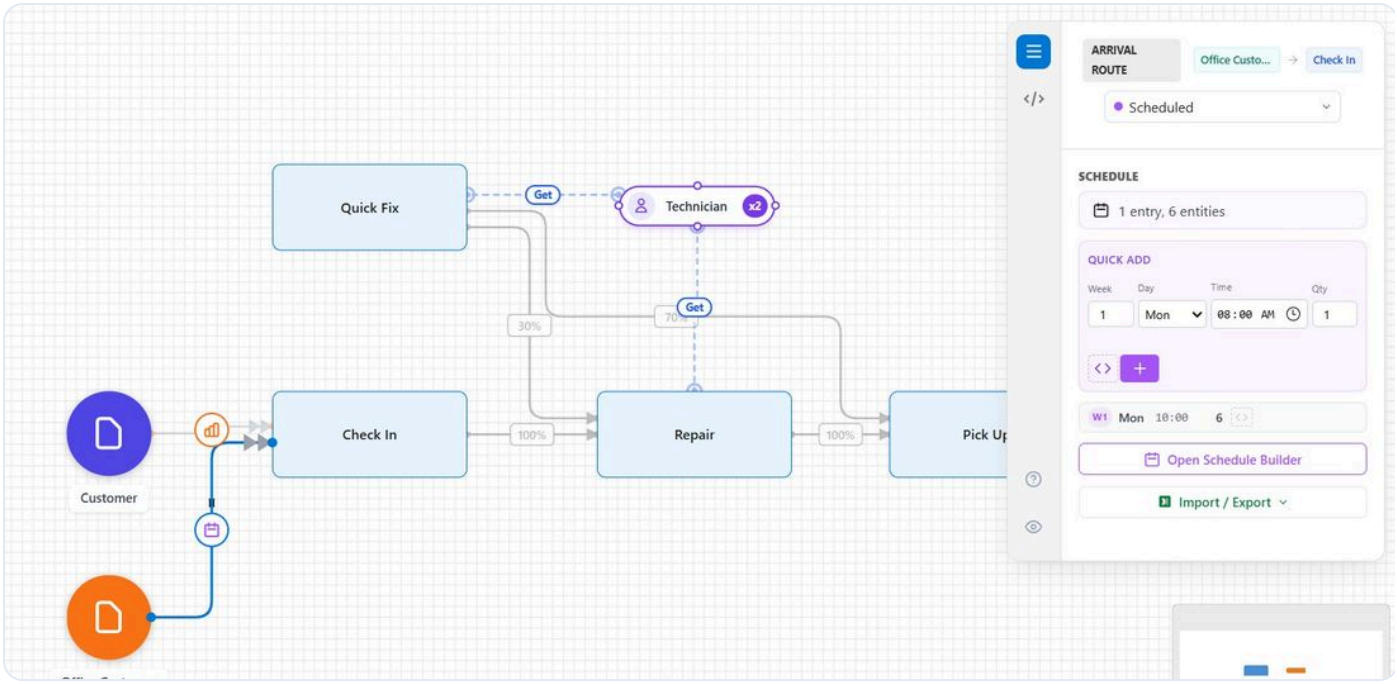



Figure 4. The scheduled arrival now starts at Office Customer, still Scheduled: the same six, at 10.

**What you should see:** the line now starts at Office Customer, and it is still Scheduled: the same six at 10, now arriving as Office Customers. Drop the end on a connection point, not on the middle of the shape, so the line moves rather than a new one being drawn.

Step 5.

# Route by condition at Check In

Now the rule: office phones are under warranty, so they skip Quick Fix and go straight to Repair. Everyone else tries Quick Fix first. The decision is made at **Check In**, and it depends on who the customer is, not on chance: a **condition**.

One rule shapes the order of what you do next: **a step routes either by chance or by condition, never both.** The routes leaving one step are all Percent or all Conditional. That is why the chance lives at Quick Fix and the condition at Check In. It is also why you change Check In's existing route to Conditional **before** drawing its new one: a new route takes the type of the routes already there.

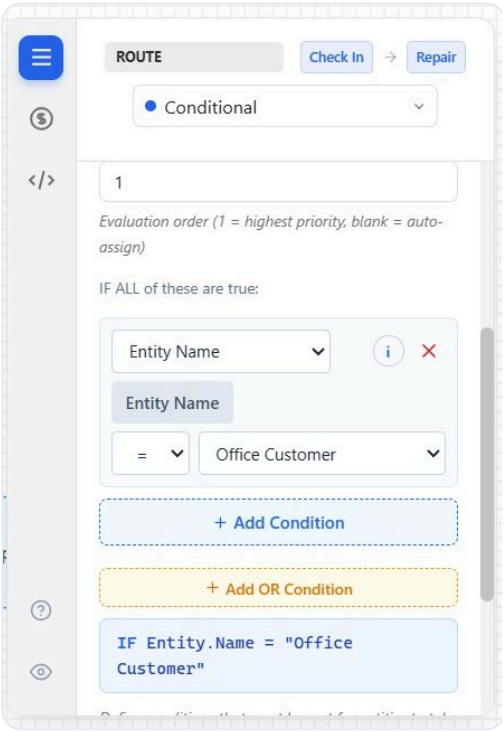



Figure 5. The condition: IF Entity.Name = "Office Customer". Office phones go straight to Repair.

1. Click the line from **Check In** to **Repair**, open its **Type** list, and choose **Conditional**.
2. In **Priority**, type **1**. Conditional routes are checked in priority order, 1 first, and the first one that matches wins.
3. Scroll down and click **Add Condition**. A new row appears.
4. In its first list, choose **Entity Name**: the rule looks at each customer's kind.
5. The middle list is already **=**, equals. In the last list, choose **Office Customer**.

> **What you should see:** the panel sums the rule up as **IF Entity.Name = "Office Customer"**.

6. Now draw a line from **Check In** to **Quick Fix**, and click it.

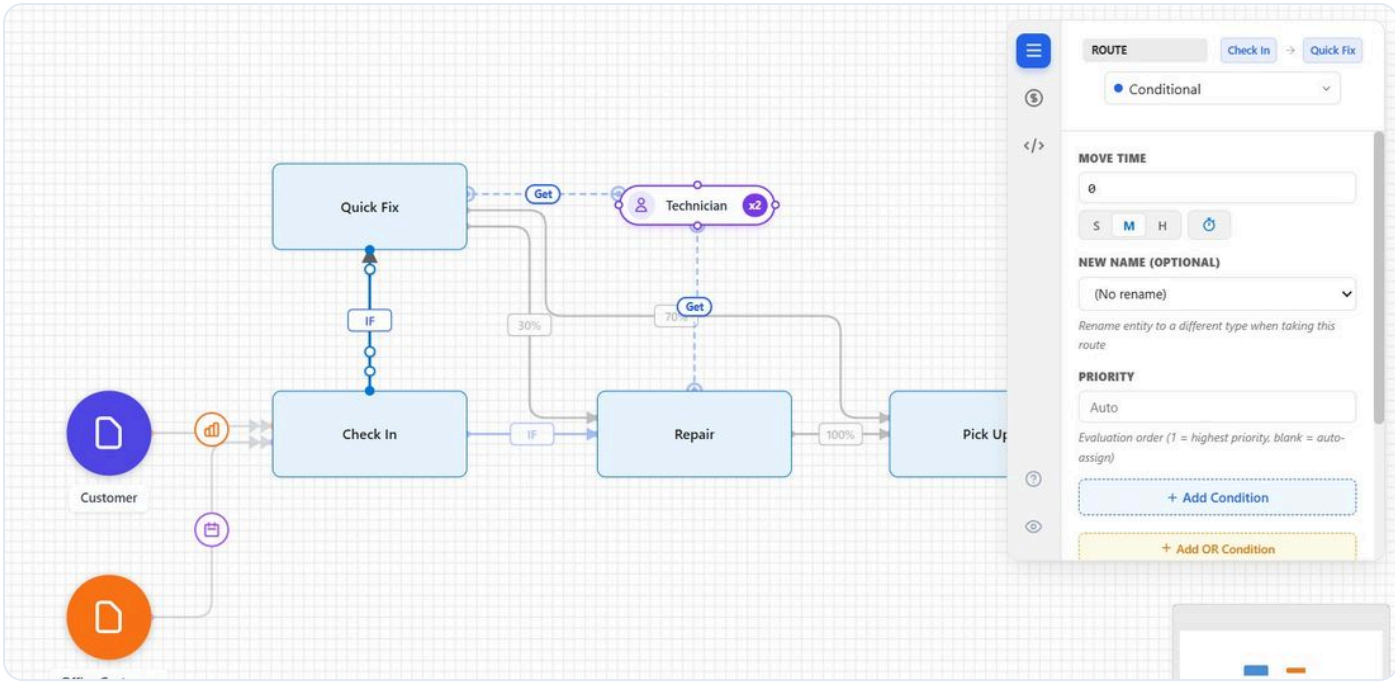



Figure 6. The new route is born Conditional, with no condition: the catch-all.

> **What you should see:** it is born **Conditional**, with Priority **Auto** and **no condition**. Leave it that way. A Conditional route with no condition is the **catch-all**: a customer who matches no other condition takes it. So office customers match the rule and go to Repair, and everyone else, all the walk-ins, goes to Quick Fix. Both of Check In's lines are now marked **IF**.

Step 6.

## Run the day and watch the paths

1. Click **Run Simulation**.

From 7, walk-ins arrive. Each one checks in, then goes up to Quick Fix. Most leave from there for Pick Up, and about 3 in 10 go on to Repair. At 10, the six office customers arrive together, in orange. Watch them **skip Quick Fix** and go straight to Repair, while the walk-ins keep going to Quick Fix.

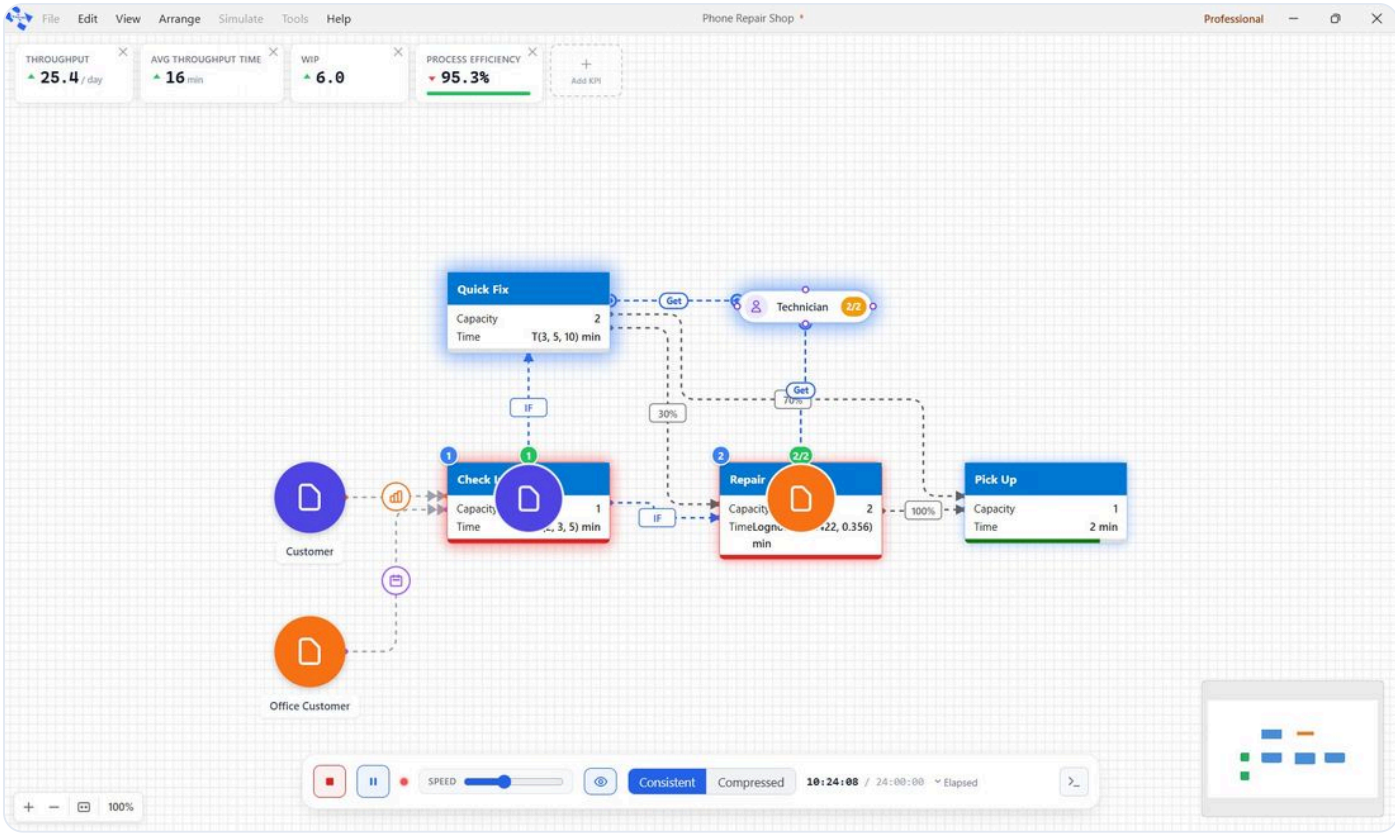



Figure 7. Just after 10: an office customer, in orange, at Repair, while walk-ins check in on their way to Quick Fix.

At noon, rest the pointer on the **Technician**: its popup shows how many of the two are in use, and at the busiest moments it is both. They are busier now, because they do both jobs. Yet no long line builds: most walk-ins are done within minutes at Quick Fix.




Figure 8. Just after noon: both Technicians in use, 2 of 2.

Step 7.

## Read the cards

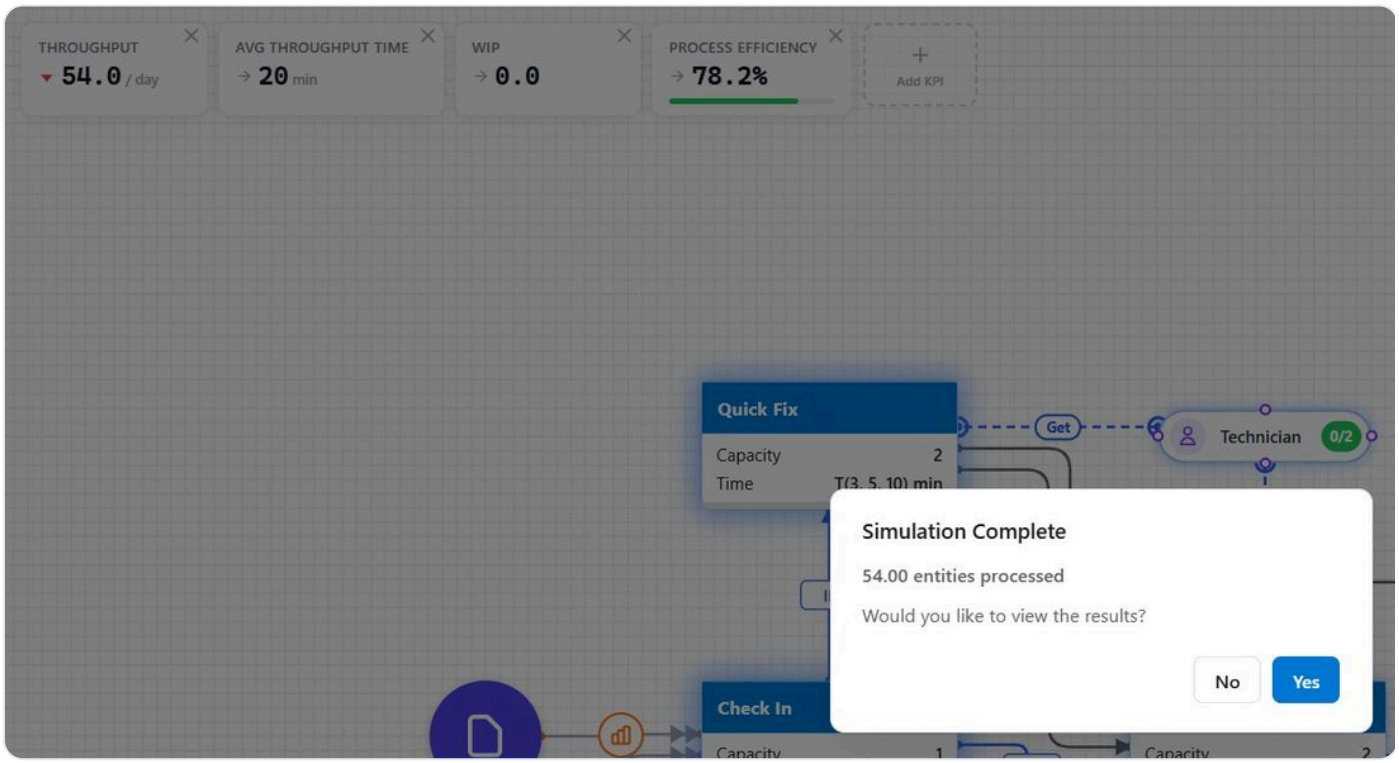



Figure 9. The day with routing: 54 customers, 20 minutes in the shop, 78.2% efficiency.

- **54** went through, as in Chapter 4: the same customers came.
- **Avg Throughput Time, 20 minutes**, down from 24 in Chapter 4: most phones now leave after a short fix instead of a full repair.
- **WIP, 0**: at midnight the shop is empty.
- **Process Efficiency, 78.2%**, up from 69%.

The same customers, the same two Technicians, and a better day, because each job takes the path it needs. That is the power of routing. One run, one possible Monday, as always; Chapter 10 runs many. Click **No**.

Step 8.

## Save it

1. Click **File**, then **Save**. The star leaves the title bar. Chapter 6 starts from this shop.

## Your practice model

START FROM

**Chapter 4. Make Customers Come When They Really Do**

The shop as Chapter 4 left it. Use your own saved Phone Repair Shop instead if you built it.

COMPARE WITH

**Chapter 5. Send Each Job Down the Right Path**

The shop as this chapter leaves it. Open it when you are done, and check your settings against it.

Both are in **File** ▸ **Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**Quick Fix has two Percent routes. You type 70 for the route to Pick Up. What does the route to Repair become, and why?**

- **A** 30, because Percent routes leaving one step always total 100%
- **B** 50, because it keeps its own value
- **C** 70, because routes from one step share the same value

The answer is A. Every job that leaves a step must go somewhere, so the shares total 100%. ProcessModel rebalances the other route for you.

QUESTION 2

**Why does the 70/30 chance live at Quick Fix, while the condition lives at Check In?**

A Because Quick Fix cannot have a condition

B Because the routes leaving one step are all by chance or all by condition, never a mix

C Because conditions only work on the first step

The answer is B. A step routes by percentage or by condition, not both. Check In decides by who the customer is; Quick Fix decides by chance.

QUESTION 3

**The route from Check In to Quick Fix has no condition. Which customers take it?**

A None: a route with no condition is never used

B Every customer who matches no other condition: all the walk-ins

C Only the Office Customers

The answer is B. A Conditional route with no condition is the catch-all. Office Customers match the Priority 1 rule to Repair; everyone else takes the catch-all.

## Try it yourself

How much is the quick fix worth? Suppose only half the phones could be fixed quickly.

1. Click the route from Quick Fix to Pick Up, scroll to **Sibling Routes**, and type **50** for Pick Up. Repair becomes 50.

2 Run the day and read the cards before you answer Simulation Complete.

**What to expect.** Expect Avg Throughput Time to rise above this chapter's 20 minutes. In one run of the practice model it was 23 minutes, and Process Efficiency fell from 78.2% to 74.0%: more phones now go on to the full repair, which takes longer, and with the Technicians busier, more customers wait for one. The 70% quick-fix rate is what made the shop faster, so in a real project it is the number most worth checking against your records. Set Pick Up back to **70** and save before Chapter 6.

CHAPTER 6 OF 15

# Chapter 6. Give Each Line a Size and an Order

By the end of this chapter you can

**Set how much a step holds and who in a line goes first.**

Watch the video (7 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 6. Give Each Line a Size and an Order**
www.youtube.com/watch?v=14NL73FBsuY

## Before you start

So far, a customer waiting for Repair joined a line with no end: Repair's own waiting room could hold 999 people. No real shop has a waiting room like that. A bench holds a few people; a counter has room for a few phones. And every waiting area has an **order**: who goes next when a Technician is free.

Both matter, and in a way that surprises people the first time. When a waiting area is **full**, the customer who has finished the step before it has nowhere to go. They stay where they are, and the step they are standing in cannot take anyone new. A full line downstream **stops work upstream**. This is called blocking, and it is how a small bottleneck in one place causes a long line somewhere else.

In this chapter you give the shop's repair line a size and an order, then watch blocking happen. Two settings carry the whole lesson:

- **Capacity** of a waiting area: how many can wait there.
- **Queue discipline**: which waiting customer goes next.

Here is what changes in the shop:

| Part | Chapter 5 | Chapter 6 |
|---|---|---|
| The office group | Six customers at 10 on Monday | Twelve customers at 10 on Monday |

| Part | Chapter 5 | Chapter 6 |
|---|---|---|
| Repair's own waiting room | Input Q 999: a line with no end | Input Q 0: none |
| Repair Queue | None | A new waiting area before Repair, for two customers; the one longest in the shop goes first |
| Routes into Repair | From Check In and from Quick Fix | Both now end at the Repair Queue, which leads on to Repair |
| Steps, staff and run | As Chapter 5 left them | Unchanged; Technician, Repair and Pick Up move right to make room |

**You need** your Phone Repair Shop from Chapter 5 (or the practice model below) and about forty minutes.

Step 1.

## Open your shop

1. On the Welcome screen, under **Recent Models**, click **Phone Repair Shop**.

> **What you should see:** the shop as Chapter 5 left it, with Quick Fix and the orange Office Customer.

Step 2.

## Send twelve from the office

To see what a full line does, the shop needs a burst of work bigger than it can take at once. The office now sends twelve customers at 10.

1. Click the office group's arrival line, from **Office Customer**. It reads 1 entry, 6 entities.
2. Click **Open Schedule Builder**. The row holds 6 customers at 10 on Monday morning.
3. Click its **Quantity** and type **12**. The foot reads **Total Quantity 12**.

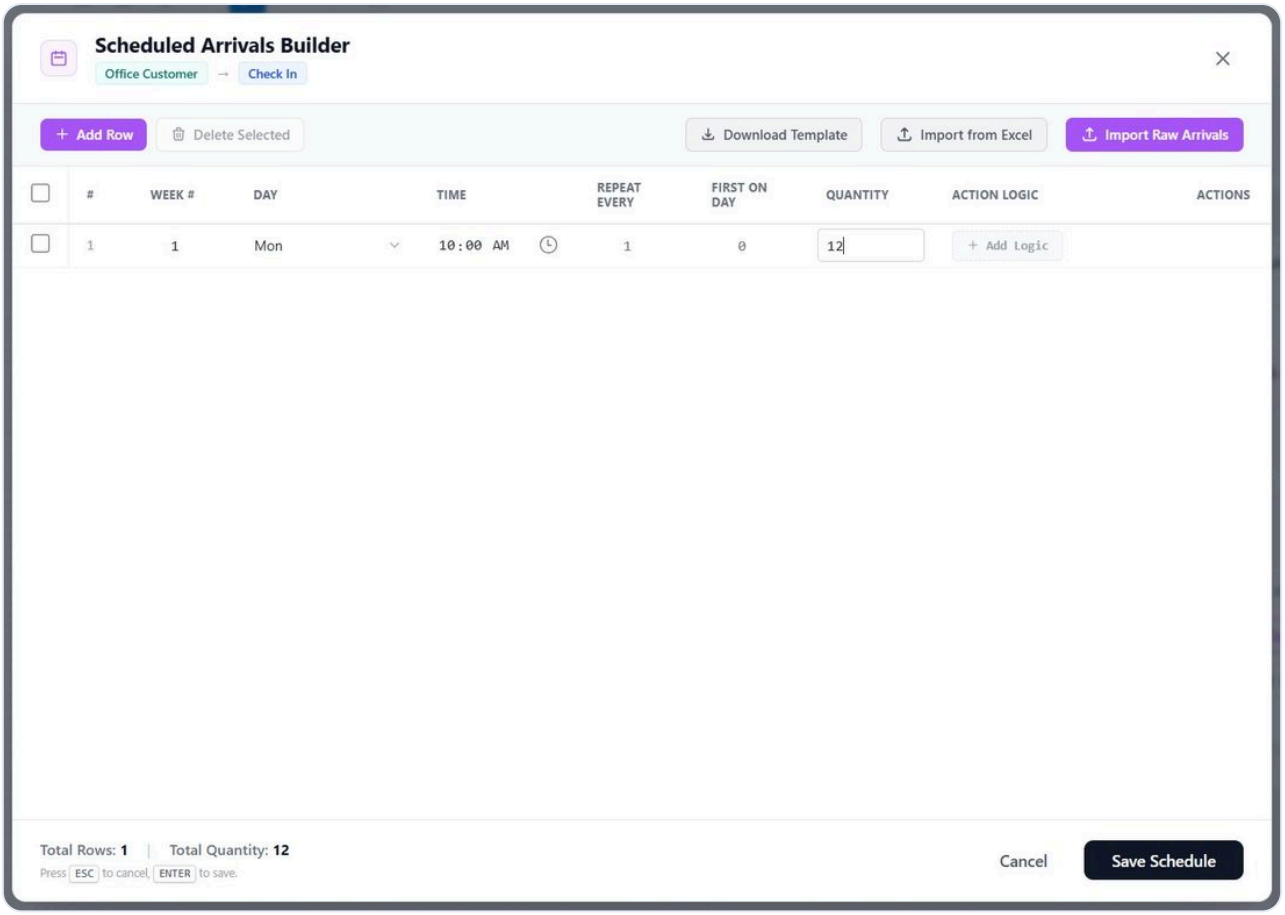



Figure 1. The office group grows to twelve, still at 10 on Monday.

4 Click **Save Schedule**. The panel now reads **1 entry, 12 entities**.

Step 3.

## Take away Repair's own waiting room

Every activity has three small boxes above its time: **Input Q**, **Capacity** and **Output Q**. Capacity is how many it works on at once. Input Q is its own waiting room in front of it, and Output Q is room after it for customers who are done but cannot move on yet. In both, 0 means none.

1. Click **Repair**. Capacity reads 2: two repairs at once. Input Q reads **999**: a line with no end.
2. Click **Input Q** and type **0**.

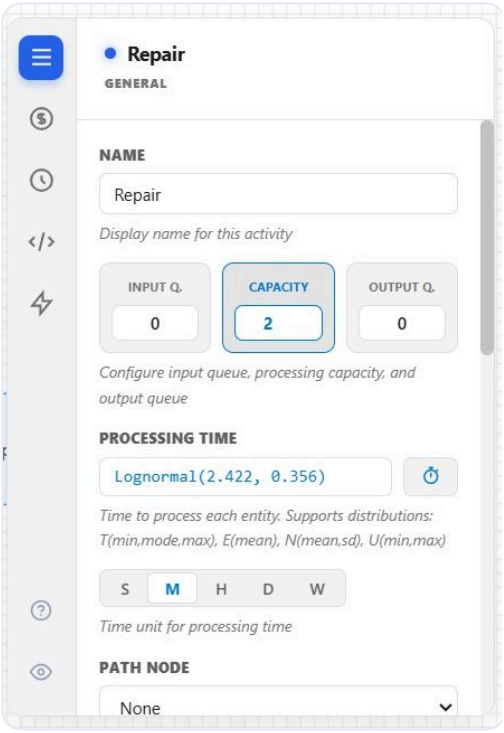



Figure 2. Repair keeps no waiting room of its own: Input Q 0.

**Why:** you want one place, and only one, where customers wait for a repair. That place is the Repair Queue you add next, and you will give it a size and an order.

Step 4.

## Make room and add the Repair Queue

The Repair Queue goes where Repair stands now, so first move three shapes right.

1. Click **Technician**, then hold **Ctrl** and click **Repair** and **Pick Up**. All three are selected.
2. Drag them right together. Every line stays attached as they move.

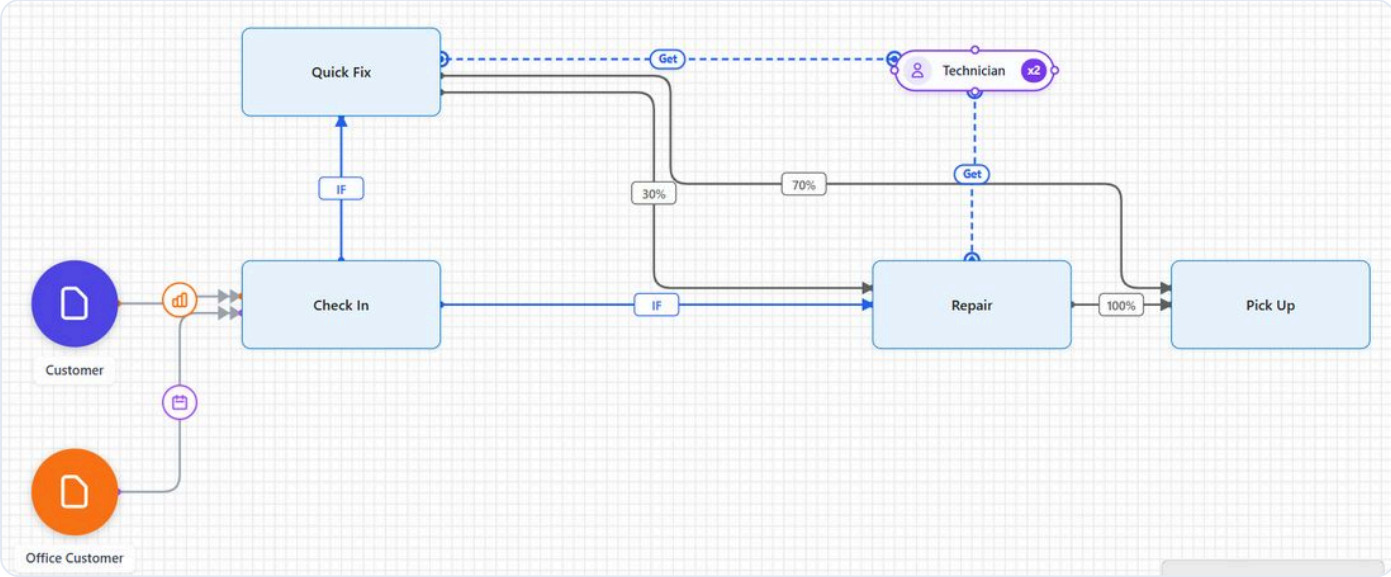



Figure 3. Technician, Repair and Pick Up moved right together, every line still attached, with room where Repair stood.

3. Drag a **Queue** from the palette and drop it where Repair was, in line with Check In. A **Queue** is a waiting area: nothing is done to the customers there, they only wait.
4. Name it **Repair Queue**.
5. Click **Capacity** and type **2**: the queue holds two customers.
6. Click **Queue Discipline** and choose **Oldest in System**.

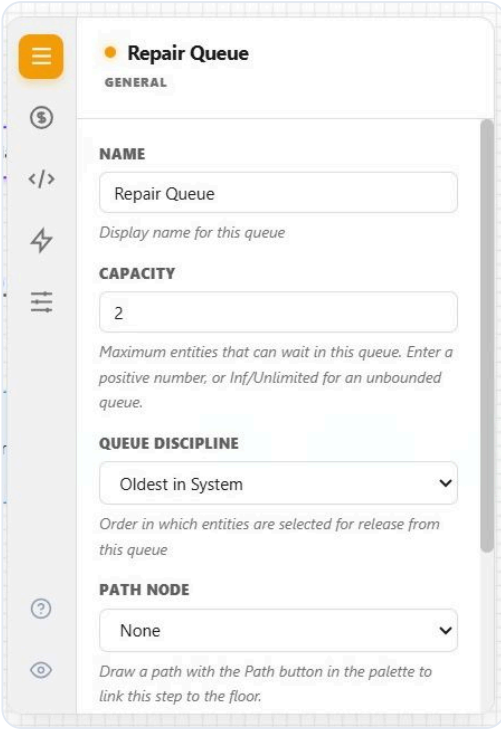



Figure 4. The Repair Queue: room for two, and the customer longest in the shop goes first.

Queue discipline is the order. A queue that serves first come, first served is fair to the queue, but **Oldest in System** is fair to the customer: it looks at how long each one has been in the shop, not just in this line. A walk-in who already tried Quick Fix has been here longer than an office customer who just checked in, so they go first. The list offers other orders too, such as last in first out, random, or by an attribute (Chapter 8 teaches attributes).

7 Scroll down to **Animation Display** and tick **Show queue fill indicator**.

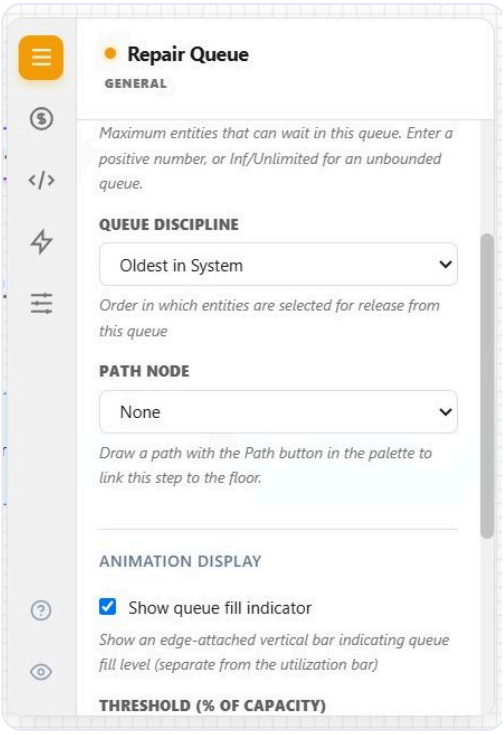



Figure 5. The fill indicator: a bar on the queue's edge that shows how full it is during a run.

During the run, a bar on the queue’s edge will show how full it is, so you can see the moment it fills.

Step 5.

## Send every repair through the queue

Now the two routes that went into Repair must end at the Repair Queue instead.

1. Click the office route, from **Check In** to **Repair**. Drag its end off Repair and drop it on the **Repair Queue**. Its panel stays open: it is still **Conditional**, Priority **1**, and scrolling down shows the condition is kept, IF Entity.Name equals “Office Customer”.
2. Click Quick Fix’s route to **Repair** and drag its end onto the Repair Queue too. It is still **Percent**, with Probability **30**.

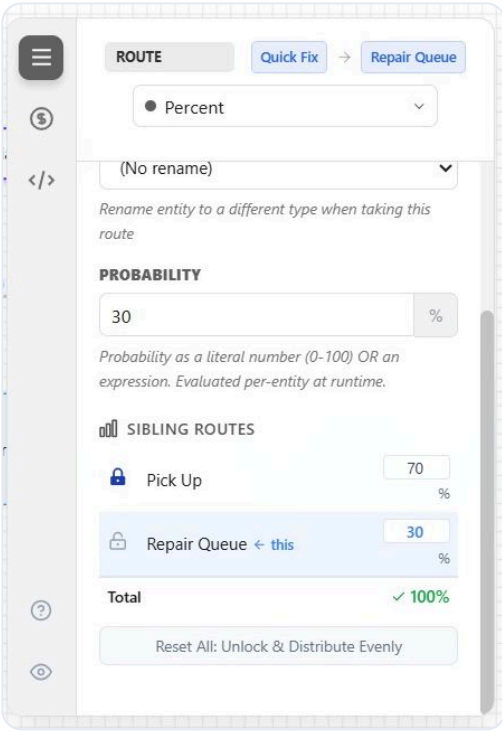



Figure 6. Moving a route's end keeps its settings: still 30%, now to the Repair Queue.

Moving a route’s end keeps everything you set on it. You do not have to build the condition or the percentages again.

3. Draw a line from **Repair Queue** to **Repair**. It reads **100%**: every customer in the queue goes on to Repair.

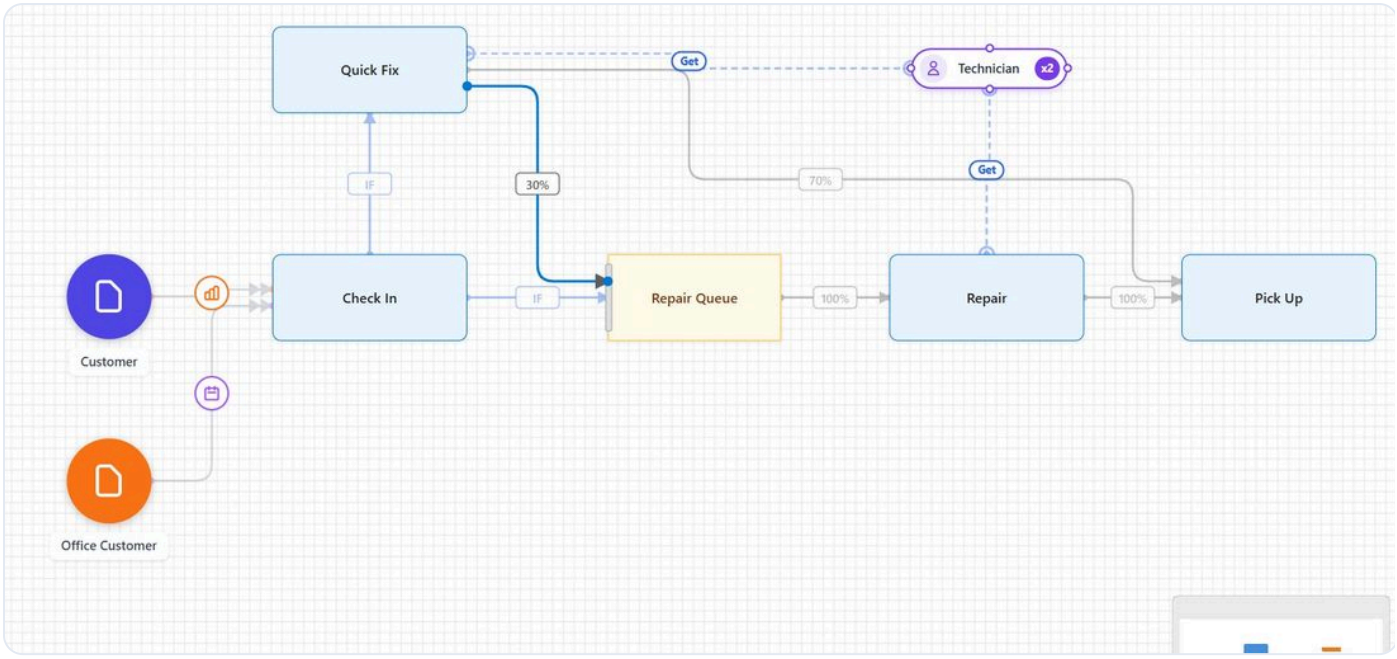



Figure 7. Every repair now waits in the Repair Queue, the one place to wait for a repair.

Step 6.

## Watch the queue fill and Check In stop

1 Click **Run Simulation**.

Until 10, the day goes as in Chapter 5. At **10**, the office sends twelve customers at once, and here is the chain to watch, link by link:

- Check In serves one customer at a time, and each office customer goes on to the Repair Queue.
- Repair works on two. The Repair Queue holds two more. That is four office customers in the repair area.
- The next one finishes Check In and has **nowhere to go**: the queue is full, and Repair has no waiting room of its own. So they stay at Check In.
- While they stay, Check In cannot take anyone new. Check In's line grows.

Watch the bar on the queue's edge: yellow with one customer, then red and pulsing with two. Rest the pointer on **Check In**: its popup reads **Blocked**.

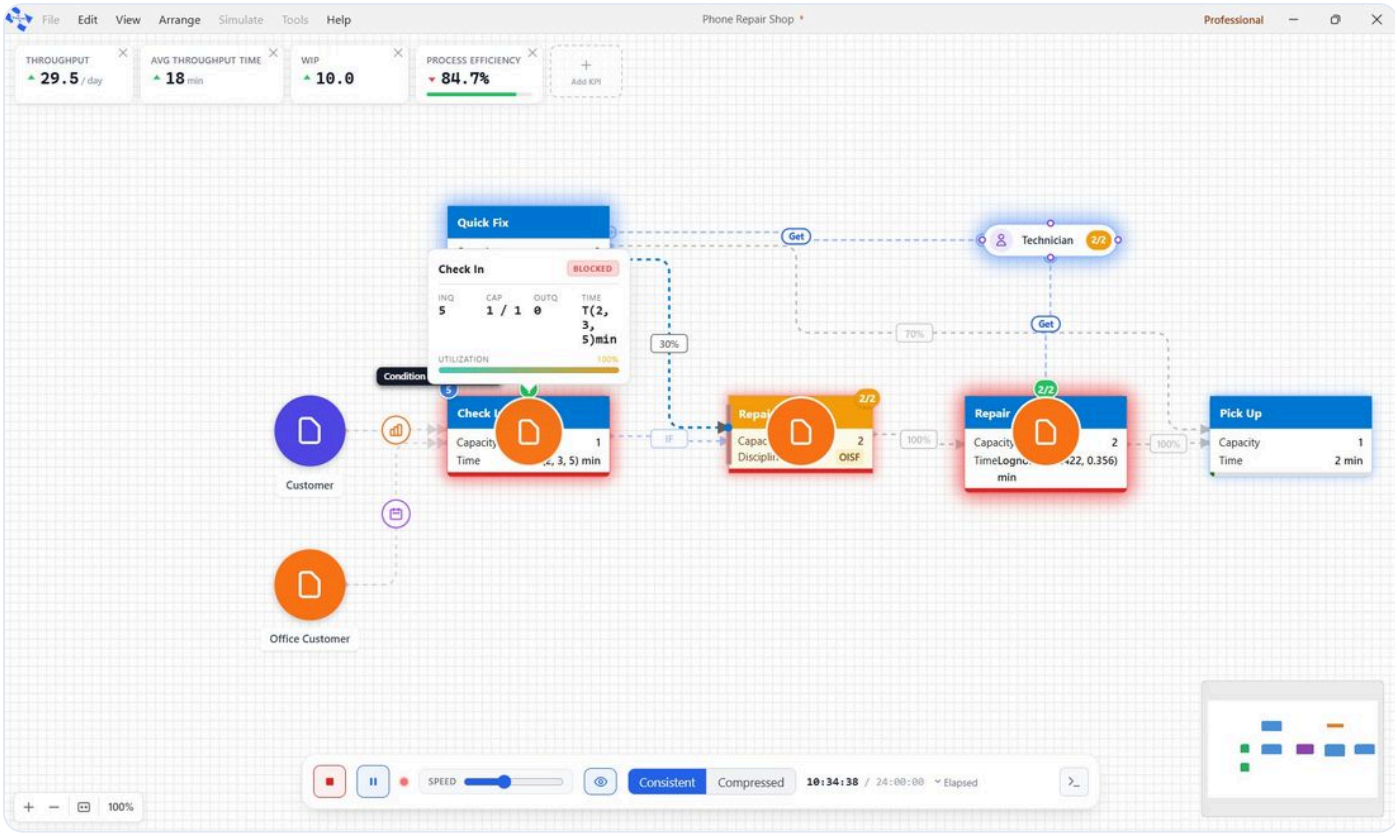



Figure 8. Check In is blocked: the Repair Queue is full, so the customer at Check In has nowhere to go.

When a repair ends, the queue moves up, the customer at Check In moves into it, and Check In starts again. Soon after 11 its line is gone, and then the queue empties.

At noon, the walk-ins' rush comes, and the queue's order shows its worth: a walk-in who tried Quick Fix has usually been in the shop longest, so they move up ahead. The afternoon is calmer, and the queue does not fill again.

Step 7.

## Read the cards

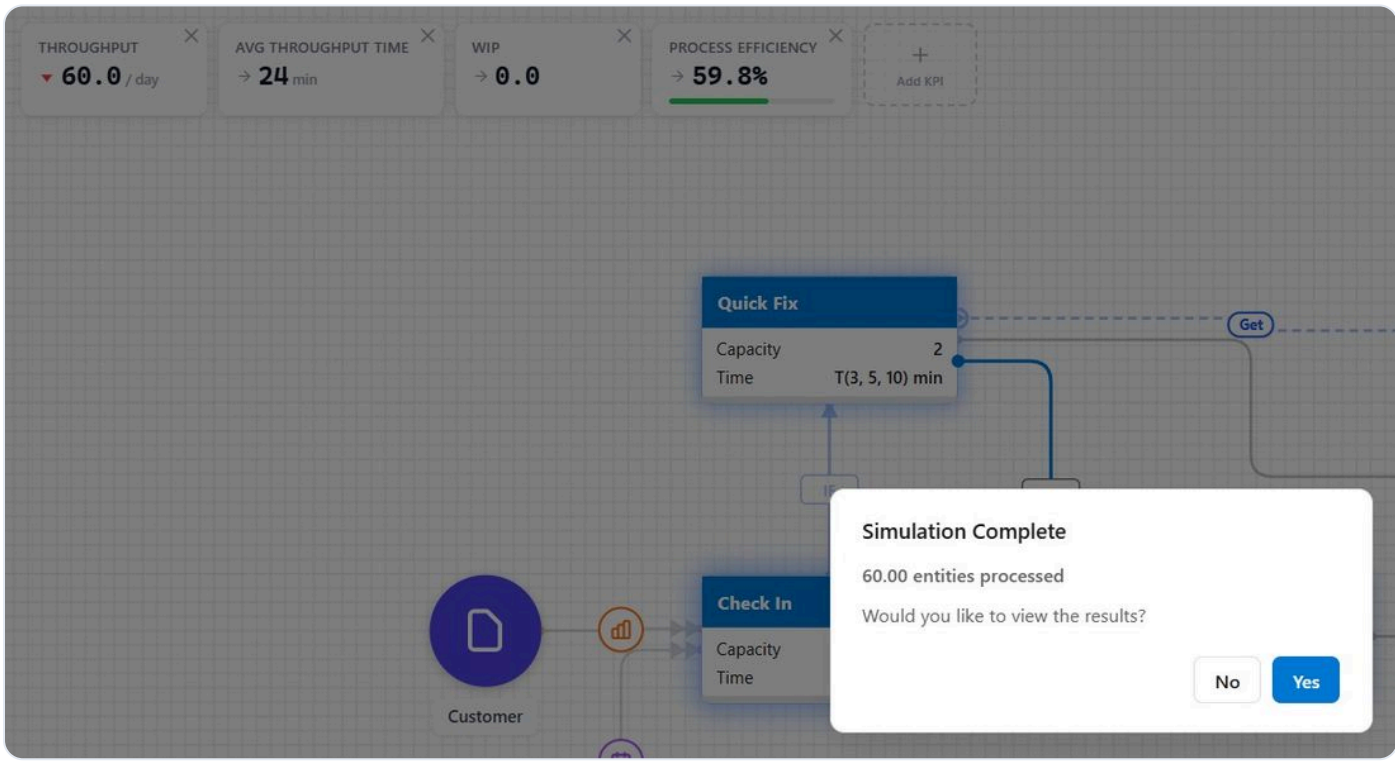



Figure 9. One Monday with a sized queue: 60 customers, 24 minutes in the shop, 59.8% efficiency.

- **60** went through: the 48 walk-ins and the 12 from the office.
- **Avg Throughput Time, 24 minutes**, up from 20 in Chapter 5: twelve customers at once means more waiting.
- **WIP, 0**: at midnight the shop is empty.
- **Process Efficiency, 59.8%**: more of each customer's time is spent waiting.

The small arrows on the cards compare with the run's last hour, not with Chapter 5, so compare chapters by the numbers themselves. As always, one run is one possible Monday; Chapter 10 runs many. Click **No**.

Step 8.

## Save it

1. Click **File**, then **Save**. The star leaves the title bar. Chapter 7 starts from this shop.

## Your practice model

START FROM

**Chapter 5. Send Each Job Down the Right Path**

The shop as Chapter 5 left it. Use your own saved Phone Repair Shop instead if you built it.

COMPARE WITH

**Chapter 6. Give Each Line a Size and an Order**

The shop as this chapter leaves it. Open it when you are done, and check your settings against it.

Both are in **File** ▸ **Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**The Repair Queue is full and Repair has no waiting room. A customer finishes Check In and needs Repair. What happens?**

A. They are turned away and leave the shop

B. They stay at Check In, and Check In cannot take anyone new until the queue has room

C. They wait in Check In's own input line, which still has room

The answer is B. A customer with nowhere to go stays where they are. That blocks Check In: a full line downstream stops work upstream.

QUESTION 2

**Why set Repair's Input Q to 0 before adding the Repair Queue?**

- **A** So the Repair Queue is the one place customers wait for a repair, with its own size and order
- **B** Because a Queue cannot be joined to an activity that has an Input Q
- **C** To make repairs faster

The answer is A. With Input Q at 999, customers would simply wait inside Repair's own endless room and the Repair Queue would never fill.

QUESTION 3

**With Oldest in System, who leaves the Repair Queue first?**

- **A** The customer who joined the queue first
- **B** The customer who has been in the shop longest, wherever they waited before
- **C** The customer with the shortest repair

The answer is B. Oldest in System looks at time in the whole shop. A walk-in who already tried Quick Fix usually goes ahead of an office customer who just checked in.

## Try it yourself

How big should the queue be? Make it bigger and find out.

1. Click the **Repair Queue** and set its **Capacity** to **6**.
2. Run the day. At 10, rest the pointer on **Check In** while the office group comes through.

**What to expect.** With room for six, expect the queue to absorb most of the office group, so Check In is held up far less, or not at all, and its line clears sooner. In one run of the practice model, Check In was not held up at all: its line reached 11 just after 10 and was gone by about 11. The repairs still take as long, so the wait moves rather than disappears: customers now wait in the Repair Queue instead of at the door, about 14 minutes on average in that run. The Avg Throughput Time card may now show two figures, one for each kind of customer: in that run, 18 minutes for Customer and 58 for Office Customer. Sizing a waiting area decides **where** people wait. Set the capacity back to **2** and save before Chapter 7.

CHAPTER 7 OF 15

# Chapter 7. Give Your Technicians Real Working Hours

By the end of this chapter you can

**Give your people real shifts, with breaks.**

Watch the video (7 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 7. Give Your Technicians Real Working Hours**
www.youtube.com/watch?v=d2dF9Zskvh8

## Before you start

Until now, the two Technicians were on all day and all night, without a break. Real people work a shift, and they take a lunch break, often at the worst possible moment for the line. A model that ignores the hours people really work will promise a shop that cannot exist.

This chapter teaches three ideas:

- A **shift schedule** sets when a resource works. Outside its hours the resource is **off shift** and takes no new work. Every resource starts **Always Available**, around the clock.
- A **break** is time inside a shift when the resource stops. ProcessModel keeps breaks inside work hours.
- A **resource group** lets a step use whichever of several resources is free and on shift. Two Technicians on two different shifts must be two resources, because one resource has one schedule for all its units. The group joins them back up.

Here is what changes in the shop:

| Part | Chapter 6 | Chapter 7 |
|---|---|---|
| Technician | Two, on all day and all night | **Early Technician**: one person, 7 in the morning to 3 in the afternoon, a break at 11 |
| Late Technician | None | One person, noon to 8:30 at night, a break at 1:30 |

| Part | Chapter 6 | Chapter 7 |
|---|---|---|
| Who does the work | The Technician | Early Technician and Late Technician as **one group**: whichever is on shift and free |
| Everything else | As Chapter 6 left it | Unchanged |

**You need** your Phone Repair Shop from Chapter 6 (or the practice model below) and about forty minutes.

---

Step 1.

## Open your shop

1. On the Welcome screen, under **Recent Models**, click **Phone Repair Shop**.

> **What you should see:** the shop as Chapter 6 left it, with the Repair Queue, and the Technician badge reading x2.

Step 2.

## Add a second Technician for the afternoon

1. Drag a **Resource** from the palette and drop it right above the **Technician**, so no line has to move.
2. Name it **Late Technician**. **Quantity** reads 1: one person.

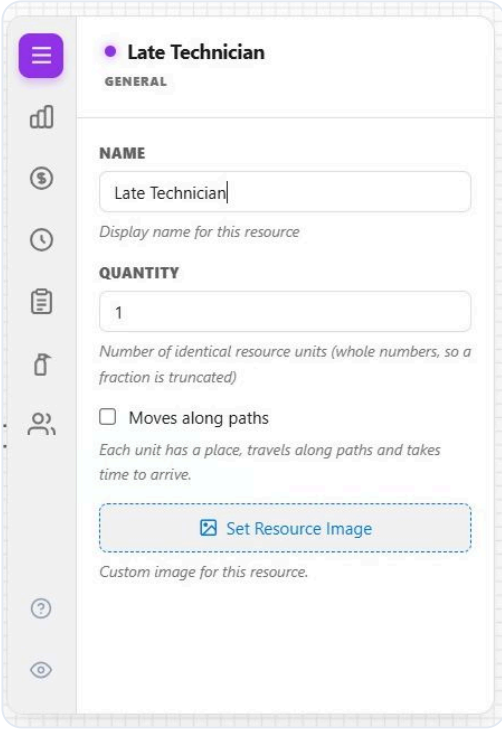



Figure 1. Late Technician: one person.

You do not draw a line from Late Technician to any step. In Step 5 it joins the Technician in a group, and the group does the work.

Step 3.

## Build the late shift

1. Click the **Shift** tab. **Shift Schedule** reads Always Available.
2. Click **New Schedule**. The **Shift Schedule Builder** opens, with the cursor in **Schedule Name**. Type **Late Shift**.

The shop runs one Monday, so Monday, already selected in the day list, is the day to set.

3. Click **Add Work**. A work segment appears. Click its **End Time** and type **8:30 PM**, then click its **Start Time** and type **12 PM**. The blue bar on the timeline runs from noon to 8:30, half an hour after the last customers arrive.

4 Click **Add Break**. A half-hour break lands in the middle of the shift. Click its **Start Time** and type **1:30 PM**, then its **End Time** and type **2 PM**: a half-hour break after the noon rush.

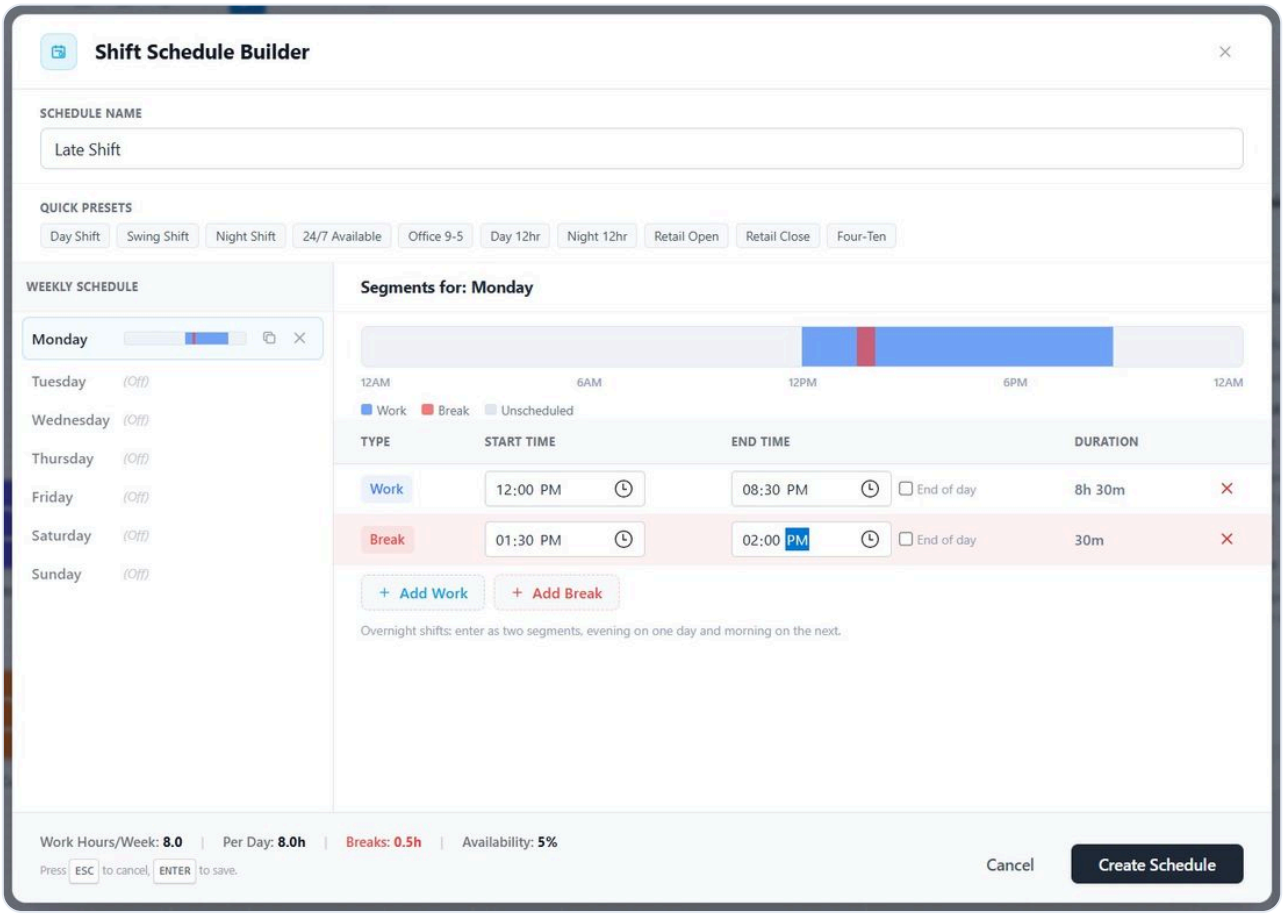



Figure 2. The late shift: noon to 8:30, with a break from 1:30 to 2.

> **What you should see:** a blue work bar with a red break inside it, and a foot that adds up the hours: 8 hours of work, half an hour of break.

5 Click **Create Schedule**.

**Create Schedule saves the schedule to the model; it does not yet give it to Late Technician.** So:

6 Open the **Shift Schedule** list and choose **Late Shift**. Schedules you make are at the end of the list, after Always Available and the built-in presets.

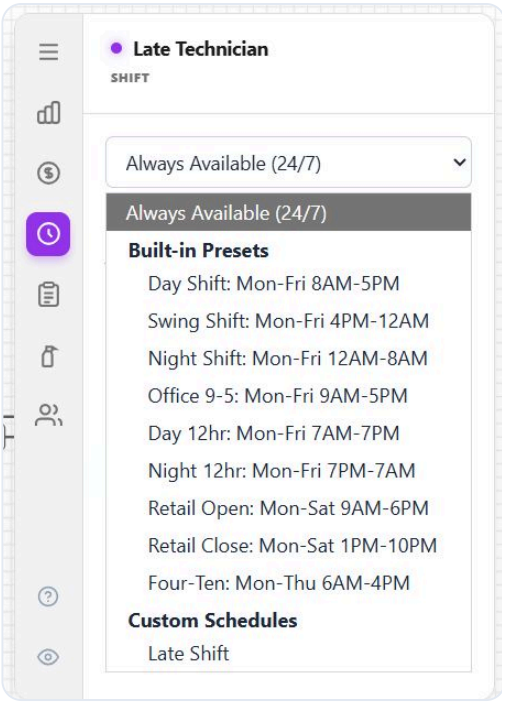



Figure 3. The Shift Schedule list: your Late Shift is at the end, under Custom Schedules.

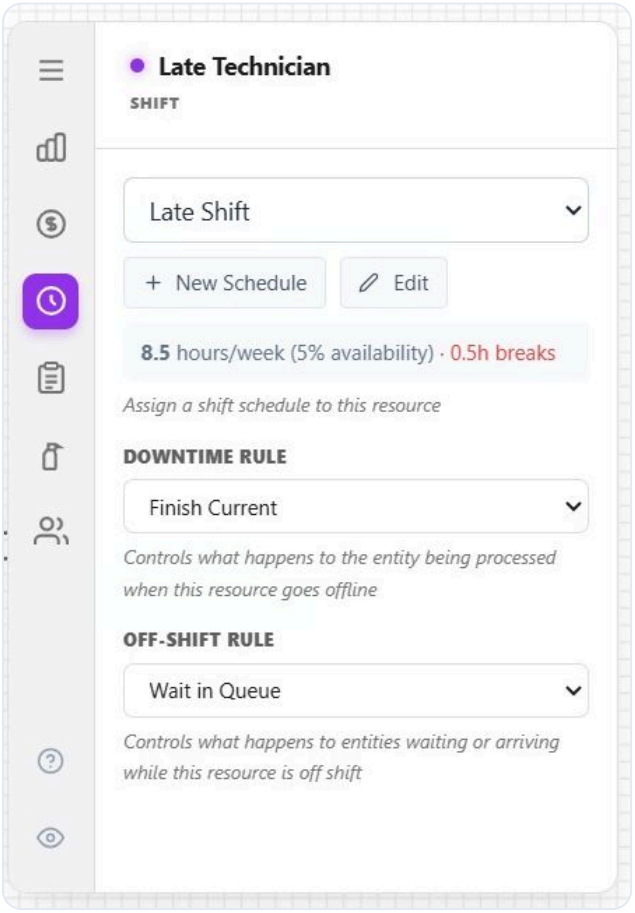



Figure 4. Late Technician now works the Late Shift.

**What you should see:** Shift Schedule reads **Late Shift**, with a summary of its hours and breaks under it.

Step 4.

## Turn the Technician into the Early Technician

1. Click the **Technician** and open its **General** tab.
2. Name it **Early Technician**, and set **Quantity** to **1**: one person on each shift now.
3. Click the **Shift** tab, then **New Schedule**. Type **Early Shift**.
4. Click **Add Work**. Click **End Time** and type **3 PM**, then **Start Time** and type **7 AM**.
5. Click **Add Break**. This time leave it where it lands: in the middle of the shift, **11:00 to 11:30**, an early lunch before the noon rush.

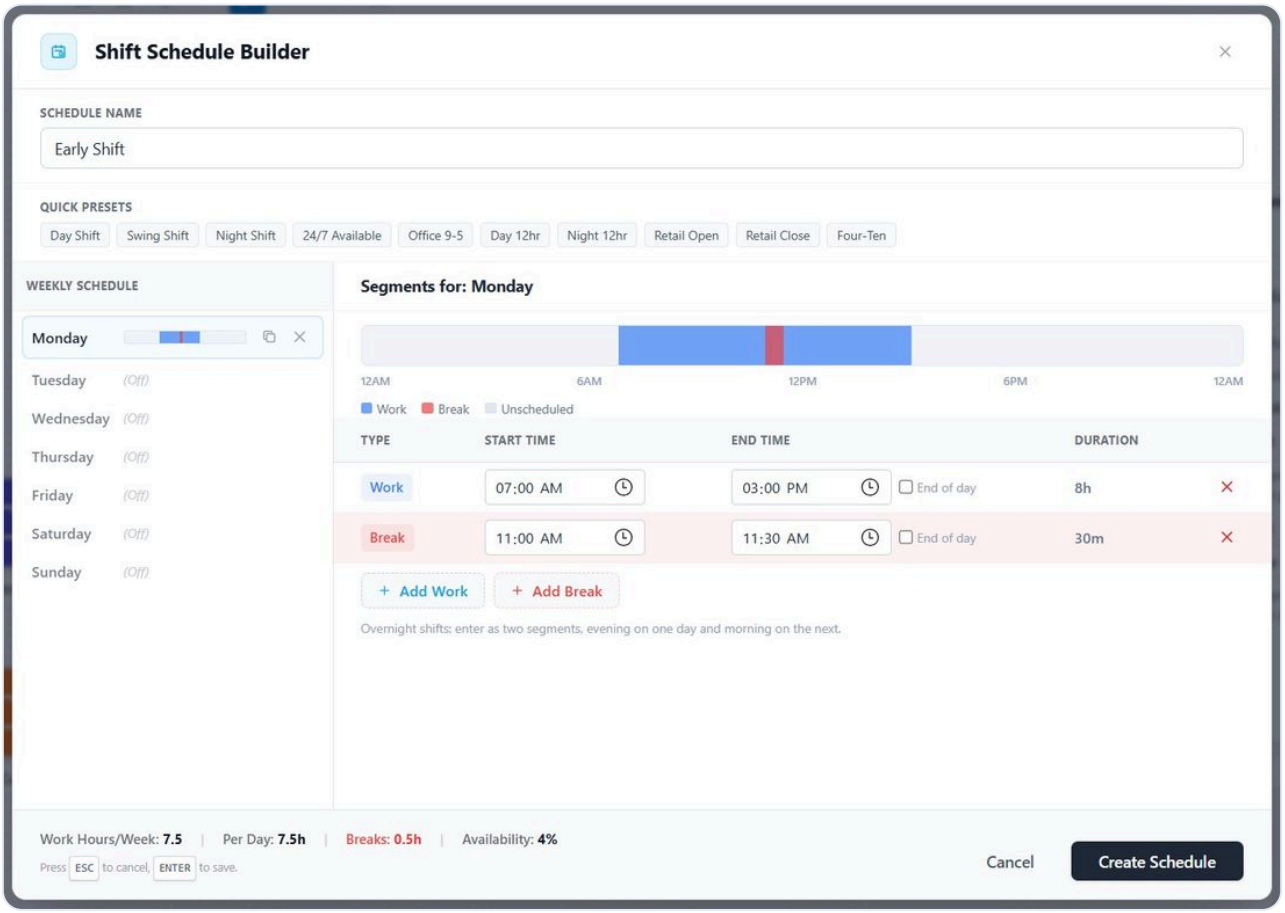



Figure 5. The early shift: 7 to 3, with a break from 11 to 11:30.

6 Click **Create Schedule**, then open the **Shift Schedule** list and choose **Early Shift**.

Step 5.

## Make the two Technicians one group

Quick Fix and Repair are joined to Early Technician; Late Technician has no lines. A group lets both steps use either one.

1 With Early Technician selected, click the **Group** tab, then **Open Resource Group Builder**.

2 Early Technician is already in the **Main Requirement**, as its owner. Click **Add Resource** and choose **Late Technician**.

3 Read the **Downtime Rule** at the top: **Finish Current**. When a shift ends or a break starts, a Technician busy with a customer finishes that customer first, then goes. The list holds three more rules: **Preempt & Wait** stops at the shift boundary and resumes the same entity when the resource returns on shift; **Preempt & Pass** hands the work to an alternate resource if one is connected; and **Clear Queue First** keeps working until everything already waiting has been processed. The Resource Shift help explains each.

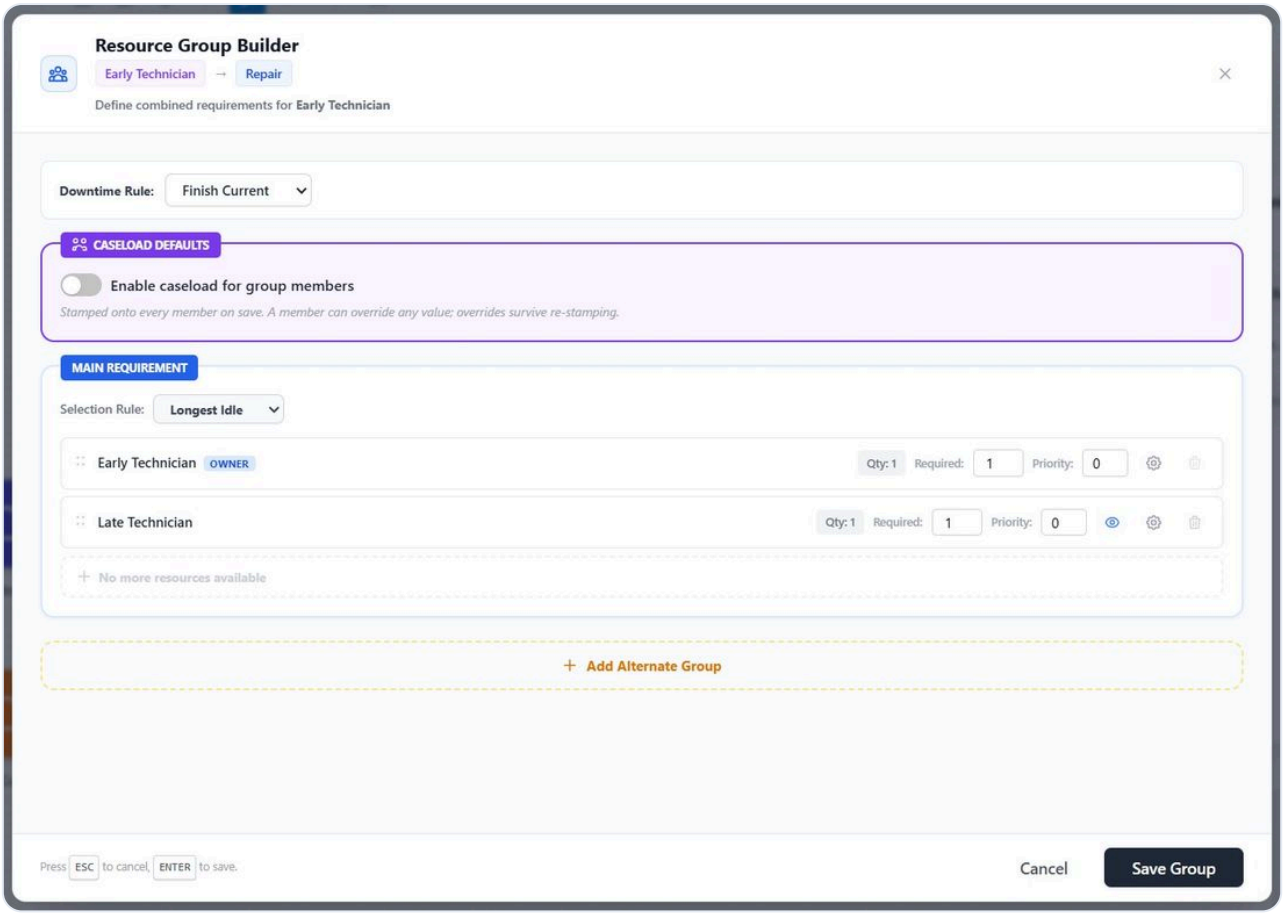



Figure 6. One group: Early Technician and Late Technician, Finish Current.

4 Click **Save Group**.

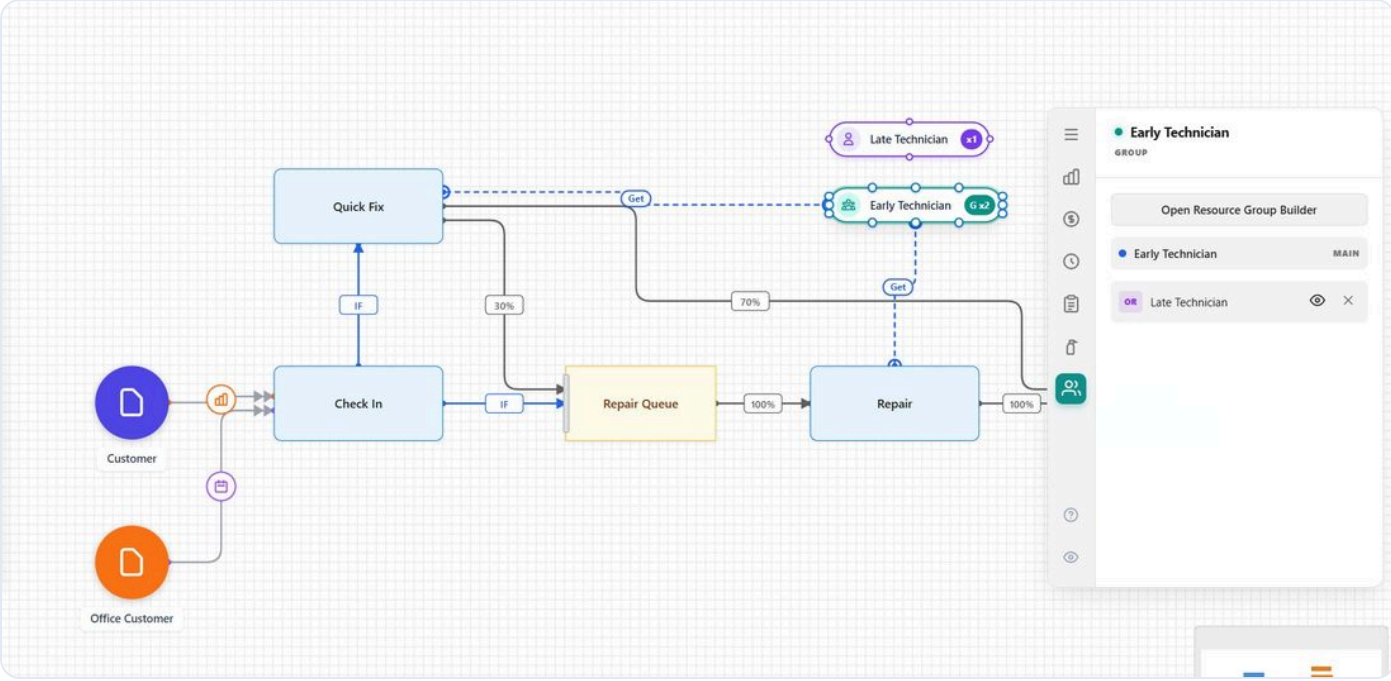



Figure 7. The badge reads G x2: a group of two.

> **What you should see:** the badge on Early Technician reads **G x2**, a group of two. Quick Fix and Repair ask for Early Technician; as a group, whichever Technician is on shift and free answers.

Step 6.

## Watch the shifts change

1. Click **Run Simulation**. Rest the pointer on **Early Technician** as the day goes; its popup tells you its state.

During the run, each Technician's badge has a color: **orange** in use, **green** idle, **blue** on break, **gray** off shift.

- Before 7, Early Technician's popup reads **Off Shift**. At 7 its shift starts and the first customer comes.
- Until noon, Early Technician is the only one on. At 10 the office sends twelve, the Repair Queue fills, and Check In's line grows long.
- At **11**, Early Technician's break is due, but it is busy with a customer. Under Finish Current it finishes that customer first, then takes its break, for the full half hour. Nobody else is on, so the line at Check In stays long.

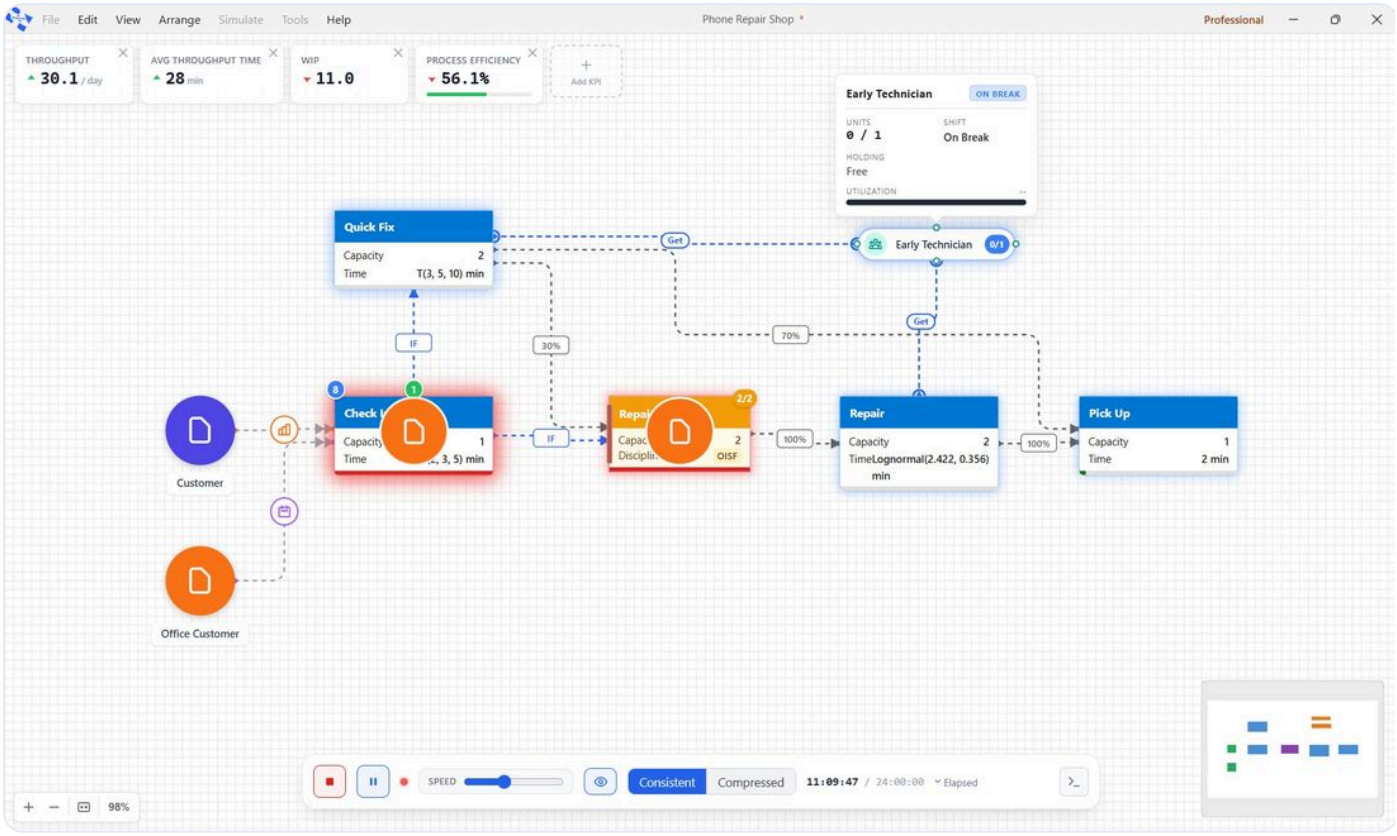



Figure 8. 11:09: Early Technician on break, with the line still long and nobody else on shift.

- At noon, Late Technician starts, and the two work through the lunchtime rush. At 1:30 Late Technician's break waits for the customer in hand too, then runs its full half hour, while Early Technician covers the shop.
- At **3**, Early Technician's shift ends: its badge turns gray and its popup reads **Off Shift**. Late Technician carries the shop alone through the evening, and goes off shift at 8:30.

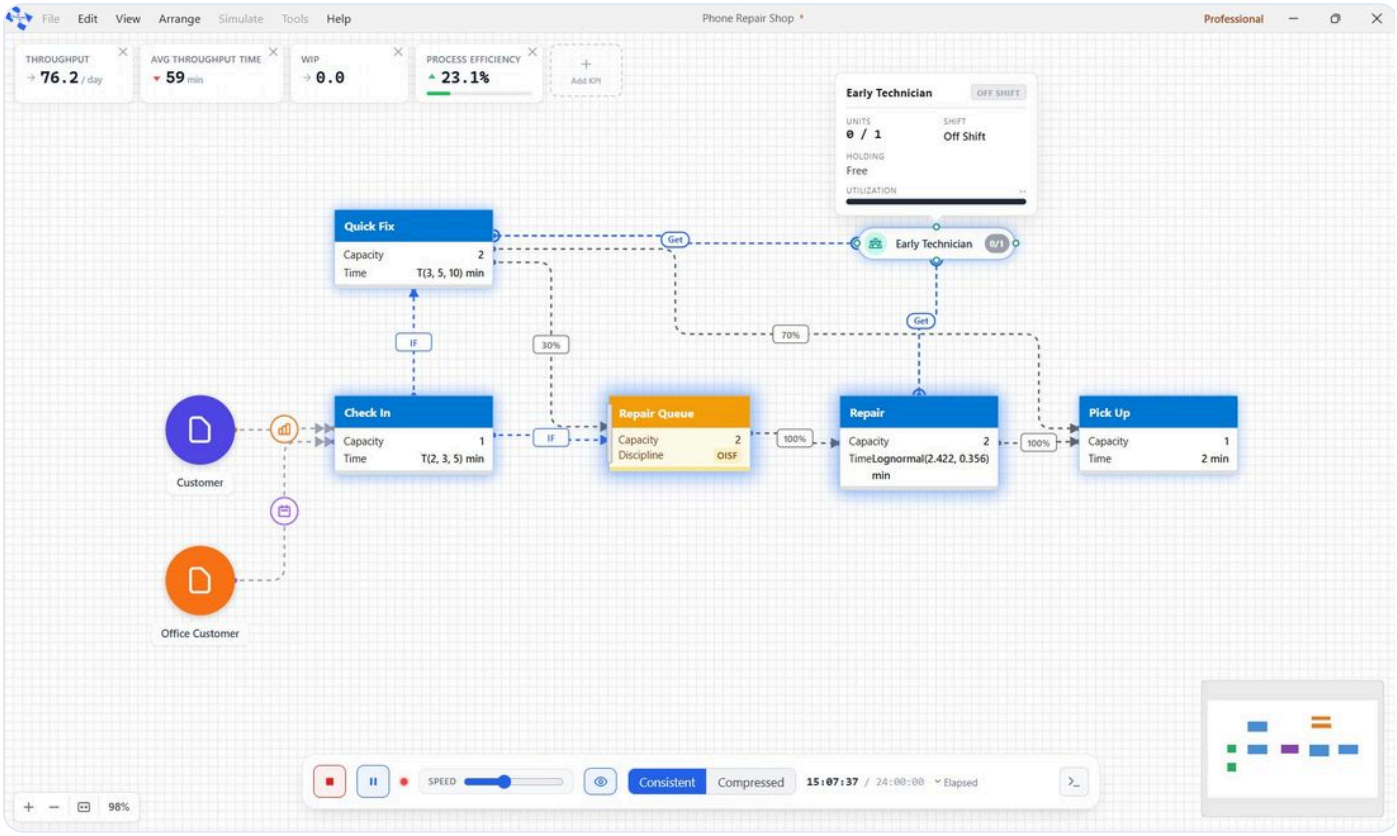



Figure 9. After 3: Early Technician is off shift, and Late Technician works alone.

Step 7.

## Read the cards

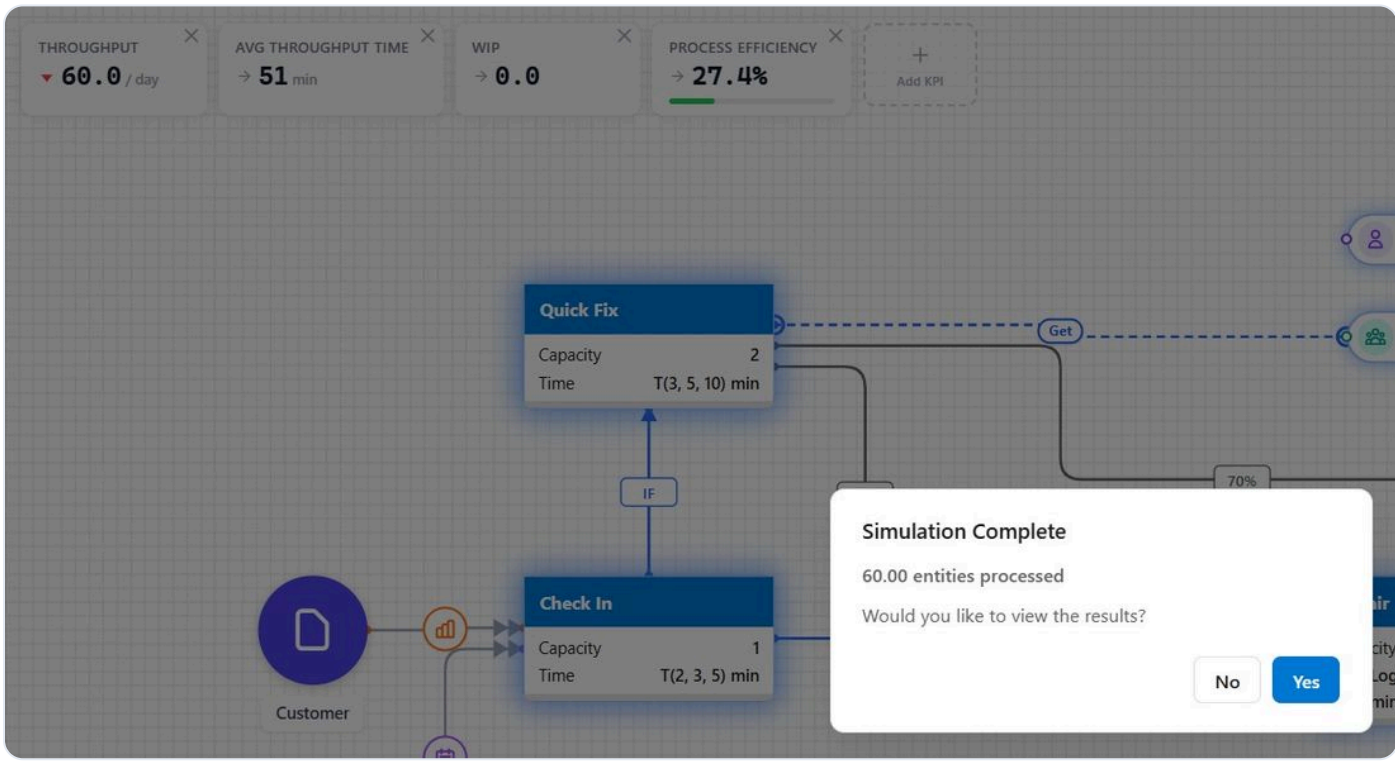



Figure 10. One Monday with real hours: 60 customers, 51 minutes in the shop, 27.4% efficiency.

- **60** went through, as in Chapter 6.
- **Avg Throughput Time, 51 minutes**, up from 24. With one Technician until noon, and a lunch break while the line is long, customers wait much longer.
- **WIP, 0**: at midnight the shop is empty.
- **Process Efficiency, 27.4%**, down from 59.8%: most of each customer's time is now spent waiting.

The shop did not get worse. Your model got **honest**. Chapter 6's shop had two people on around the clock, which no real shop has; this one has the hours a real shop would staff. Its numbers are the ones to plan with. As always, one run is one possible Monday. Click **No**.

Step 8.

## Save it

1. Click **File**, then **Save**. The star leaves the title bar. Chapter 8 starts from this shop.

## Your practice model

START FROM

**Chapter 6. Give Each Line a Size and an Order**

The shop as Chapter 6 left it. Use your own saved Phone Repair Shop instead if you built it.

COMPARE WITH

**Chapter 7. Give Your Technicians Real Working Hours**

The shop as this chapter leaves it. Open it when you are done, and check your settings against it.

Both are in **File** ▸ **Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**You click Create Schedule in the Shift Schedule Builder. Is Late Technician now on the Late Shift?**

A Yes: creating a schedule gives it to the resource

B No: Create Schedule saves it to the model; you then choose it in the Shift Schedule list

C Only after the model is saved

The answer is B. Creating a schedule adds it to the model's list of schedules. Each resource then picks its own from the Shift Schedule list.

QUESTION 2

**At 11, Early Technician's break is due, but it is in the middle of a repair. With the Downtime Rule Finish Current, what happens?**

A It stops the repair, and the customer waits for it to come back

B It finishes the repair first, then takes its full half-hour break

C It skips the break

The answer is B. Finish Current lets the customer in hand be finished before the break or the shift end; the break then runs its full length.

QUESTION 3

**Why are the two Technicians two resources joined in a group, rather than one resource with Quantity 2?**

A Because a resource has one shift schedule for all its units, and these two work different hours

B Because a group is faster

C Because a resource cannot have a Quantity of 2

The answer is A. Different hours need different schedules, so they must be two resources. The group lets Quick Fix and Repair use whichever is on shift.

## Try it yourself

Can a better break time help? Move the early lunch away from the office group.

1. Click **Early Technician**, open the **Shift** tab, and edit **Early Shift** (the **Edit** button beside its name). Move the break to **2:00 PM to 2:30 PM**, after both rushes, and save the schedule.
2. Run the day and read the Avg Throughput Time card before you answer Simulation Complete. It may show two figures, one for each kind of customer.

**What to expect.** Expect Avg Throughput Time to come down from 51 minutes. In one run of the practice model, the card showed 28 minutes for Customer and 94 for Office Customer: over all the customers that is 41 minutes, below the 51 of this chapter's run, though the office group still waits longest. At 11 the line from the office group is still long; a break then leaves nobody on shift to clear it. At 2, Late Technician is on, so one Technician still works while the other rests. The same half hour of rest costs less when it falls where someone can cover it. Set the break back to **11:00 to 11:30** and save before Chapter 8.

CHAPTER 8 OF 15

# Chapter 8. Tag Each Customer and Count the Jobs

By the end of this chapter you can

**Tag each customer with an attribute and count jobs with a variable and Action Logic.**

Watch the video (7 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 8. Tag Each Customer and Count the Jobs**
www.youtube.com/watch?v=k4QefRtLKcg

## Before you start

So far every walk-in customer was alike, and nothing in the model kept count of anything. A real shop knows more: what kind of phone each customer brought, and how many jobs are done today. This chapter gives the model that memory, with three tools that work together in almost every serious model:

- An **attribute** travels with each customer. Every customer carries its own value, so two customers can hold different ones. Here: each customer's **phone type**, Newer or Older. Attribute names start with **a**_.
- A **variable** is one value for the whole model, shared by everything in it. Here: **the number of jobs done**. Variable names start with **v**_.
- **Action Logic** is a short set of instructions that runs on each customer as it passes a step. It is how you set an attribute, raise a count, and act when something happens.

A name must be declared before logic can use it. ProcessModel never makes one up: a name used in logic that has not been declared is refused. So you declare the attribute and the variable first, then write the logic.

Here is what changes in the shop:

| Part | Chapter 7 | Chapter 8 |
|---|---|---|
| Phone type | None | Check In gives each customer a phone type: 8 in 10 Newer, 2 in 10 Older |
| Older phones | Walk-ins all try Quick Fix first | Customers with an Older phone skip Quick Fix and go straight to the Repair Queue, with the office customers |

| Part | Chapter 7 | Chapter 8 |
|---|---|---|
| Job count | None | Pick Up counts each finished job, and shows a message at the 30th |
| Everything else | As Chapter 7 left it | Unchanged; no shape moves |

**You need** your Phone Repair Shop from Chapter 7 (or the practice model below) and about forty minutes.

Step 1.

## Open your shop

1. On the Welcome screen, under **Recent Models**, click **Phone Repair Shop**.

> **What you should see:** the shop as Chapter 7 left it, with two Technicians on real shifts.

Step 2.

## Declare the attribute

1. On the toolbar, click **Logic Builder**. It opens on **Entity attributes**.
2. Click **Add new**, and click in **Name**. The box shows the **a_** prefix; type **PhoneType** after it.
3. Set **Type** to **Descriptive**: its values are words, not numbers.
4. Click **Descriptor List** and type **Newer, Older**: the values a phone type can take.

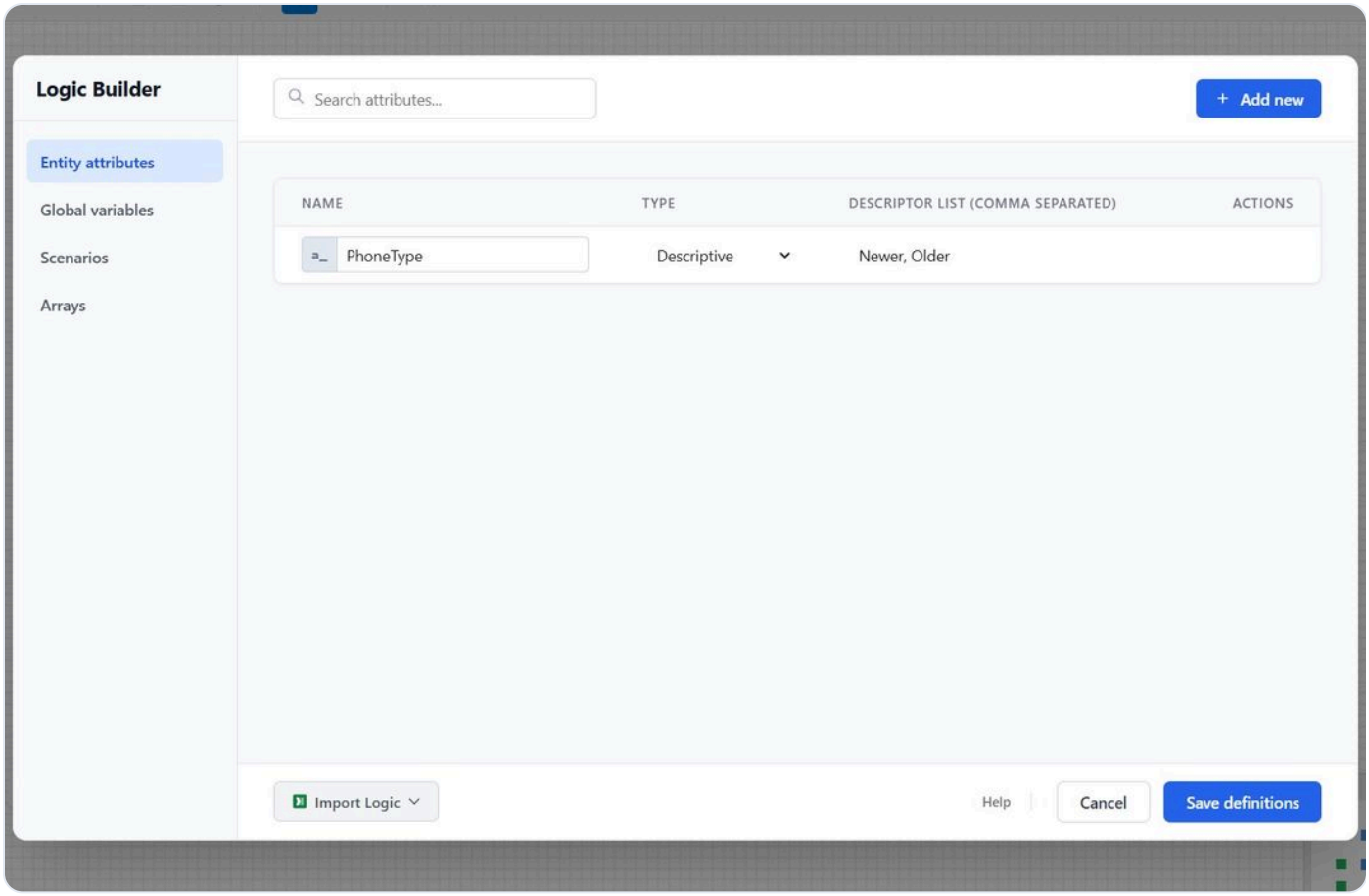



Figure 1. The attribute a_PhoneType, a word: Newer or Older.

Step 3.

## Declare the variable

1. Click **Global variables**, then **Add new**.
2. Click in **Name**, after the **v_** prefix, and type **JobsDone**. It is an **Integer**, starting at **0**.

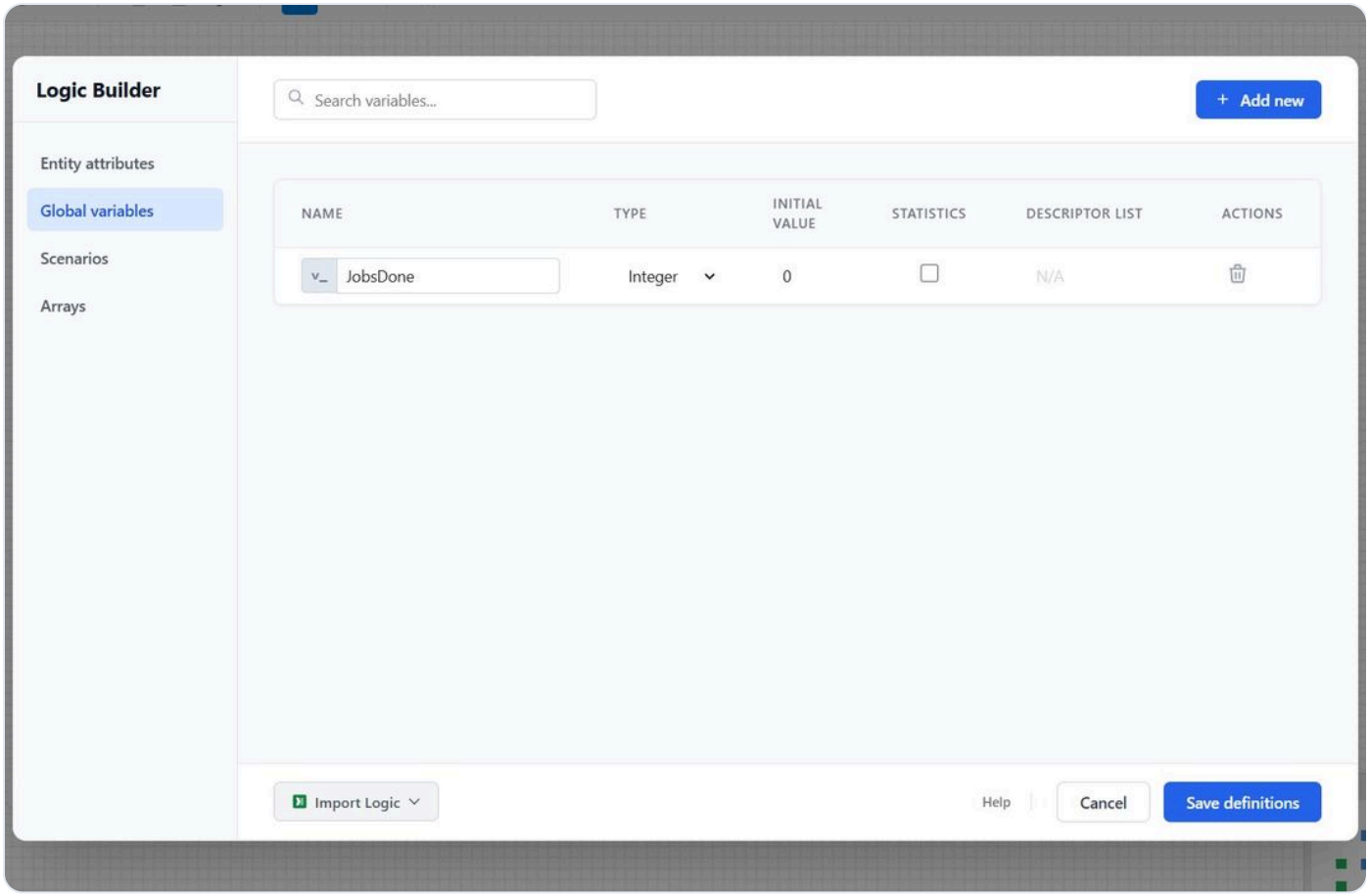



Figure 2. The variable v_JobsDone, a whole number starting at 0.

3 Click **Save definitions**. Nothing is kept until you do.

**Why one is an attribute and the other a variable:** the phone type belongs to each customer, so it must travel with them. The job count belongs to the shop, so there is only one.

---

Step 4.

## Give each customer a phone type at Check In

1 Click **Check In** and open its **Logic** tab.

2 Click **Compose**. The **Action Logic Builder** opens. It builds the logic from cards, so you do not need to remember how to write it.

3 Click **Add Action**, search for **SET Attribute**, and pick it.

4 Choose **a_PhoneType** from its list.

5 Click the value box and type **D2(80, “Newer”, 20, “Older”)**, with the words in quotes.

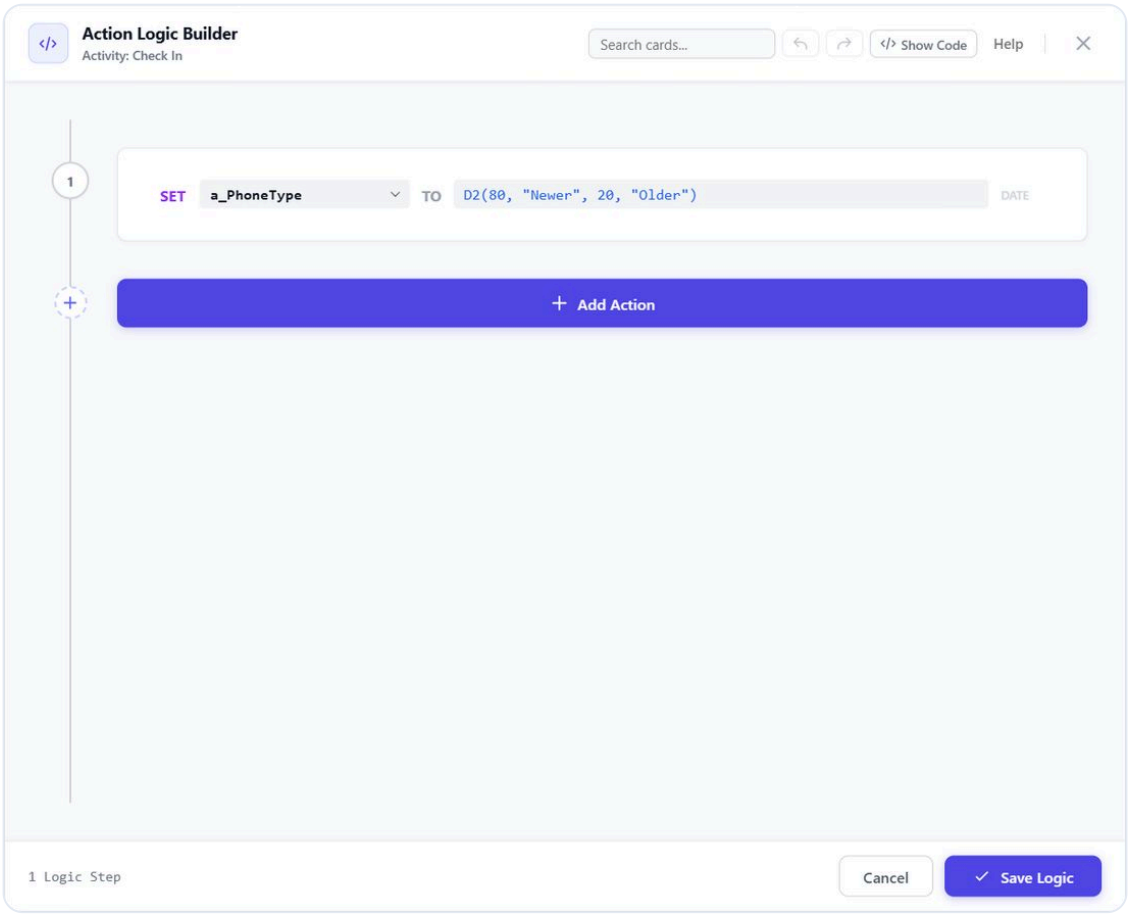



Figure 3. One card: SET a_PhoneType TO D2(80, "Newer", 20, "Older"), ready to save.

**D2** picks one of two values by percent: 80 percent of customers get “Newer” and 20 percent get “Older”. The percents must add up to 100, which the app checks as you type.

6 Click **Save Logic**. The Logic tab now shows the line the card wrote: `SET a_PhoneType TO D2(80, "Newer", 20, "Older")`.

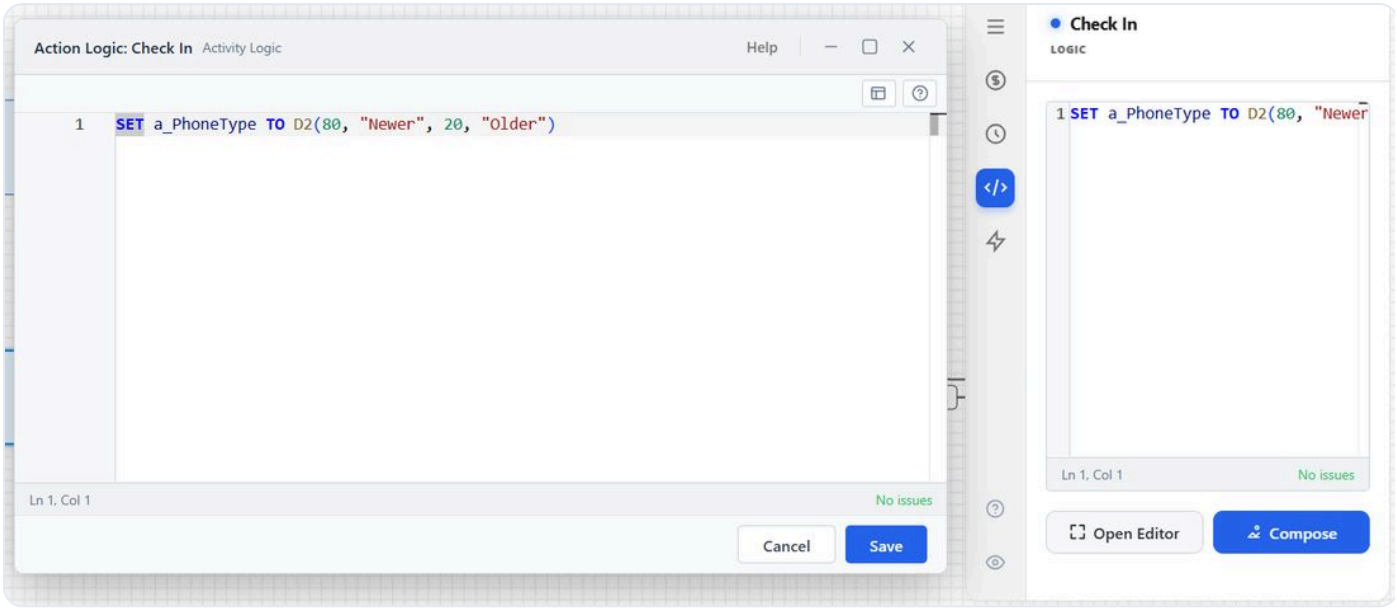



Figure 4. The line the card wrote, whole in Open Editor; the Logic tab beside it is too narrow to show all of it.

Every customer who passes Check In now carries its own phone type, drawn at random.

Step 5.

## Route older phones straight to the Repair Queue

The route from Check In to the Repair Queue already sends the office customers there by a condition. Add a second condition, so it also takes any customer with an Older phone.

1. Click the route from **Check In** to the **Repair Queue** and open its **General** tab. Its condition reads Entity Name equals Office Customer.
2. Click **Add Condition**. Set the new row's **Type** to **Attribute**, choose **a_PhoneType**, and type **Older**.
3. Click **ANY (OR)**. The line above the rows now reads **IF ANY of these are true**.

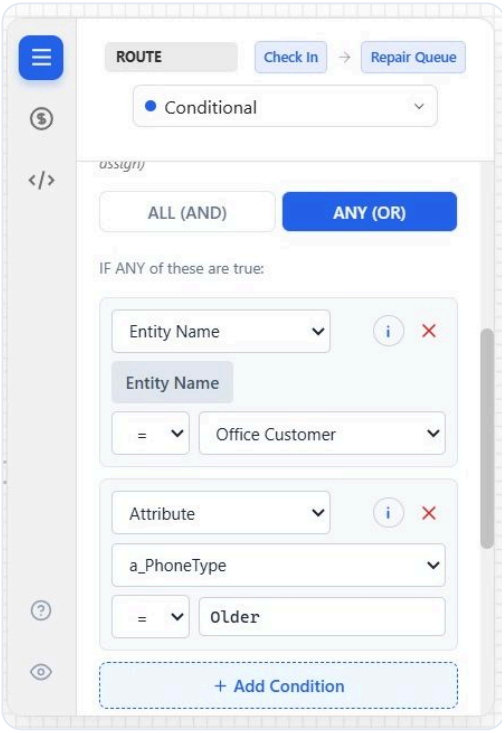



Figure 5. Two conditions joined by ANY: office customers, or any customer with an Older phone.

**ALL or ANY** matters. With **ALL (AND)**, a customer would need to be an office customer **and** have an Older phone. With **ANY (OR)**, either one is enough. This route now takes the office customers, or any customer whose phone type is Older. Everyone else still takes the catch-all to Quick Fix.

Step 6.

## Count the jobs at Pick Up

Every customer leaves through Pick Up, so it is the place to count.

1. Click **Pick Up**, open its **Logic** tab, and click **Compose**.
2. Click **Add Action**, search for **INC Variable**, and pick it. Choose **v_JobsDone**; **by** shows 1. Each customer now adds 1 to the count.
3. Click **Add Action** again, search for **If**, and pick it. For its condition, type **v_JobsDone = 30**.

4 Inside the If card, click its own **Add Action** and pick **DISPLAY**. Click the message box and type **{v_JobsDone} jobs done**. The braces put the count's live value into the message.

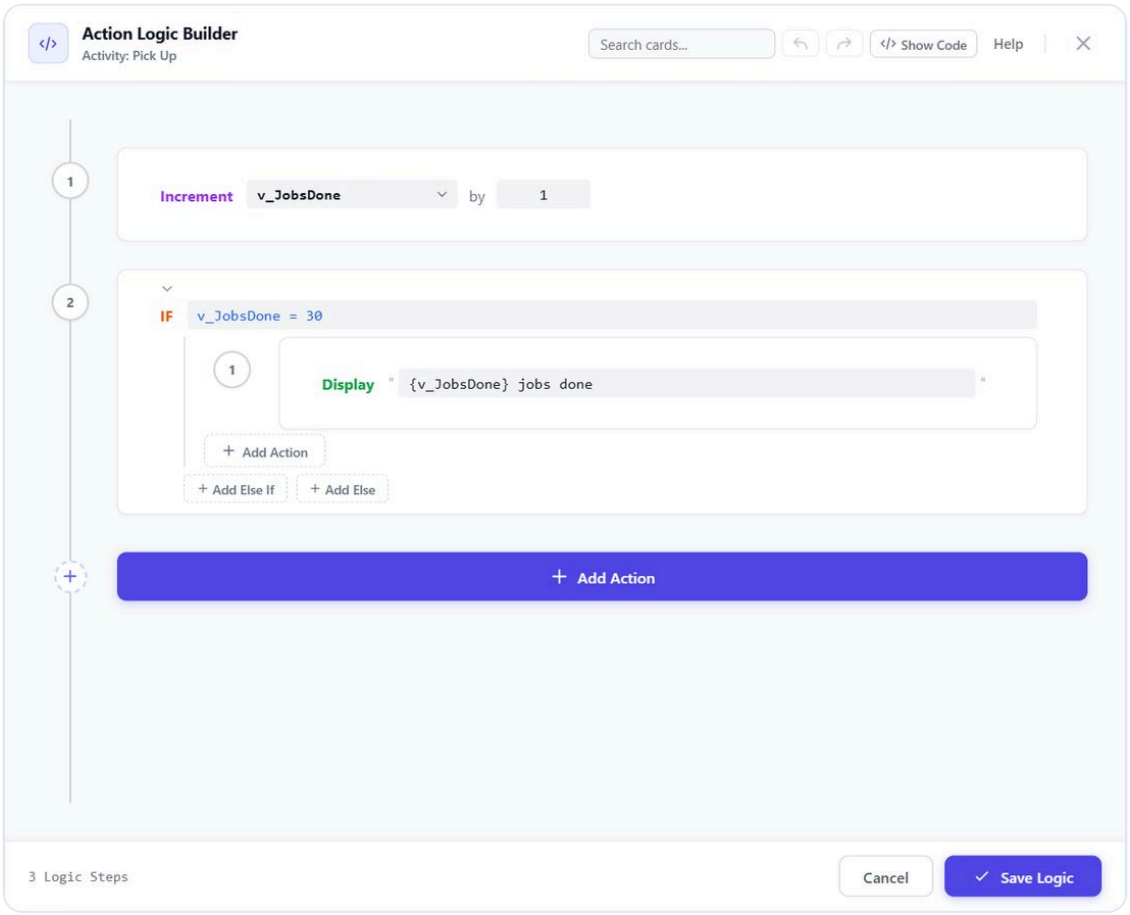



Figure 6. Count each job, and at the 30th, show a message.

5 Click **Save Logic**.




Figure 7. The four lines the cards wrote. The If adds ENDIF to close itself.

**What you should see:** four lines in the Logic tab. Reading them is worth a moment, because this is what logic looks like everywhere in ProcessModel: raise the count; if it has reached 30, show the message; end of the if.

Step 7.

## Watch the logic work

1 Click **Run Simulation**, and watch **Check In** from 7.

At 7:06, a walk-in leaves Check In straight for the Repair Queue, and the label over Check In reads **Condition, Repair Queue**. Before 10 only walk-ins arrive, so every customer on that route has an Older phone: the attribute at work.

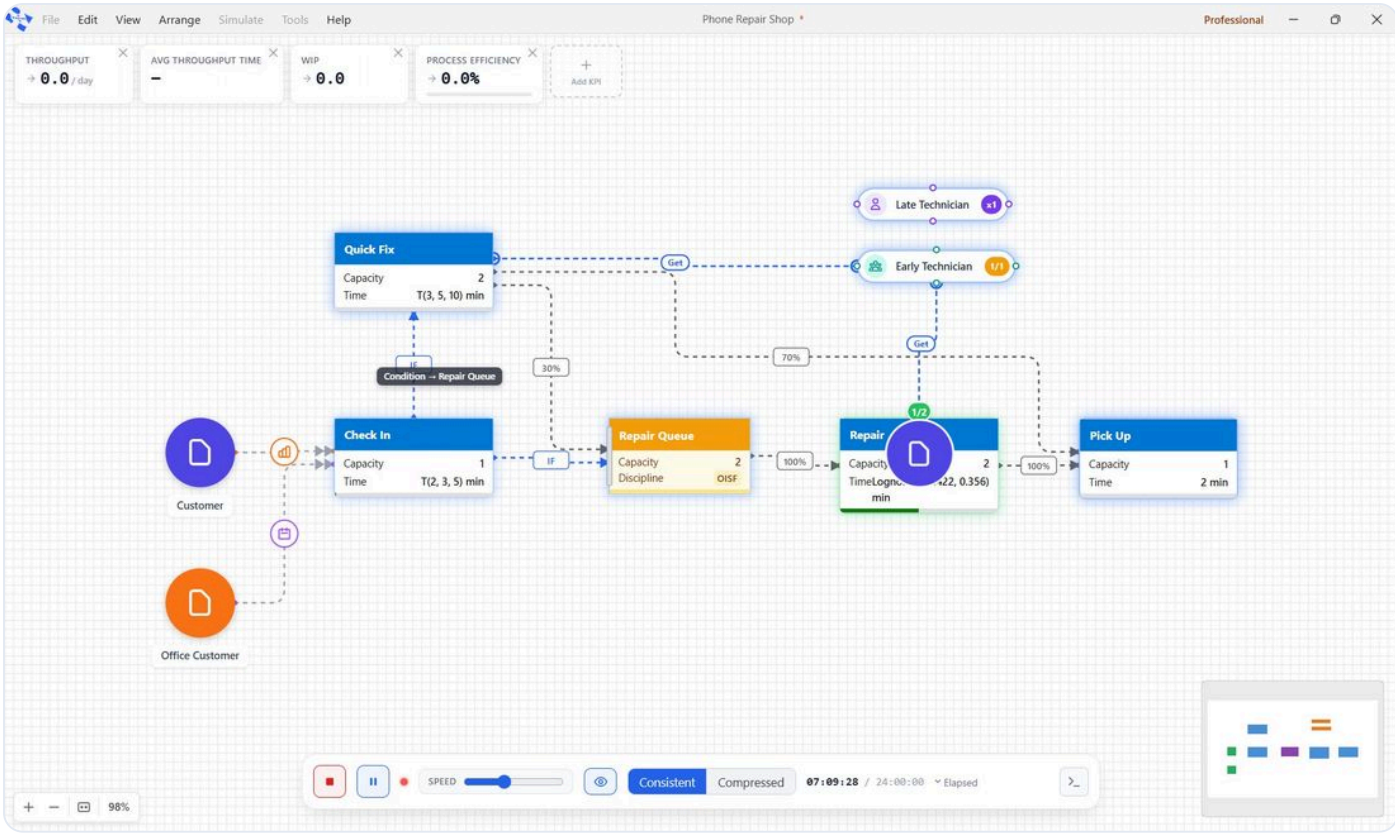



Figure 8. The first customer with an Older phone goes straight to the Repair Queue.

At 10 the office sends twelve, and they take the same route. Just before 1, the 30th customer leaves Pick Up, the count reaches 30, and the run stops on a **Simulation Message**.

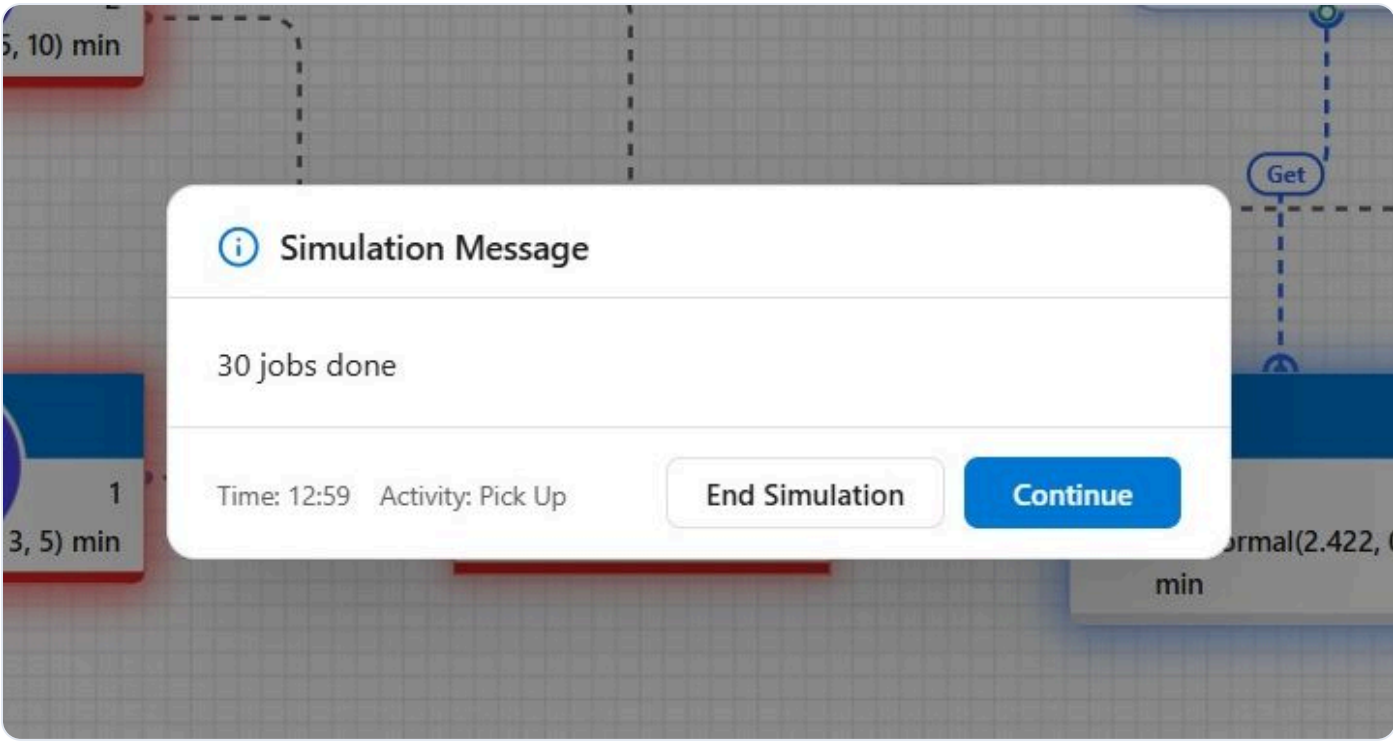



Figure 9. The message at the 30th job. Its foot names the time and the activity.

The message names the time and the step that showed it. The run holds until you click **Continue** (or **End Simulation**). Click **Continue**, and the day goes on.

Step 8.

## Read the cards

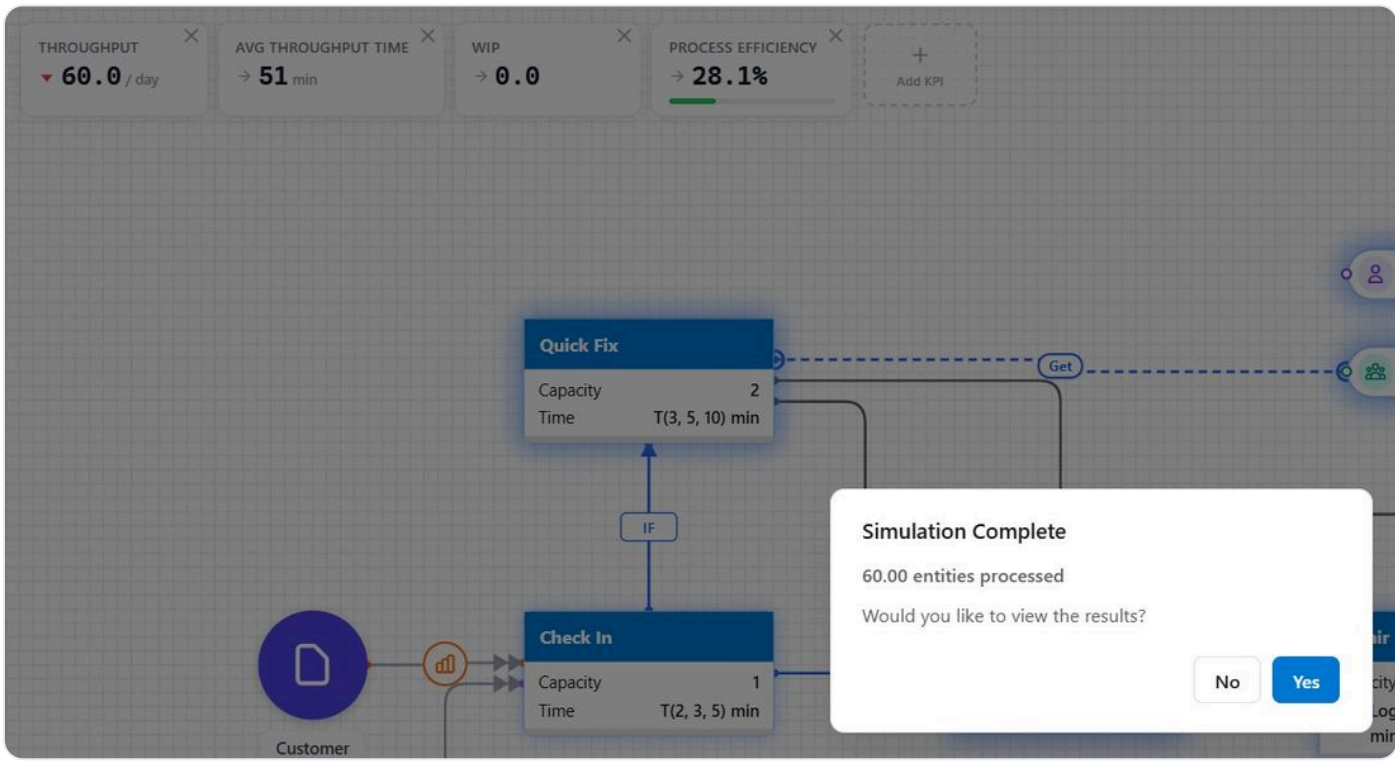



Figure 10. 60 customers, 51 minutes in the shop, 28.1% efficiency.

- **60** went through.
- **Avg Throughput Time, 51 minutes**, as in Chapter 7.
- **WIP, 0**: at midnight the shop is empty.
- **Process Efficiency, 28.1%**, up from 27.4%: a slightly larger share of each customer's time is spent being worked on.

The numbers barely moved, and that is a lesson too: this chapter changed **who goes where**, not how much work there is. The bigger gain is that the model now knows each customer's phone and the day's count, which later chapters build on. One run, one possible Monday. Click **No**.

Step 9.

## Save it

1. Click **File**, then **Save**. The star leaves the title bar. Chapter 9 starts from this shop.

## Your practice model

START FROM

**Chapter 7. Give Your Technicians Real Working Hours**

The shop as Chapter 7 left it. Use your own saved Phone Repair Shop instead if you built it.

COMPARE WITH

**Chapter 8. Tag Each Customer and Count the Jobs**

The shop as this chapter leaves it. Open it when you are done, and check your settings against it.

Both are in **File** ▸ **Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**Why is the phone type an attribute, while the job count is a variable?**

A Attributes hold words and variables hold numbers

B Each customer has its own phone type, but the shop has one count shared by all

C Variables can only be used at Pick Up

The answer is B. An attribute travels with each entity, so every customer can differ. A variable is one value for the whole model.

QUESTION 2

**The route's two conditions are joined by ANY (OR). Which customers take it?**

A Only office customers who also have an Older phone

B Office customers, and any customer with an Older phone

C Every customer

The answer is B. ANY means one true condition is enough. With ALL (AND), a customer would need both.

QUESTION 3

**You write logic that uses v_Backlog, but you never declared it in the Logic Builder. What happens?**

A ProcessModel creates it, starting at 0

B ProcessModel refuses the name: it must be declared first

C The logic runs but ignores that line

The answer is B. ProcessModel never makes up a name. Declare attributes and variables in the Logic Builder before logic uses them.

## Try it yourself

Count something else: how many customers with an Older phone come in a day.

1 In the **Logic Builder**, on **Global variables**, add **v_OlderPhones**, an Integer starting at 0, and click **Save definitions**.

2. On **Check In**'s Logic tab, click **Compose**, and under the SET card add an **If** with the condition **a_PhoneType = "Older"**. Inside it, add **INC Variable** for **v_OlderPhones**. Click **Save Logic**.

3. To see the count, open **Pick Up**'s Logic tab, click **Compose**, and change the DISPLAY message to **{v_JobsDone} jobs done, {v_OlderPhones} older phones**. Click **Save Logic**.

> **What to expect.** At the 30th job, the message now shows both counts. With 2 in 10 phones drawn as Older, expect the older-phone count to be about a fifth of the customers checked in by then; the exact number is one day's draw. In one run of the practice model it read 30 jobs done, 8 older phones, with 12 more customers still in the shop: 8 of 42. Office customers pass Check In too, so their phones are counted as well. Remove the extra logic and variable, or close without saving, before Chapter 9.

CHAPTER 9 OF 15

# Chapter 9. Make Each Repair Wait for Its Part

By the end of this chapter you can

**Make a step wait for parts from a shelf that reorders itself.**

Watch the video (4 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 9. Make Each Repair Wait for Its Part**
www.youtube.com/watch?v=Pq3_h8PZeqk

## Before you start

Until now, a repair could start the moment a Technician was free. A real repair also needs a **part**: a screen, a battery, a connector. Parts run out, a new box takes time to arrive, and while the shelf is empty, nothing gets repaired, however many Technicians are standing by.

This chapter teaches how to model stock that is used up and reordered:

- An **Inventory** is a stock location: a shelf of named items. It sits beside the flow; no route goes in or out of it. Logic reaches it by name.
- A **reorder rule** keeps it stocked: when the stock falls to the **Reorder Level**, it orders the **Reorder Qty**, which arrives after the **Lead Time**.
- **REQUEST** is the logic that takes stock. If the part is not there, the customer **waits** until it is.

Here is what changes in the shop:

| Part | Chapter 8 | Chapter 9 |
|---|---|---|
| Parts Shelf | None | A shelf holding one item, Part. It starts empty; a box of ten is ordered when three are left, and takes two hours to arrive |
| Repair Queue | A place to wait for a Technician | Each customer also takes one part here before Repair, and waits without one |

| Part | Chapter 8 | Chapter 9 |
|---|---|---|
| Everything else | As Chapter 8 left it | Unchanged |

**You need** your Phone Repair Shop from Chapter 8 (or the practice model below) and about thirty minutes.

Step 1.

## Open your shop

1. On the Welcome screen, under **Recent Models**, click **Phone Repair Shop**.

Step 2.

## Add the Parts Shelf

1. Rest the pointer on **More**, the double chevron in the palette. A list of advanced shapes opens.

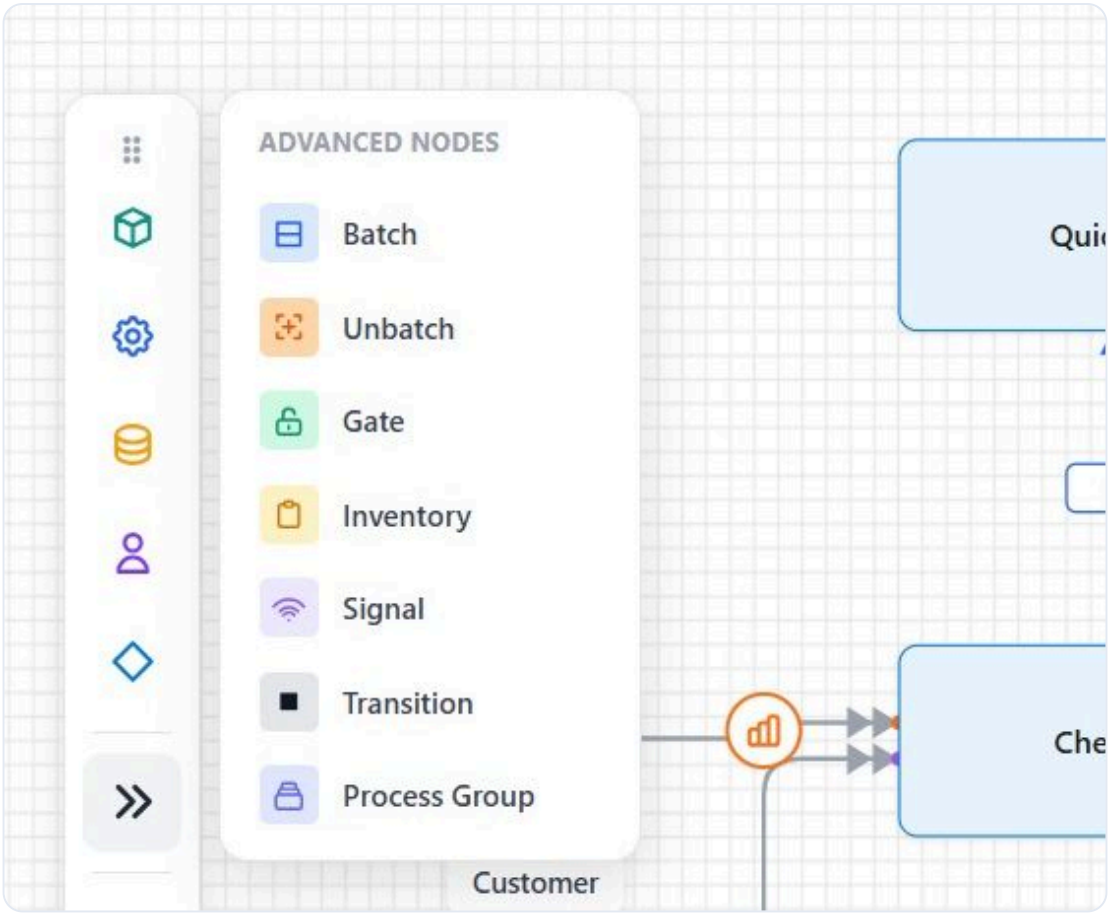



Figure 1. The advanced shapes, under More in the palette.

2. Drag an **Inventory** onto the canvas and drop it under the **Repair Queue**, clear of every line.
3. Name it **Parts Shelf**.

Step 3.

# Stock the shelf with a reorder rule

1. Click the **Items** tab, then **Add Item**.
2. Click **Name** and type **Part**. **Quantity**, the stock on the shelf when the day starts, stays at **0**: the first box is ordered at once and lands overnight.

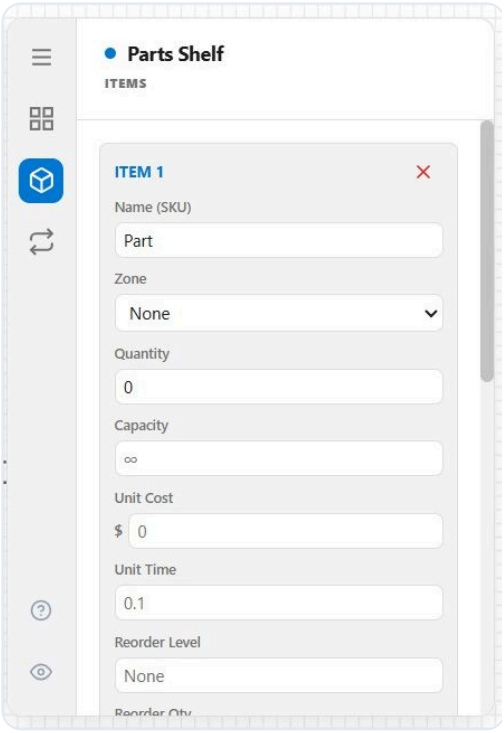



Figure 2. One item, Part, starting at 0.

3. Scroll down. Set **Lead Time Unit** to **Hr**, and type **2** in **Lead Time**: a box takes two hours to arrive.
4. In **Reorder Qty**, type **10**: a box holds ten parts.
5. In **Reorder Level**, type **3**: when three are left, the next box is ordered.

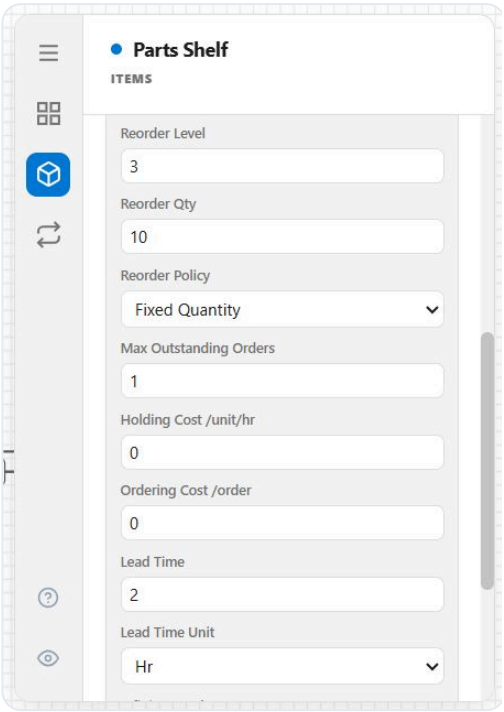



Figure 3. The reorder rule: at 3 left, order 10, which take 2 hours to arrive.

Think about what these three numbers do together. A box is ordered at three left and takes two hours. If the shop uses more than three parts in those two hours, the shelf runs dry before the box lands. That is a **stockout**, and this run will show you one.

---

Step 4.

## Make each repair take a part

Where should a customer take the part? At the **Repair Queue**, not at Repair, and the reason is worth understanding. Repair's own logic runs once a Technician has been given to the customer, so a customer waiting there for a part would keep a Technician standing idle, unable to help at Quick Fix. In the Repair Queue the customer waits for the part, and both Technicians stay free.

1. Click the **Repair Queue**, open its **Logic** tab, and click **Compose**.
2. Click **Add Action**, search for **REQUEST from**, and pick it.
3. Type **Part** for the item and **Parts Shelf** for where it comes from. The names must match the item and the shelf exactly.

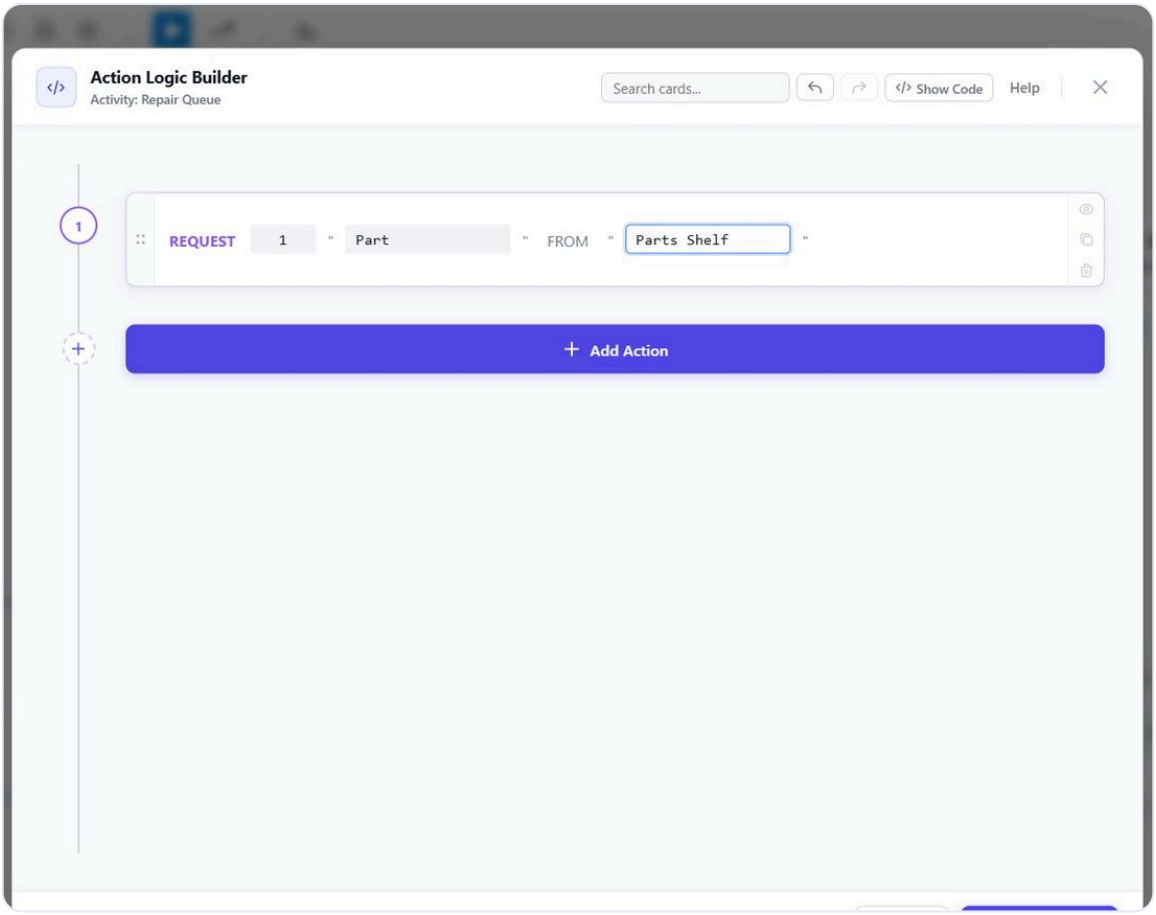



Figure 4. REQUEST 1 Part FROM Parts Shelf.

4 Click **Save Logic**.

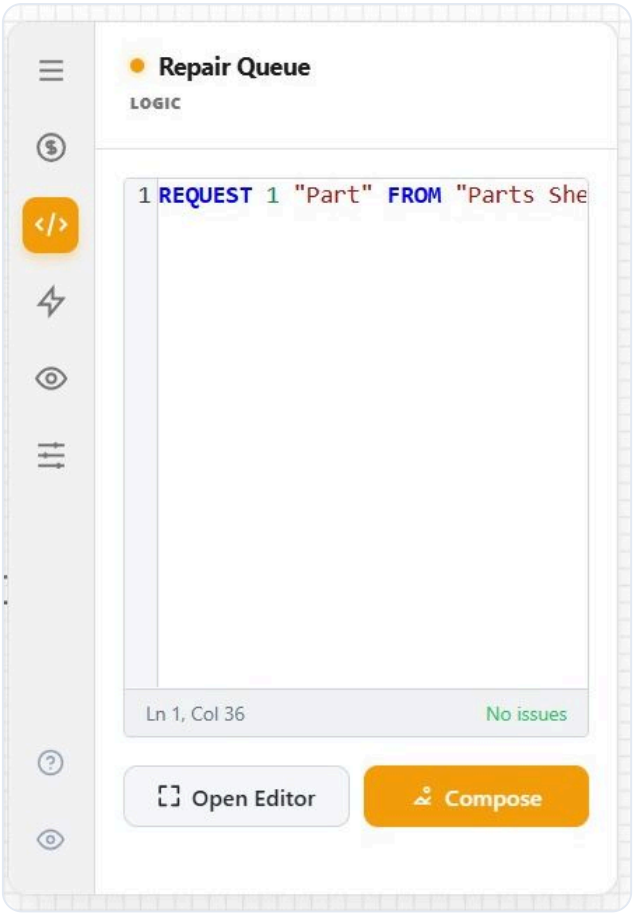



Figure 5. The line the card wrote: each customer takes one part before Repair, and waits in the queue without one.

Step 5.

## Watch the shelf

1. Click **Run Simulation** and watch the **Parts Shelf**. During the run it shows its stock as **Qty**.

Overnight, the first box lands: the shelf reads 10.

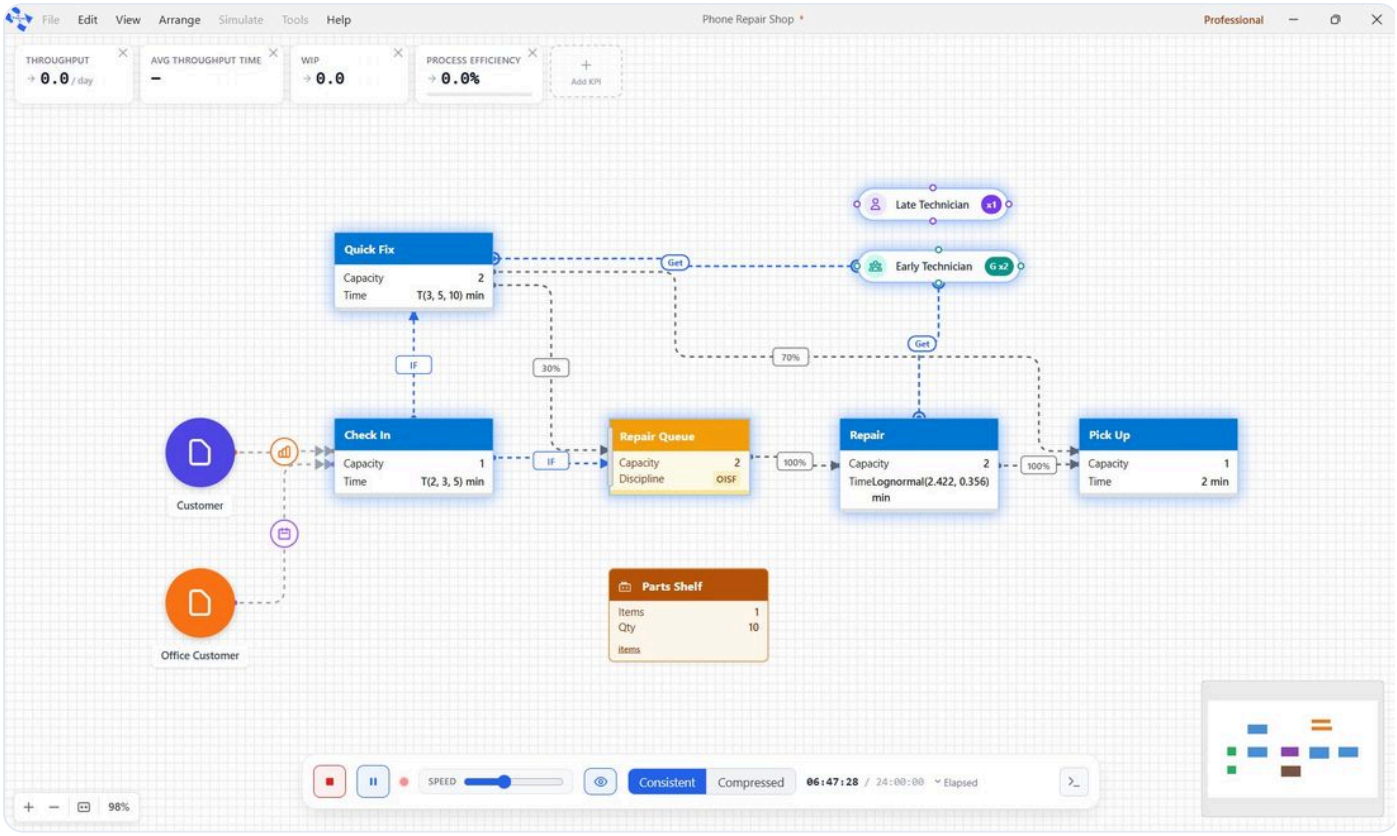



Figure 6. The first box has landed overnight: ten parts on the shelf.

From 7, each repair takes one part, and the count goes down. When three are left, an orange low mark shows and a new box is ordered. Then watch what happens when the shelf reaches **0**:

- Customers in the Repair Queue wait for a part.
- The queue fills, and, just as in Chapter 6, a full queue stops Check In.
- Repair and Quick Fix stand idle: the Technicians are free, but there is nothing they can repair.

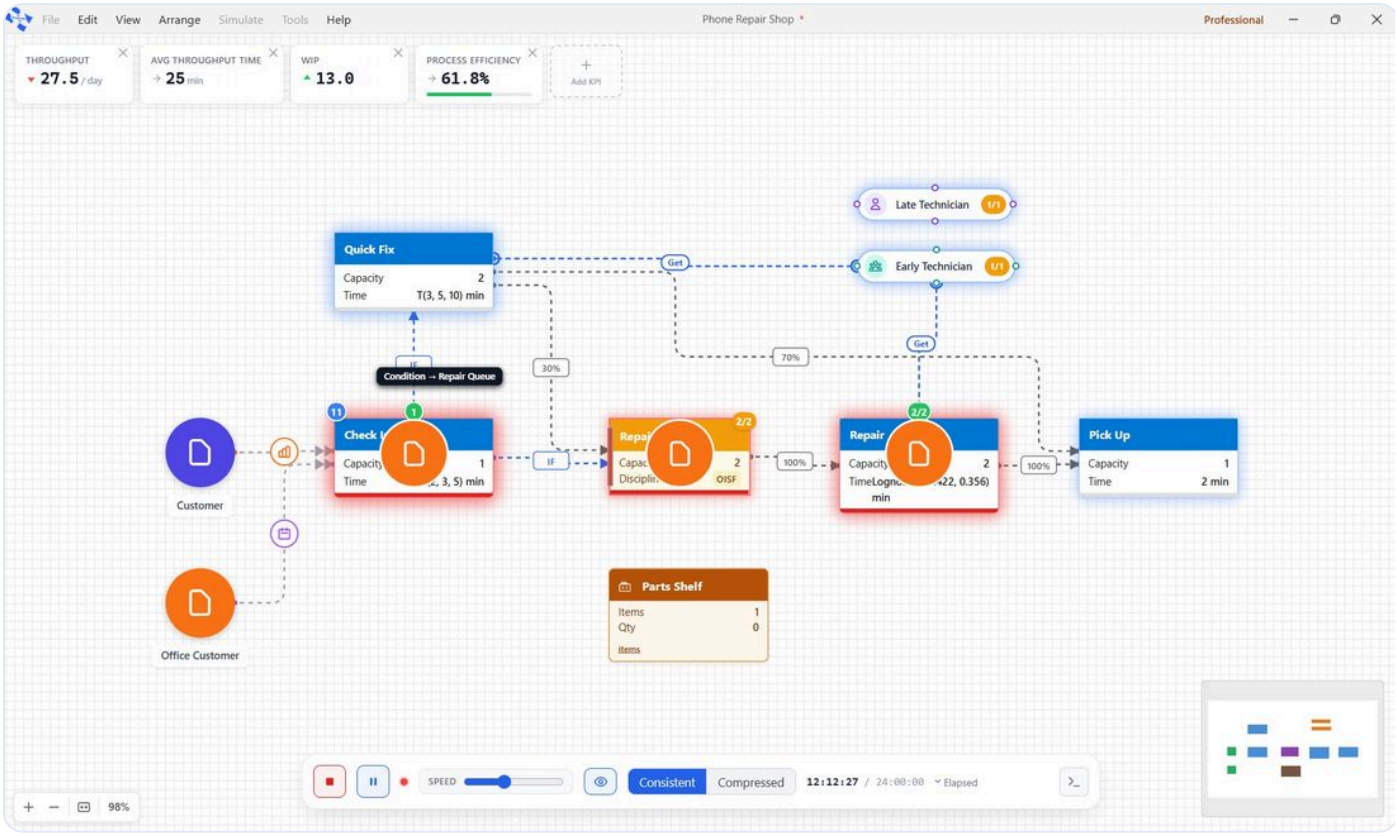



Figure 7. A stockout just after noon: no parts, a full Repair Queue, and Check In stopped.

Two hours after the order, the box lands and the repairs move on. After noon it happens again; at the 30th job, Chapter 8's message stops the run (click **Continue**); late in the day the shelf runs out once more, for less than an hour.

Step 6.

## Read the cards

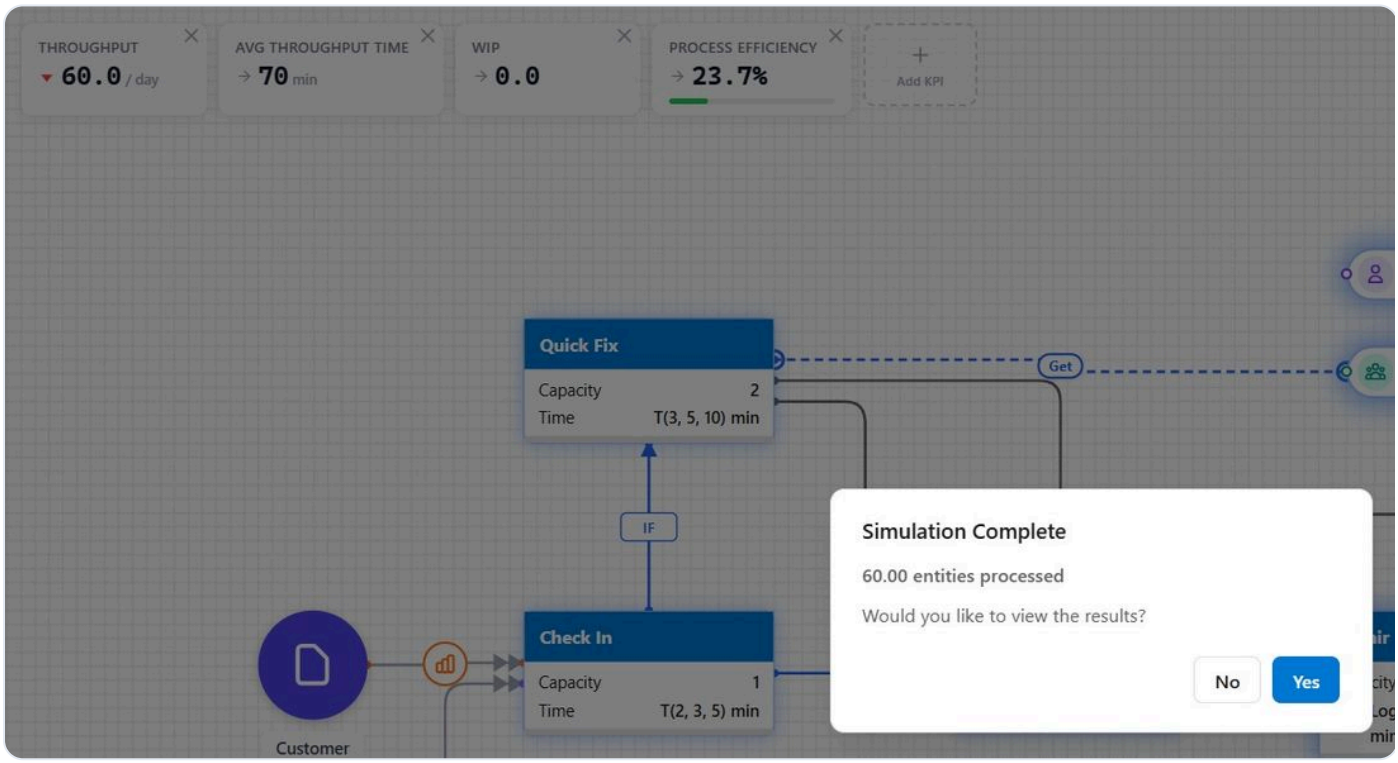



Figure 8. A Monday with parts: 60 customers, 70 minutes in the shop, 23.7% efficiency.

- **60** went through.
- **Avg Throughput Time, 70 minutes**, up from 51 in Chapter 8.
- **WIP, 0**: at midnight the shop is empty.
- **Process Efficiency, 23.7%**, down from 28.1%: more of each customer's time is spent waiting.

Nothing about the staff or the steps changed. Nineteen extra minutes per customer came from **the parts alone**: a supply problem showing up as a waiting-room problem. That is the kind of cause a model finds that a look at the shop floor would miss. One run, one possible Monday. Click **No**.

Step 7.

## Save it

1. Click **File**, then **Save**. Chapter 10 starts from this shop.

## Your practice model

START FROM

**Chapter 8. Tag Each Customer and Count the Jobs**

The shop as Chapter 8 left it. Use your own saved Phone Repair Shop instead if you built it.

COMPARE WITH

**Chapter 9. Make Each Repair Wait for Its Part**

The shop as this chapter leaves it. Open it when you are done, and check your settings against it.

Both are in **File** ▸ **Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**Why does the REQUEST go in the Repair Queue's logic and not in Repair's?**

A. Because Repair cannot have logic

B. Because in Repair a Technician would already be given to the customer and would stand idle while the part is missing

C. Because the Repair Queue is nearer the Parts Shelf

The answer is B. Repair's logic runs after its resources are granted. Waiting in the Repair Queue keeps both Technicians free for Quick Fix.

QUESTION 2

**The shelf orders a box of 10 when 3 are left, and a box takes 2 hours. When does a stockout happen?**

A Never: the box is ordered before the shelf is empty

B Whenever more than 3 parts are used in the 2 hours before the box lands

C Only when the box is late

The answer is B. The 3 left must last the whole lead time. A busy two hours uses them up before the new box arrives.

QUESTION 3

**During the stockout, the Technicians are free but Check In is stopped. Why?**

A Check In needs a part too

B The Repair Queue is full of customers waiting for parts, so the customer at Check In has nowhere to go

C The Technicians are on a break

The answer is B. A full queue downstream blocks the step before it, as in Chapter 6. Here the queue is full because of the parts, not the people.

## Try it yourself

Fix the stockouts with a smarter reorder rule, not more Technicians.

1. Click the **Parts Shelf**, open the **Items** tab, and raise **Reorder Level** from 3 to **6**, so the next box is ordered sooner.

2 Run the day and read Avg Throughput Time before you answer Simulation Complete.

**What to expect.** Ordering earlier gives the shelf more parts to last the two-hour lead time, so expect fewer or shorter stockouts and an Avg Throughput Time below 70 minutes. In one run of the practice model, the shelf ran dry twice instead of three times, and Avg Throughput Time was 59 minutes. The right reorder level is the parts you use while you wait for the next box, plus a little to spare. Set Reorder Level back to **3** and save before Chapter 10.

CHAPTER 10 OF 15

# Chapter 10. Run the Day Ten Times, Not Once

By the end of this chapter you can

**Run the day many times and read all the runs in one report.**

Watch the video (3 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 10. Run the Day Ten Times, Not Once**
www.youtube.com/watch?v=WWQrrvcuAyM

## Before you start

Every chapter so far ended the same way: “this is one run, so it is one possible Monday.” Now it is time to take that seriously. Your shop is full of chance: when customers arrive, how long each repair takes, which phones are Older. Each run draws one set of those chances, so each run is **one** Monday out of countless possible ones. A decision based on one Monday is a decision based on luck.

The cure is simple: run many Mondays, each with its own chances, and read them together. Each run is called a **replication**. Their average is far steadier than any single day, and the spread between them tells you how much a real Monday might vary.

This chapter is short and changes one number, but it explains four settings you have used without thinking about them. Here is what changes:

| Part | Chapter 9 | Chapter 10 |
|---|---|---|
| Replications | 1: one Monday | **10**: ten Mondays, each with its own random numbers |
| Run length, Warmup length, Repeatable runs | 24 hours, 0, empty | Unchanged, and now you know what each one means |
| Everything else | As Chapter 9 left it | Unchanged |

**You need** your Phone Repair Shop from Chapter 9 (or the practice model below) and about fifteen minutes.

Step 1.

## Open your shop

1 On the Welcome screen, under **Recent Models**, click **Phone Repair Shop**.

Step 2.

## Read the Run group, one setting at a time

1 On the toolbar, click **Simulation Options**. It opens on the **Run** group. Each setting has its own one-line explanation beside it; read them with this lesson.

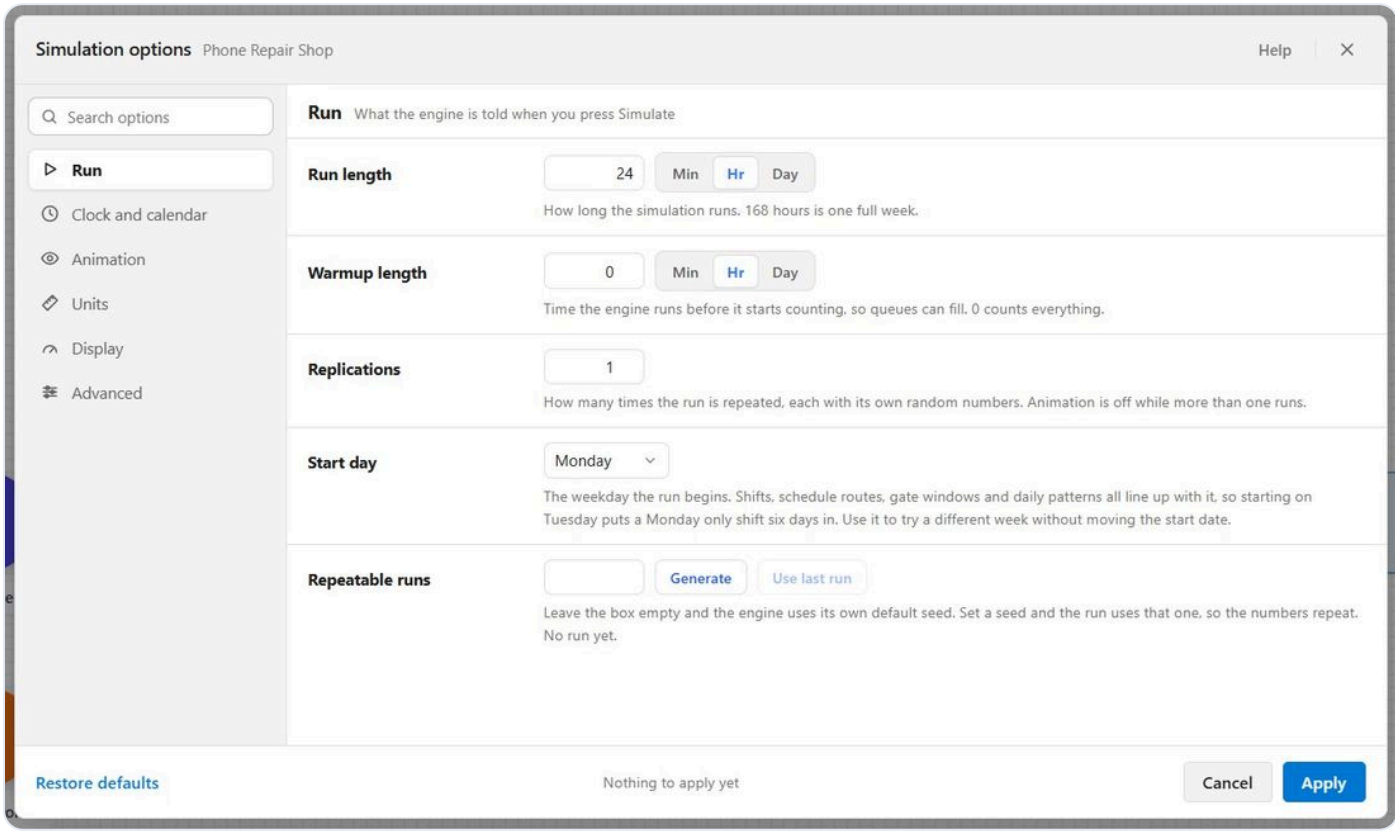



Figure 1. The Run group: four settings that decide what one press of Run Simulation means.

**Run length, 24 hours.** One Monday, from midnight to midnight. You set this in Chapter 4.

**Warmup length, 0.** A warm-up is time the model runs **before** it starts counting, so that lines can fill up to a normal level first. A process that never empties, such as a factory that runs day and night, needs one: otherwise

its first hours, starting empty, would make it look better than it is. This shop opens empty every morning, just as a real repair shop does, so the whole day should count. Its warm-up stays 0.

**Repeatable runs, empty.** This answers a question you may have wondered about: why did the same model always give exactly the same cards? Because with this box empty, ProcessModel uses its own default seed, a fixed starting point for its random numbers. Same seed, same random numbers, same results. That is useful: when you change one thing, the difference you see comes from your change, not from luck. Type your own number here, or click **Generate**, to start the random numbers from a different point.

---

Step 3.

## Run ten Mondays

1. Click **Replications** and type **10**: ten Mondays, each with its own random numbers.
2. Read the foot: **Apply will change 2 settings: Replications, Animation.** With more than one replication, nothing is drawn while they run, so they finish in seconds.

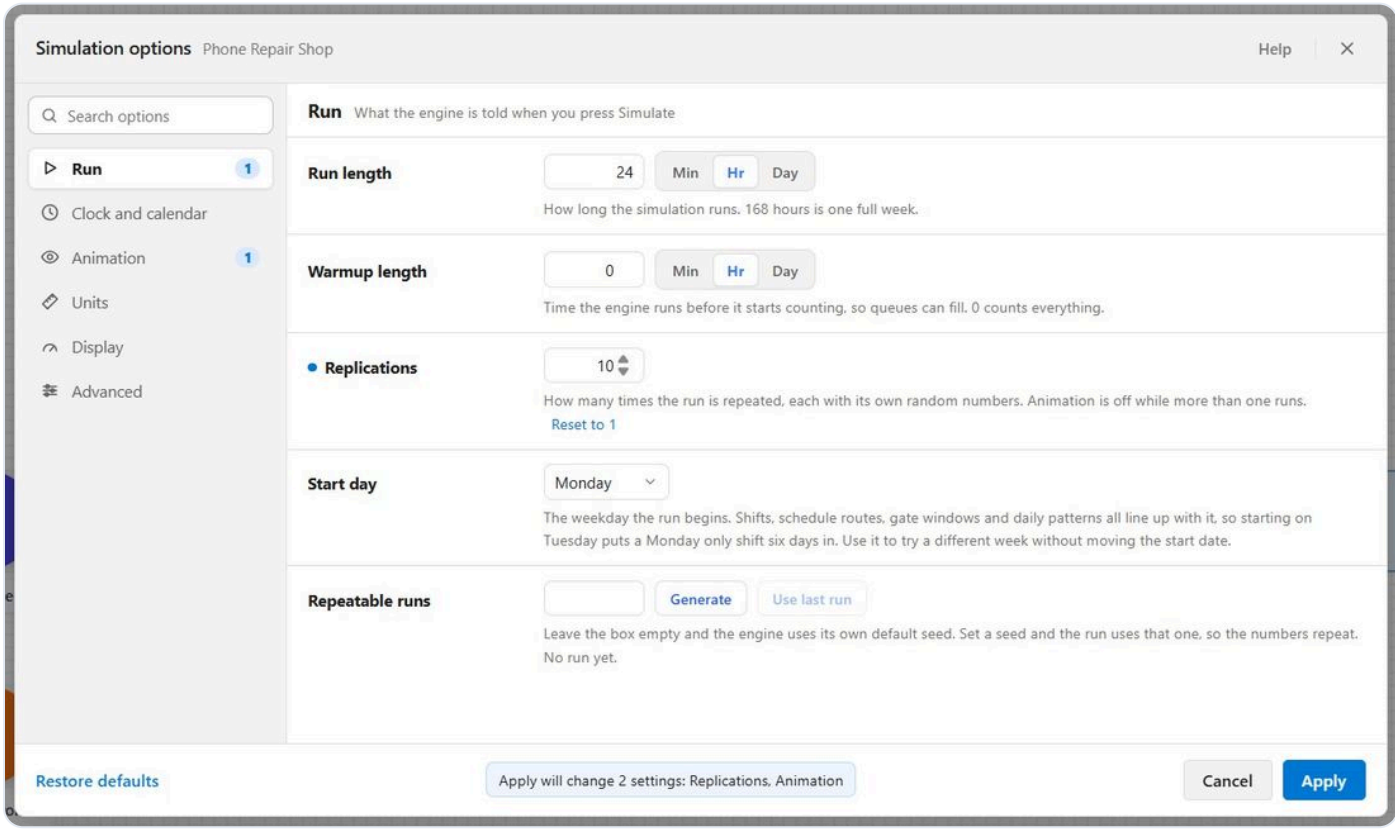



Figure 2. Replications 10. Apply also turns animation off, because nothing moves while they run.

3. Click **Apply**, then **Run Simulation**.

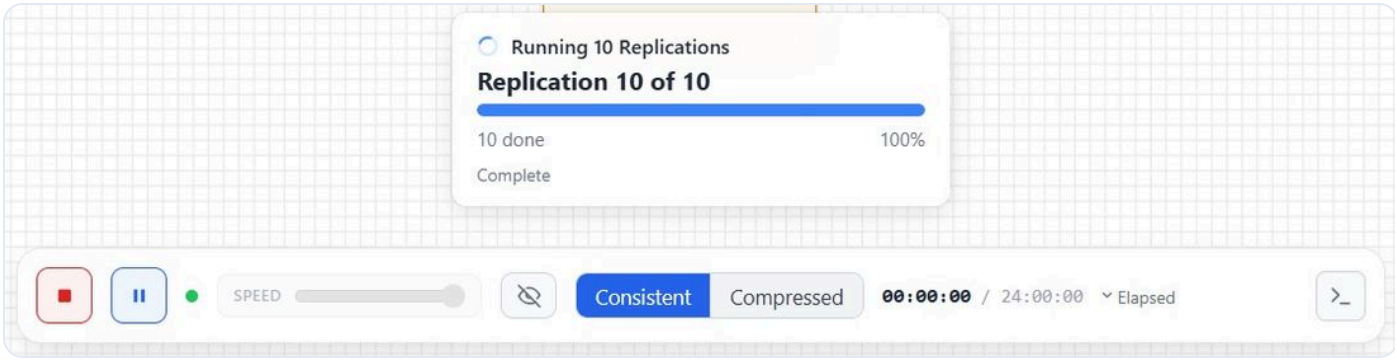



Figure 3. A card counts the ten replications as they finish.

**What you should see:** a card that counts the replications, done in a few seconds, then **Simulation Complete**.

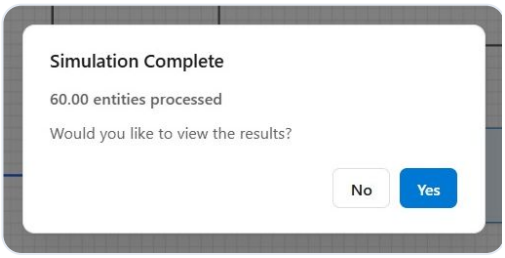



Figure 4. With several replications, the number processed is the average for one run.

With several replications, the number processed is the **average** for one run: 60 customers on an average Monday.

---

Step 4.

## Read ten Mondays in one report

Until now you answered **No** and read the live cards. With ten replications there are no live cards, and this is what the **Output Report** is for.

1. Click **Yes**.

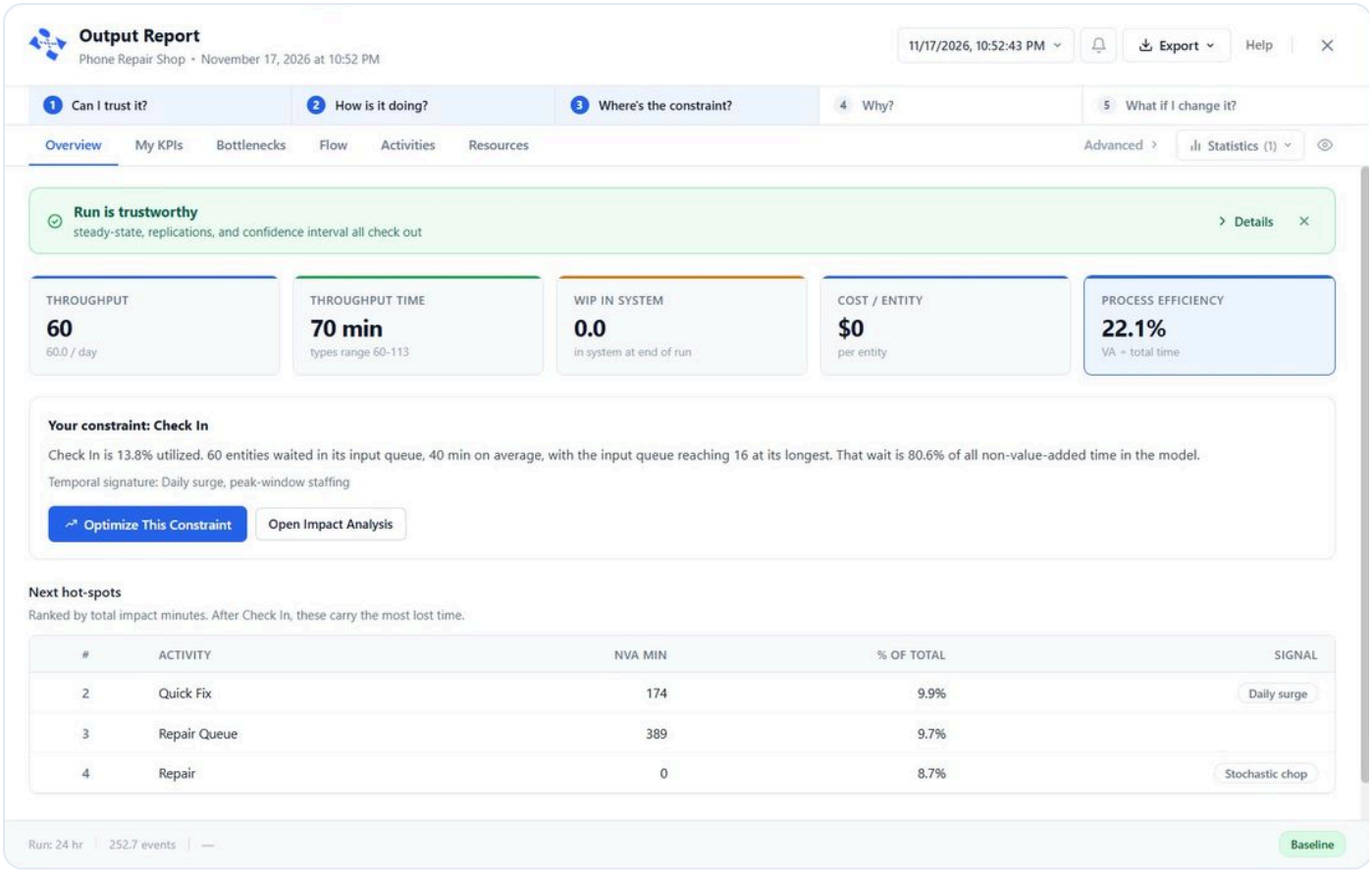



Figure 5. Ten Mondays in one report. Each figure is the average of the ten.

**What you should see:**

- The green banner: **Run is trustworthy**. The report checks whether there are enough replications and whether they agree well enough for the averages to be relied on. In Chapter 1 you saw the same green line on a demo.
- Five figures, each now the **average of ten Mondays**: **Throughput 60**, **Throughput Time 70 minutes**, **WIP 0**, **Process Efficiency 22.1%**.

Compare with Chapter 9's single run: 70 minutes and 23.7%. Throughput Time agrees; efficiency is a little lower on average than on that one Monday. The single run was not wrong, only one draw. The average of ten is the number you can plan with. Chapter 11 reads the rest of this report: where customers wait, and why.

2 Close the report with the **X** at its top right.

Step 5.

## Save it

1. Click **File**, then **Save**. Chapter 11 starts from this shop, with ten replications set.

## Your practice model

START FROM

**Chapter 9. Make Each Repair Wait for Its Part**

The shop as Chapter 9 left it. Use your own saved Phone Repair Shop instead if you built it.

COMPARE WITH

**Chapter 10. Run the Day Ten Times, Not Once**

The shop as this chapter leaves it. Open it when you are done, and check your settings against it.

Both are in **File** ▸ **Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**You ran the same model twice in earlier chapters and got exactly the same cards. Why?**

A. Because the model had no randomness in it

B. Because with Repeatable runs empty, every run starts from the same default seed, so it draws the same random numbers

C. Because ProcessModel saves the first result and shows it again

The answer is B. Same seed, same random numbers, same results. It means a difference between two runs comes from your change, not from luck.

QUESTION 2

**Why does this shop's Warmup length stay at 0?**

A Because warm-ups are only for models with shifts

**B** Because the shop opens empty every morning, so the whole day, empty start included, is real and should count

C Because a warm-up would make the run longer than 24 hours

The answer is B. A warm-up skips an unrealistic empty start in a process that never empties. A shop that really opens empty counts from the start.

QUESTION 3

**One run said Process Efficiency 23.7%; ten replications average 22.1%. Which do you report?**

A 23.7%, because it came first

**B** 22.1%, the average of ten Mondays, which is far steadier than any single day

C Neither, because they disagree

The answer is B. One run is one possible Monday. The average of many replications is the number to plan with, and the report checks that it can be trusted.

## Try it yourself

See the seed at work.

1. Click **Run Simulation** again, with nothing changed, and open the report.

2 Then open **Simulation Options**, click **Generate** beside **Repeatable runs** to set a new seed, click **Apply**, and run again.

**What to expect.** The first rerun gives exactly the same report: the same seed draws the same ten Mondays. With a new seed you get ten different Mondays, so expect the figures to move a little, and only a little: an average of ten days is steady. If a figure moved a lot, ten replications would not be enough for it. Clear the seed box again and save before Chapter 11.

CHAPTER 11 OF 15

# Chapter 11. Find Where Your Customers Really Wait

By the end of this chapter you can

**Find where customers really wait, and why, in the Output Report.**

Watch the video (4 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 11. Find Where Your Customers Really Wait**
www.youtube.com/watch?v=k9RIKkI2Upg

## Before you start

In Chapter 10 the shop ran ten Mondays and the report said the run could be trusted. Now read what those ten Mondays say. This chapter changes nothing in the shop. It is about **reading**: the skill that turns a model into an answer.

A good report reading follows a fixed order, and the Output Report is built around it. Across its top are five questions, the five any manager asks about a process:

1. **Can I trust it?** Before you believe a number, check the run behind it.
2. **How is it doing?** The headline figures.
3. **Where's the constraint?** The one place that holds everything else back.
4. **Why?** The cause, not just the symptom.
5. **What if I change it?** The fix, tried before you make it.

This chapter answers the first four. Chapter 12 answers the fifth. Keep one question in mind throughout: **where do customers really wait, and why?** In Chapter 9 you might have guessed the answer is Repair, or the parts. The report will tell you otherwise.

**You need** your Phone Repair Shop from Chapter 10, with ten replications set (or the practice model below), and about twenty minutes. Every tab is described in full in the Output Report help.

Step 1.

## Run ten Mondays and open the report

1. On the Welcome screen, under **Recent Models**, click **Phone Repair Shop**.
2. Click **Run Simulation**. The ten replications run in a few seconds.
3. When Simulation Complete asks, click **Yes**. The Output Report opens on its **Overview**.

Step 2.

## Can I trust it, and how is it doing?

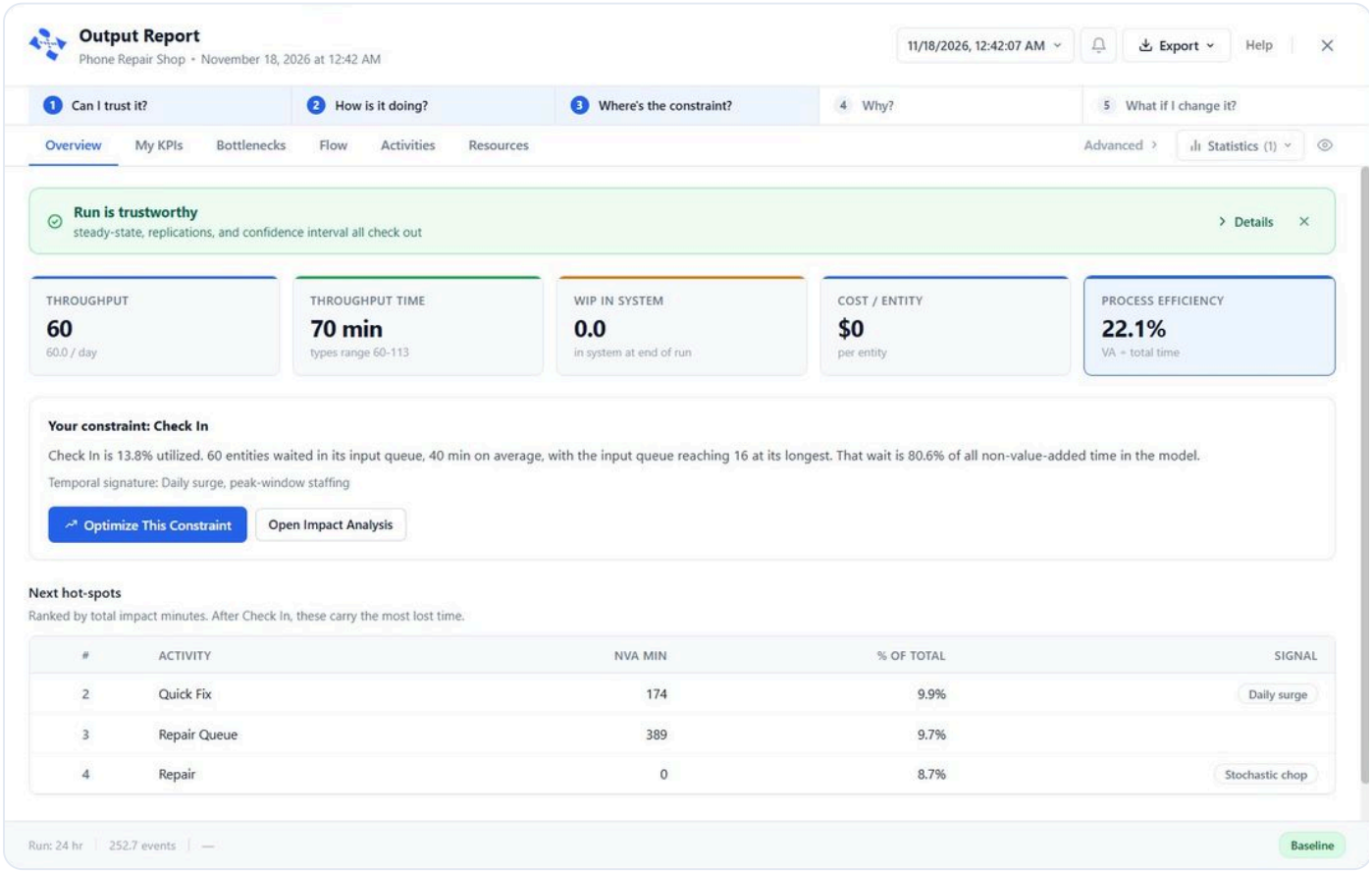



Figure 1. The Overview answers the first three questions on one page.

**Can I trust it?** The green banner says yes: there are enough replications, and they agree. Always look here first. A red or amber banner means the figures below it are not yet safe to act on.

**How is it doing?** Each figure is the average of ten Mondays. **Throughput 60**: every customer is served. **Throughput Time 70 minutes**. And **Process Efficiency 22.1%**: only about a fifth of a customer's time in the shop is spent being worked on. The other four fifths is waiting. That is the size of the prize.

---

Step 3.

## Where's the constraint?

Read the box under the figures: **Your constraint: Check In**. Customers wait there about **40 minutes** on average, and its line reaches **16** at its longest.

Stop and notice how surprising that is. Check In is the quickest step in the shop, two to five minutes, with no staff. Nobody would guess it is the problem by looking at its settings. The model found it by watching where the time goes.

Under it, **Next hot-spots** lists where the most time is lost after Check In: **Quick Fix**, with 174 minutes lost, then the **Repair Queue**, then **Repair**, with none lost: Repair is in the list for its work, not for waiting.

1. Click **Open Impact Analysis**.

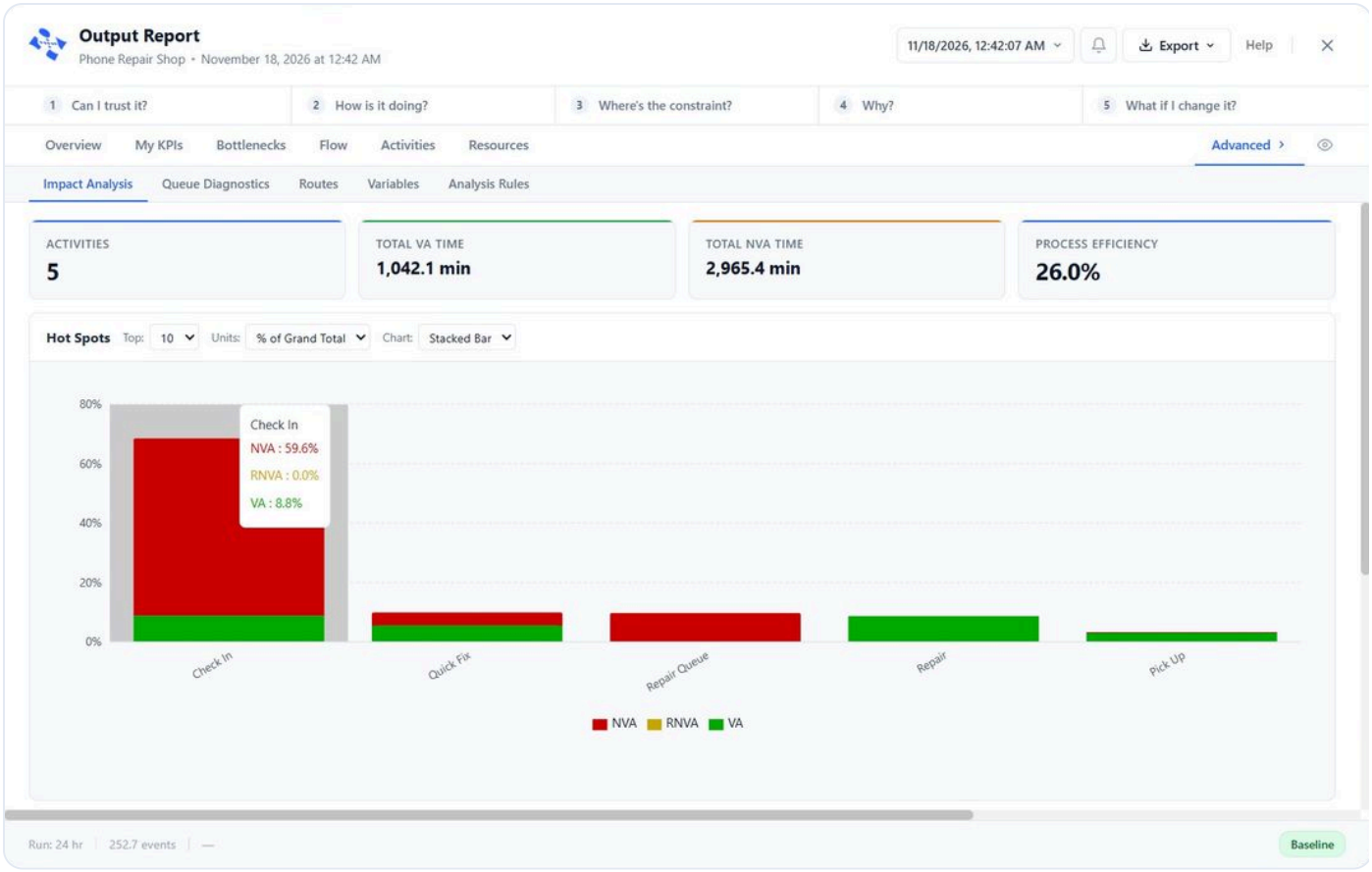



Figure 2. Impact Analysis: each bar splits a step's time into work (green) and waiting (red). Check In's is nearly all waiting.

Each bar splits a step's time into **value-added** work (green) and **non-value-added** waiting (red). Check In's bar is nearly all red. This is the picture to show a manager: it makes the waste visible at a glance.

---

Step 4.

## Make the report your own: My KPIs

1. Click **My KPIs**.

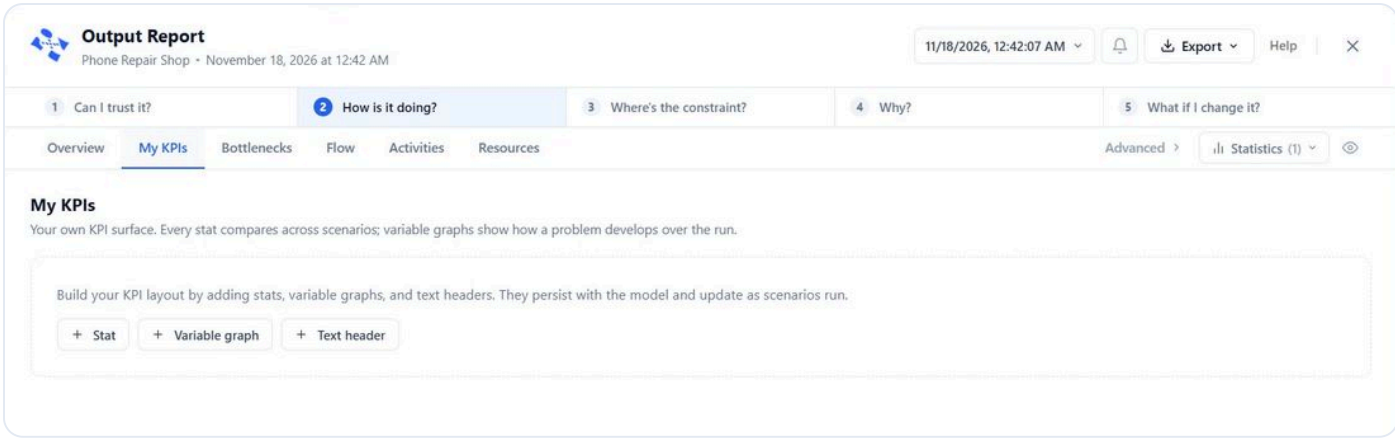



Figure 3. My KPIs: your own page of stats, graphs and headers, saved with the model.

This page starts empty. It is yours: add stats, graphs of variables and headers, and they are saved with the model, so the same page is waiting every time you run it. In a real project, this is where you put the three or four numbers your manager cares about.

---

Step 5.

## When does the line build? Bottlenecks

1. Click **Bottlenecks**.

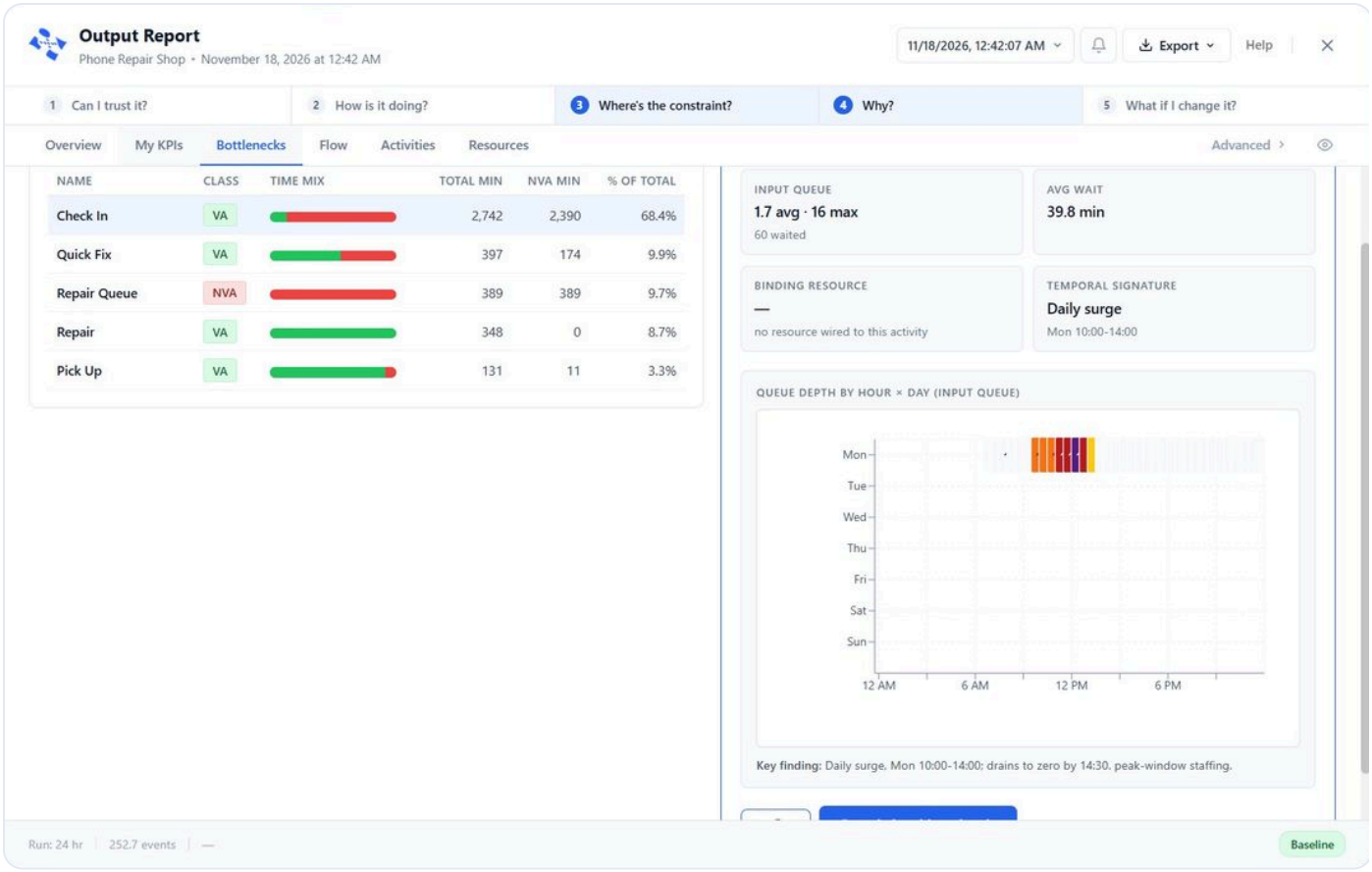



Figure 4. Bottlenecks: Check In first, and a chart of when its line builds.

Check In comes first. Its chart shows the line by hour: empty all morning, building from **10**, when the office group of twelve arrives, and gone by about half past 2.

Now the cause begins to show. The line is not there all day. It is a **surge**, and it starts at the moment twelve customers arrive at once.

---

Step 6.

## Who waits longest? Flow

1. Click **Flow**.

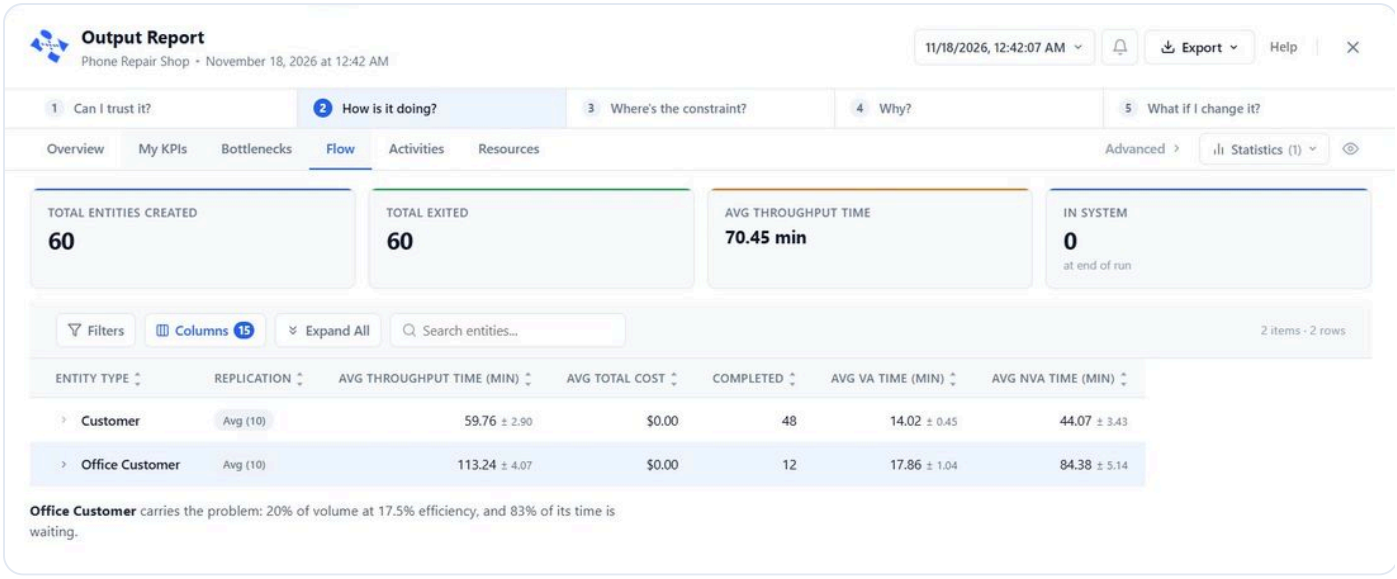



Figure 5. Flow: one row for each kind of customer. The office customers wait far longest.

One row for each kind of customer. A walk-in spends about **an hour** in the shop. An office customer spends nearly **two**, and most of it waiting. The office group is a fifth of the customers but carries most of the problem.

---

Step 7.

## Why? Activities

1. Click **Activities**.

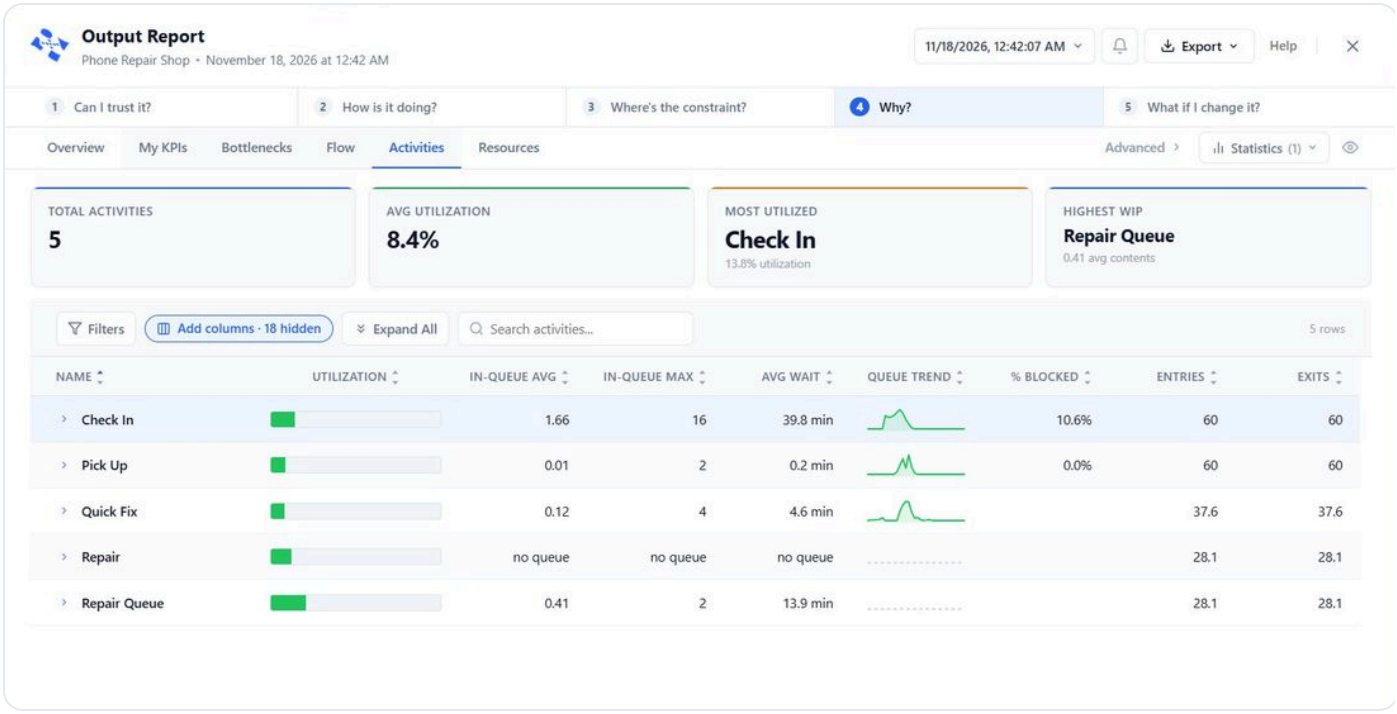



Figure 6. Activities: Check In is busy only 13.8% of the time, but blocked 10.6% of it.

Here is the **why**. Check In is busy only **13.8%** of the time: it is not overworked. But it is **blocked 10.6%** of the time: it holds a customer it cannot pass on. You met blocking in Chapter 6: when the Repair Queue is full, the customer at Check In has nowhere to go, so Check In stops. And in Chapter 9 you saw what fills the Repair Queue: customers waiting for **parts**.

Put the clues together, the way the report laid them out:

- The line is at Check In, but Check In is not slow.
- It builds when the office group arrives, and office customers go straight to the Repair Queue.
- Check In is blocked: the Repair Queue behind it is full.
- The Repair Queue fills when the parts run out.

The wait shows up at the front door, but its cause is at the back: the parts supply. That is why reading the report in order matters. Each tab moves you one step from the symptom to the cause.

---

Step 8.

## The Technicians: Resources

1 Click **Resources**.

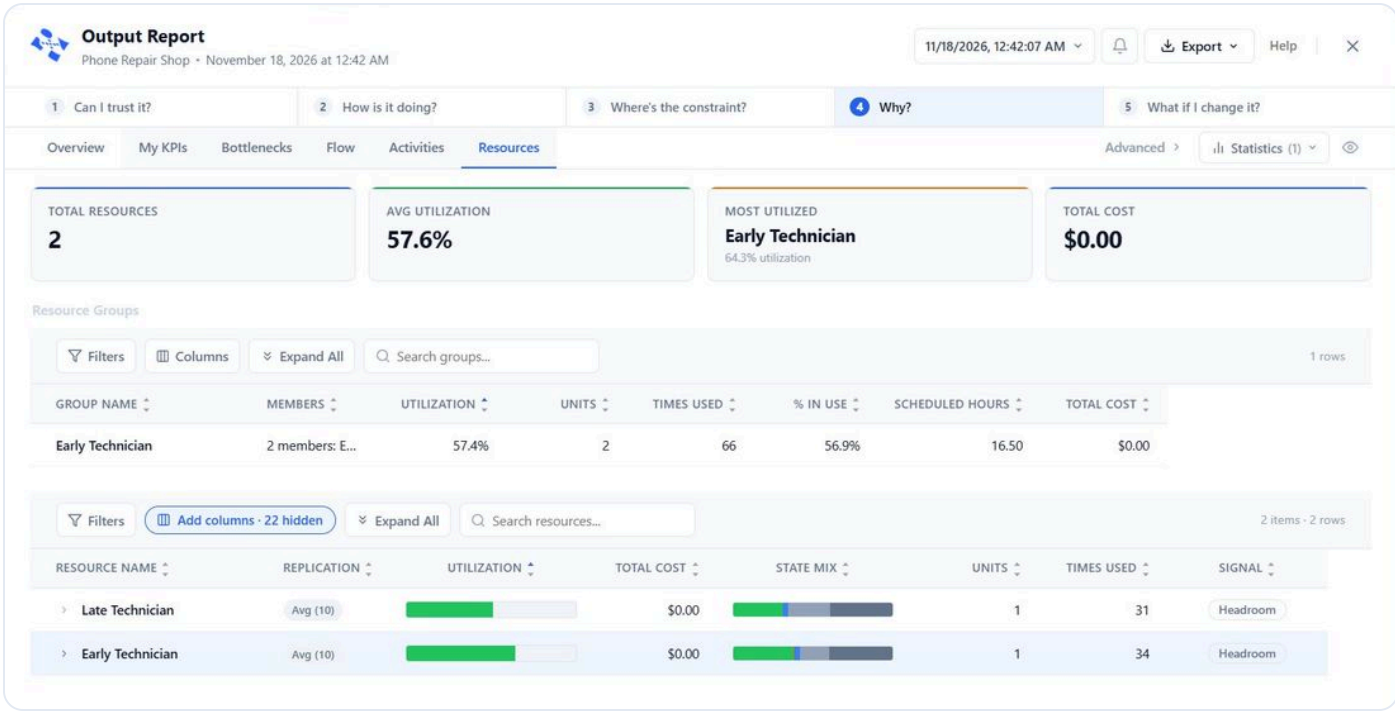



Figure 7. Resources: both Technicians show Headroom. They have time to spare.

The two Technicians work as one group, and both show **Headroom**: they have time to spare. So hiring a third Technician is not the obvious fix it might seem. Keep that in mind for Chapter 12.

2 Close the report. The last question, **What if I change it?**, is Chapter 12.

Nothing in the model changed, so there is nothing to save.

---

## Your practice model

START FROM

**Chapter 10. Run the Day Ten Times, Not Once**

The shop as Chapter 10 left it. Use your own saved Phone Repair Shop instead if you built it.

COMPARE WITH

**Chapter 11. Find Where Your Customers Really Wait**

The shop as this chapter leaves it. Open it when you are done, and check your settings against it.

Both are in **File** ▸ **Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**Check In is busy only 13.8% of the time, yet it is the constraint. How can that be?**

A The report is wrong: a step that is rarely busy cannot be a constraint

B It is blocked: the full Repair Queue behind it leaves the customer at Check In nowhere to go

C Its processing time is too long

The answer is B. A step can hold up the line without working hard. Check In is blocked 10.6% of the time, waiting for room in the Repair Queue.

QUESTION 2

**Why read the banner on the Overview before any figure?**

A It is at the top of the page

B It says whether the run behind the figures can be trusted; if it cannot, the figures are not yet safe to act on

C It holds the most important number

The answer is B. Can I trust it? comes first. The banner checks the replications and how well they agree.

QUESTION 3

**Both Technicians show Headroom. What does that suggest about adding a third?**

A It is the best fix, because more staff always means less waiting

B It may help little: the Technicians already have time to spare, so the wait has another cause

C It is impossible to tell

The answer is B. Headroom means spare time. Chapter 12 tries one more Technician and a bigger parts box side by side to see which helps more.

## Try it yourself

Find the cause from the other end: start at the parts.

1. Run the ten Mondays again and open the report.
2. Click **Activities** and find the **Repair Queue** row. Read its average wait and its maximum line.
3. Compare them with what this chapter found at **Check In**.

**What to expect.** A short line, but a full one. In one run of the practice model, the Repair Queue held 0.41 customers on average and 2 at most, with an average wait of 13.9 minutes. Two is its whole capacity: at its busiest it is full, and a full Repair Queue is what blocks Check In, where customers wait about 40 minutes. The long wait at the door and the full queue behind it are one jam seen from both ends, and the trail leads back, past the queue, to the parts.

CHAPTER 12 OF 15

# Chapter 12. Try Two Fixes Side by Side

By the end of this chapter you can

**Try two fixes side by side before you spend on either.**

Watch the video (4 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 12. Try Two Fixes Side by Side**
www.youtube.com/watch?v=7KPGRrMyCBQ

## Before you start

Chapter 11 found where customers wait, at Check In, and traced it back to the parts. Now the fifth question: **what if I change it?**

There are two fixes a manager might suggest. Hire one more Technician for the morning; or order parts in bigger boxes, 20 instead of 10. Both cost money. In a real shop you would have to pick one and hope. In a model you can **try both first**, side by side, and spend only on the one that works.

The tool for this is the **scenario**: a what-if copy of the shop that changes a few settings and leaves the shop itself alone. You build each fix as a scenario, run today's shop and every scenario together, and read them in one report.

| Part | Chapter 11 | Chapter 12 |
|---|---|---|
| Scenario: One more Technician | None | Early Technician's Quantity 1 becomes 2 |
| Scenario: Box of 20 | None | The Parts Shelf's Reorder Qty 10 becomes 20 |
| The shop itself | As Chapter 10 left it | Unchanged |

**You need** your Phone Repair Shop (or the practice model below) and about twenty minutes. For more on scenarios, see Scenarios. Make a guess before you start: which fix will help more?

Step 1.

## Open your shop

1. On the Welcome screen, under **Recent Models**, click **Phone Repair Shop**.

Step 2.

## Build the first scenario: one more Technician

1. On the toolbar, click **Logic Builder**, then the **Scenarios** tab. It reads “No scenarios defined” for now.
2. Click **Add new** and name it **One more Technician**.
3. Click **Add Object**. The menu lists every object in the shop. Scroll to **Early Technician** and pick it.
4. Click **Quantity** and type **2**.

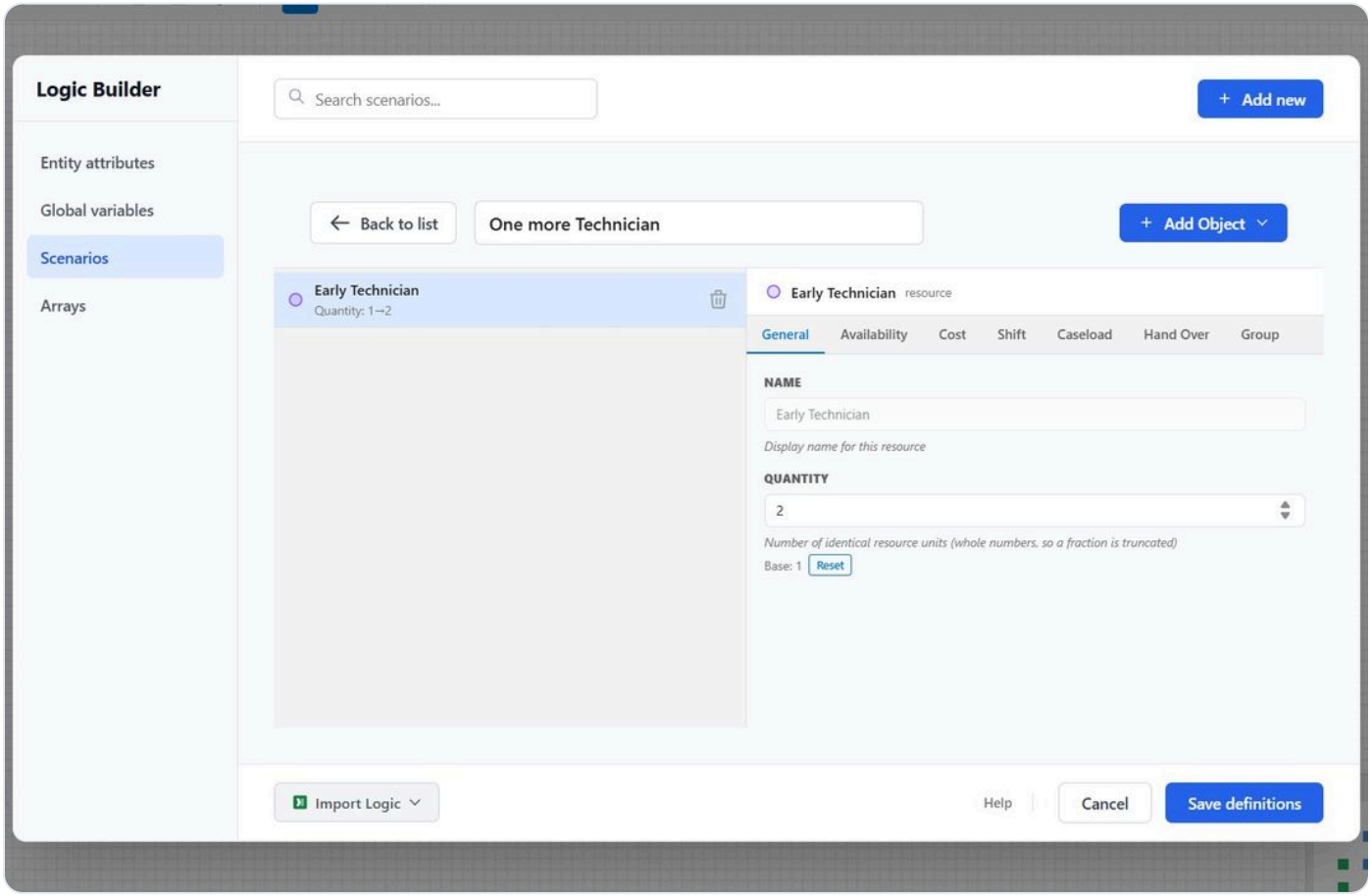



Figure 1. The scenario One more Technician: Early Technician's Quantity, 1 in the shop, 2 in the scenario.

> **What you should see:** the object's line reads **Quantity: 1 to 2**, and under the box, **Base: 1**, the shop as it is. A scenario always shows you both, so you never lose sight of what is real.

5. Click **Back to list**.

---

Step 3.

## Build the second scenario: a box of 20

1. Click **Add new** again and name it **Box of 20**.
2. Click **Add Object**, scroll to **Parts Shelf**, and pick it.
3. Scroll down to **Reorder Qty**, click it, and type **20**. Under it, **Base** reads 10.

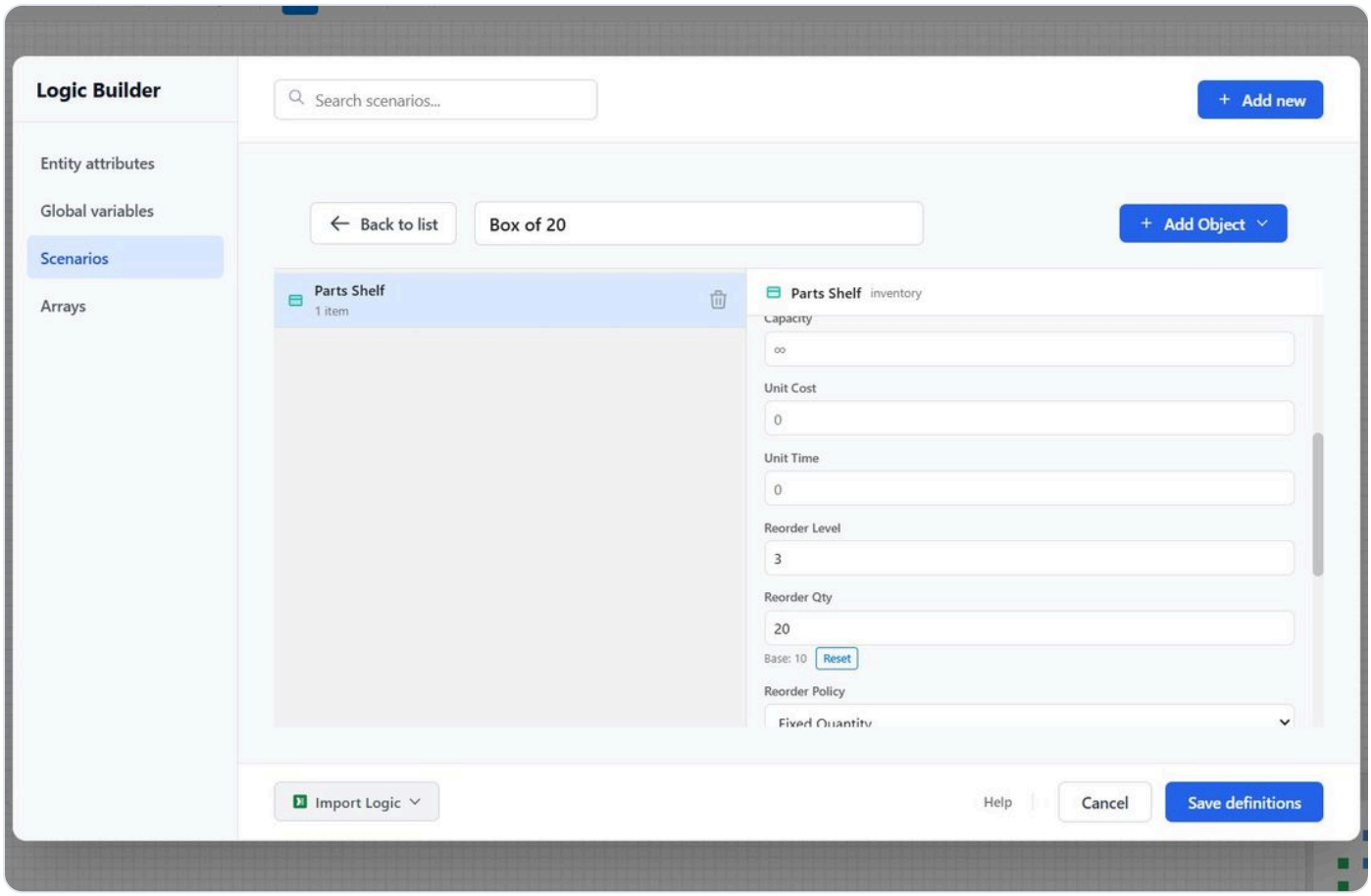



Figure 2. The scenario Box of 20: the Parts Shelf orders 20 at a time instead of 10.

4. Click **Back to list**.

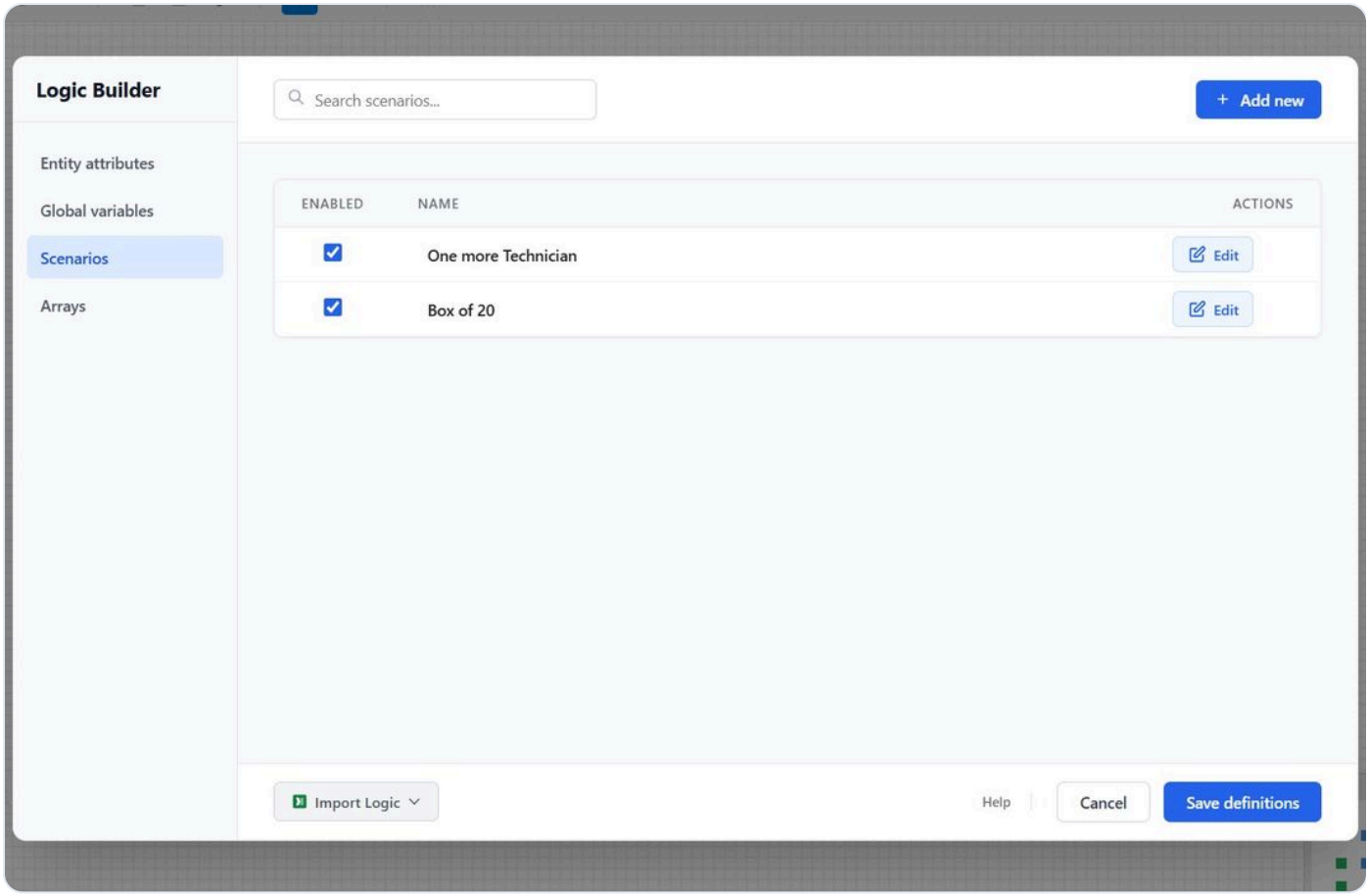



Figure 3. Both scenarios on the list, each switched on.

> **What you should see:** both scenarios on the list, each with **Enabled** ticked. Only enabled scenarios run.

5 Click **Save definitions**.

Notice what you have not done: you have not changed the shop. Early Technician is still 1 and the box is still 10. The scenarios sit beside the shop, ready to be tried.

---

Step 4.

## Run all three

1 Click **Run Simulation**.

The shop as it is runs ten Mondays, and then each scenario runs ten of its own: thirty Mondays in all, in a few seconds.

2 Click **Yes** when Simulation Complete asks.

Step 5.

## Compare each fix with today’s shop

1 Click the **Experiments** tab. It compares a scenario with the shop as it is (the **Baseline**), and opens on **Box of 20**.

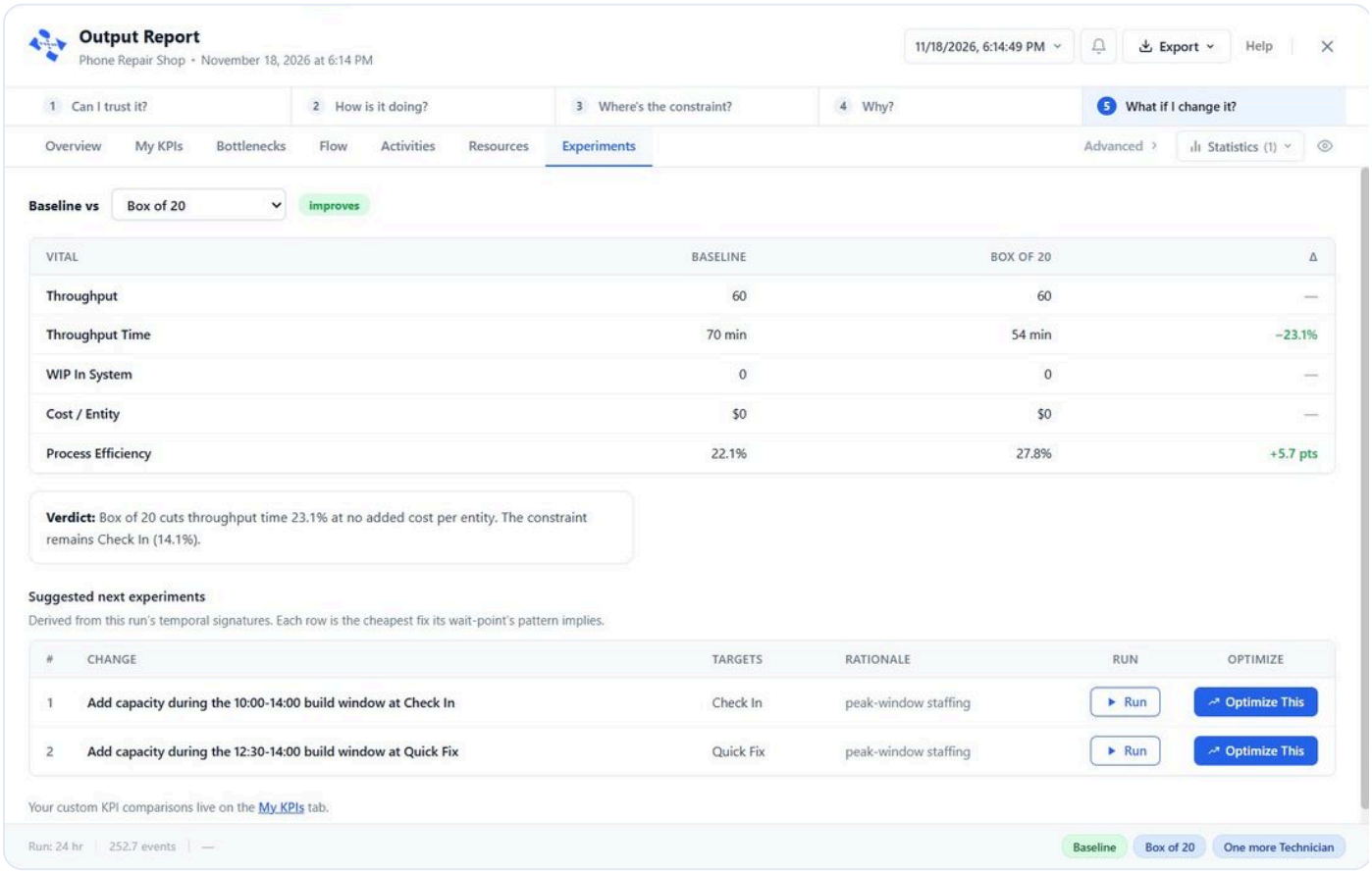



Figure 4. Box of 20 against the Baseline: Throughput Time from 70 minutes to 54.

**Box of 20** improves the shop: Throughput Time goes from **70 minutes to 54**, a cut of 23.1%, and the verdict under the table says so in words.

2 Pick **One more Technician** from the list beside “Baseline vs”.

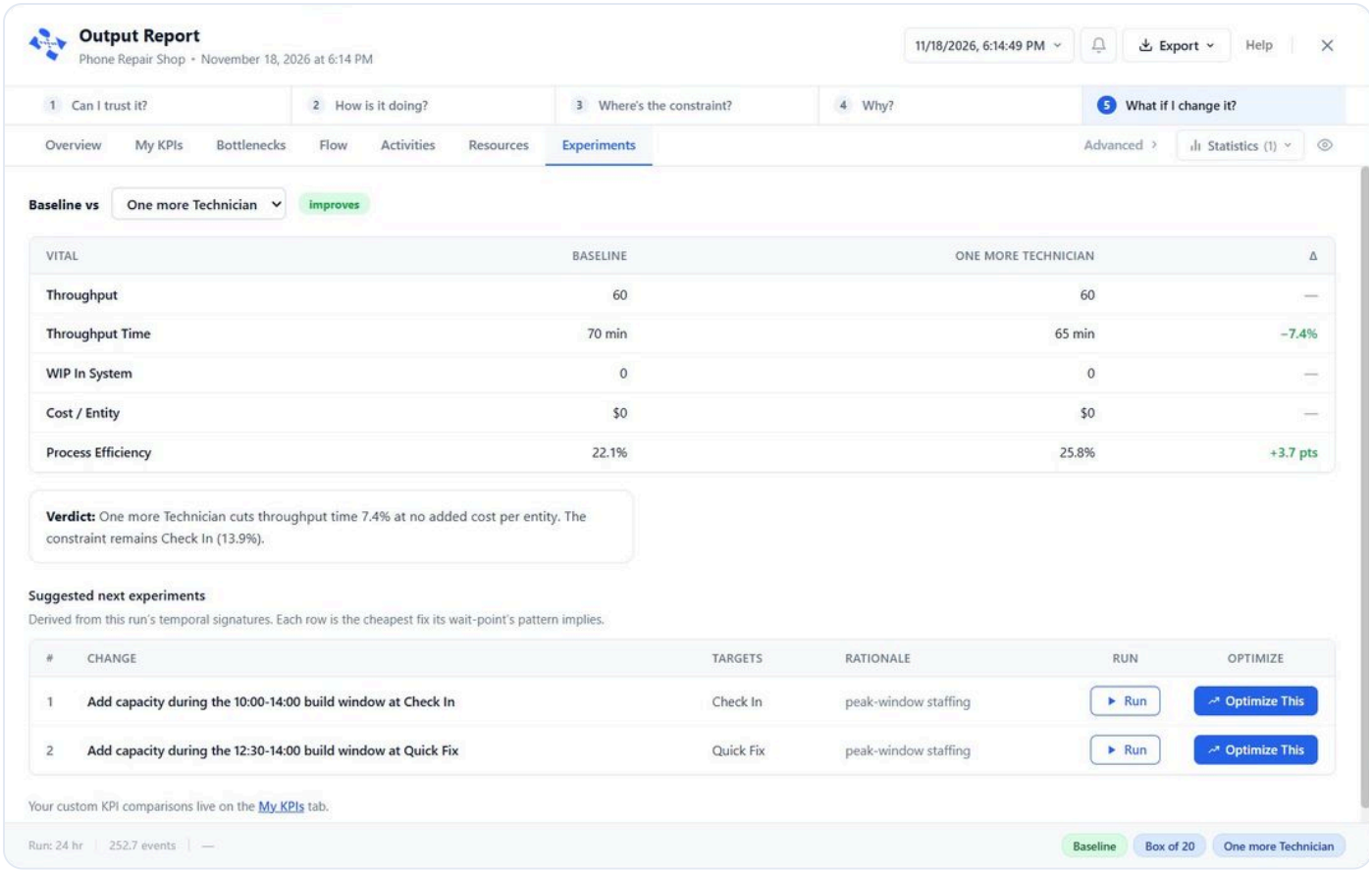



Figure 5. One more Technician against the Baseline: Throughput Time from 70 minutes to 65.

It improves too, but less: from **70 minutes to 65**, a cut of 7.4%.

---

Step 6.

## Read the answer at Check In

The constraint was Check In, so look there.

1. Click the **Activities** tab. Each step now has a row for each run: the Baseline and each scenario.

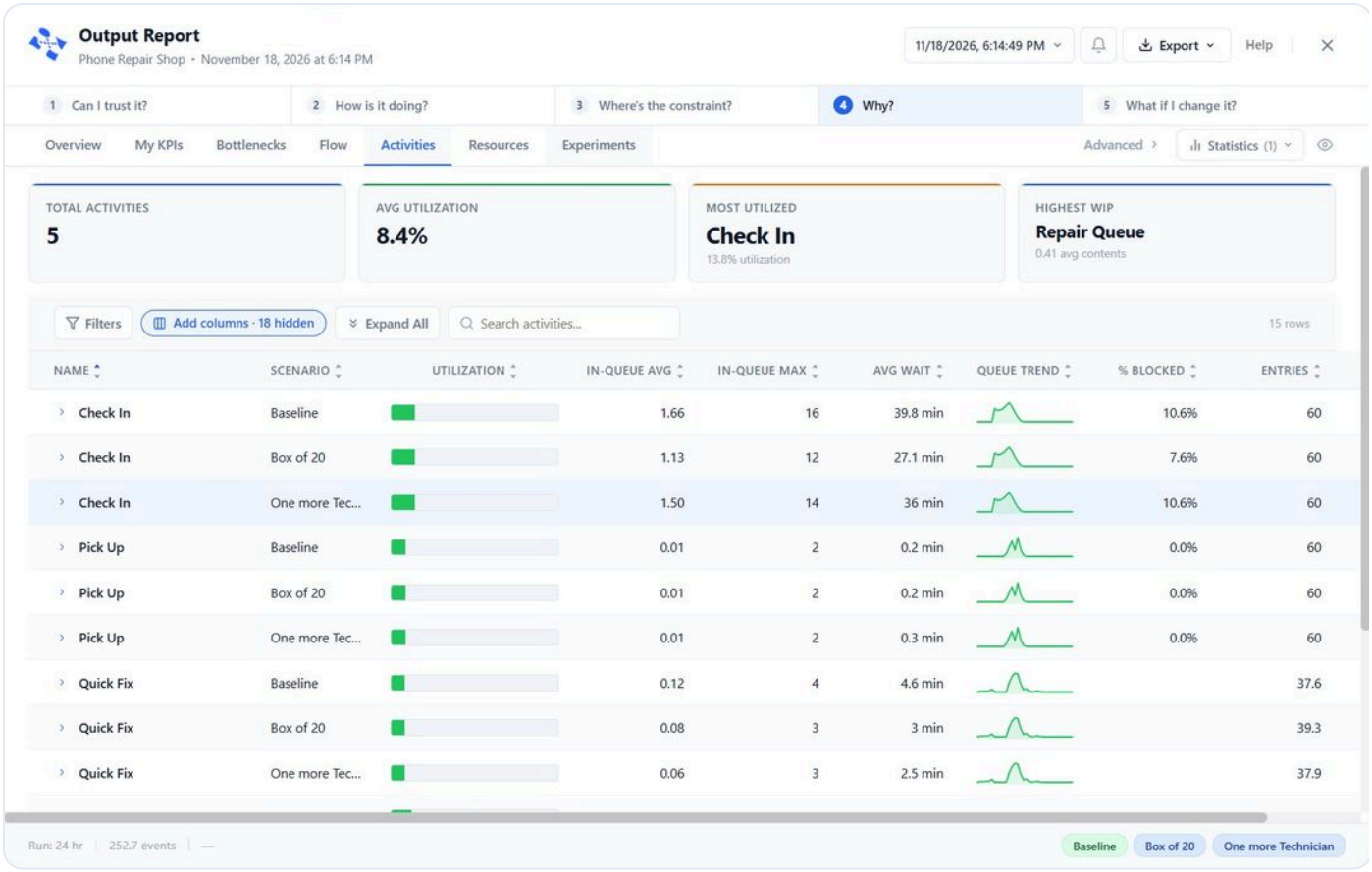



Figure 6. Check In's wait: 39.8 minutes today, 27.1 with a box of 20, 36 with one more Technician.

Check In's average wait: **39.8 minutes** in the shop as it is, **27.1** with a box of 20, **36** with one more Technician. The bigger box helps about three times as much.

Here is why, and it is the whole of Chapters 9 and 11 in one sentence: repairs were waiting for **parts**, not for a Technician. A second Technician cannot repair a phone without a part. A bigger box keeps the shelf from running dry, so the Repair Queue moves, and Check In is no longer blocked.

If you guessed the Technician, you are in good company: it is the fix most people reach for first. The model just saved the cost of a hire that would have helped little.

3 Close the report.

---

Step 7.

## Save it

1 Click **File**, then **Save**. The scenarios are saved with the model. Chapter 13 starts from this shop.

## Your practice model

START FROM

**Chapter 11. Find Where Your Customers Really Wait**

The shop as Chapter 11 left it. Use your own saved Phone Repair Shop instead if you built it.

COMPARE WITH

**Chapter 12. Try Two Fixes Side by Side**

The shop as this chapter leaves it. Open it when you are done, and check your settings against it.

Both are in **File** ▸ **Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**You build the scenario One more Technician. What happens to the shop itself?**

A Early Technician becomes 2 in the shop

B Nothing: the scenario is a what-if copy, and the shop keeps its own settings

C The shop is replaced by the scenario until you delete it

The answer is B. A scenario changes settings only for its own runs. The Base value shown under each changed setting is the shop as it is.

QUESTION 2

**Why did a bigger parts box help more than one more Technician?**

A Because Technicians are slower than parts

B Because repairs were waiting for parts, not for a Technician; a second Technician cannot repair without a part

C Because the box of 20 also added a Technician

The answer is B. Fix the cause, not the symptom. Chapter 11 traced the wait back to the parts, and the scenarios confirm it.

QUESTION 3

**Where do you compare a scenario with today's shop?**

A On the canvas, by switching between the scenarios

B In the Output Report's Experiments tab, scenario against Baseline, and row by row in tabs such as Activities

C Only in the Logic Builder

The answer is B. Experiments puts each scenario beside the Baseline with a verdict; the other tabs show a row per scenario.

## Try it yourself

Try both fixes at once.

1. Open the **Logic Builder**, the **Scenarios** tab, and add a third scenario, **Both**, that sets Early Technician's Quantity to **2** and the Parts Shelf's Reorder Qty to **20**. Click **Save definitions**.

2 Run and open the report. In **Experiments**, pick **Both**.

**What to expect.** Expect Both to cut Throughput Time at least about as much as Box of 20 alone, since it removes the parts shortage and adds hands. In one run of the practice model, Both cut it from 70 minutes to 28, against 54 for Box of 20 alone and 65 for One more Technician. But compare it with Box of 20 alone: is the extra cut, 26 minutes in that run, worth a second hire? That trade-off, how much waiting a resource buys, is exactly what Chapter 13 hands to the software. Delete the Both scenario, or close without saving, before Chapter 13.

CHAPTER 13 OF 15

# Chapter 13. Let the Software Find the Best Setup

By the end of this chapter you can

**Let PM Optimizer find the best setup within your limits.**

Watch the video (5 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 13. Let the Software Find the Best Setup**
www.youtube.com/watch?v=Wsy1XNDqGkw

## Before you start

In Chapter 12 you tried two fixes by hand, and the bigger parts box helped most. But the real question is wider: **how many Technicians, and how big a box, keep the wait at Check In under 30 minutes, with as few Technicians as possible?** There are many combinations, and trying each one by hand is slow. **PM Optimizer** tries them for you.

Four words carry this chapter. Learn them once and every optimization makes sense:

- An **experiment** is one search: what to change, what to measure, and how.
- A **factor** is a setting the optimizer may change, with the values to try. Here: the number of early Technicians (1, 2 or 3) and the size of the parts box (10 or 20).
- A **response** is what you measure, with its goal: lower is better or higher is better. Here: the wait at Check In, and the Technicians used. A response can also carry a **limit**: any combination over it cannot win.
- The **design** is how the combinations are explored. **Full Factorial** tries every one.

Three Technician counts times two box sizes is **6 combinations**. With ten Mondays each, as Chapter 10 taught, that is 60 runs.

**You need** your Phone Repair Shop from Chapter 12 (or the practice model below) and about twenty minutes. The PM Optimizer help covers every setting.

Step 1.

## Open your shop and start an experiment

1. On the Welcome screen, under **Recent Models**, click **Phone Repair Shop**.
2. On the toolbar, click **PM Optimizer**. It works in three steps, shown across its top: **Define**, **Run** and **Results**.
3. Click **New Experiment**, click the pencil beside it, and name it **Fewest Technicians**.

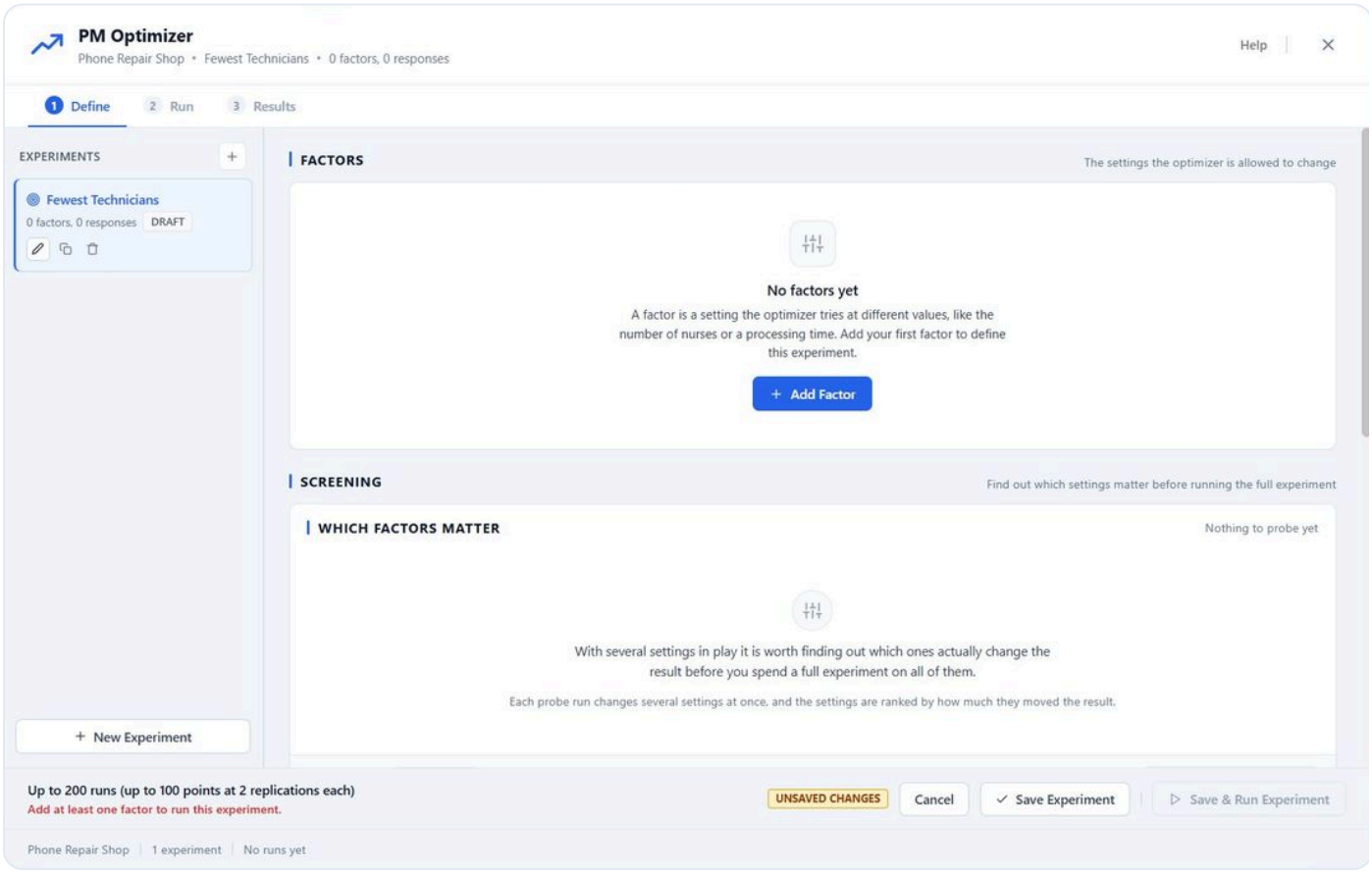



Figure 1. A new experiment, Fewest Technicians, with no factors yet.

Step 2.

## Add the first factor: the early Technicians

1. Click **Add Factor**. Under **Resources**, pick **Early Technician**, and click **Quantity**. It shows **Today: 1**.
2. Choose **Use a range**: **Minimum 1**, **Maximum 3**, **Step 1**.

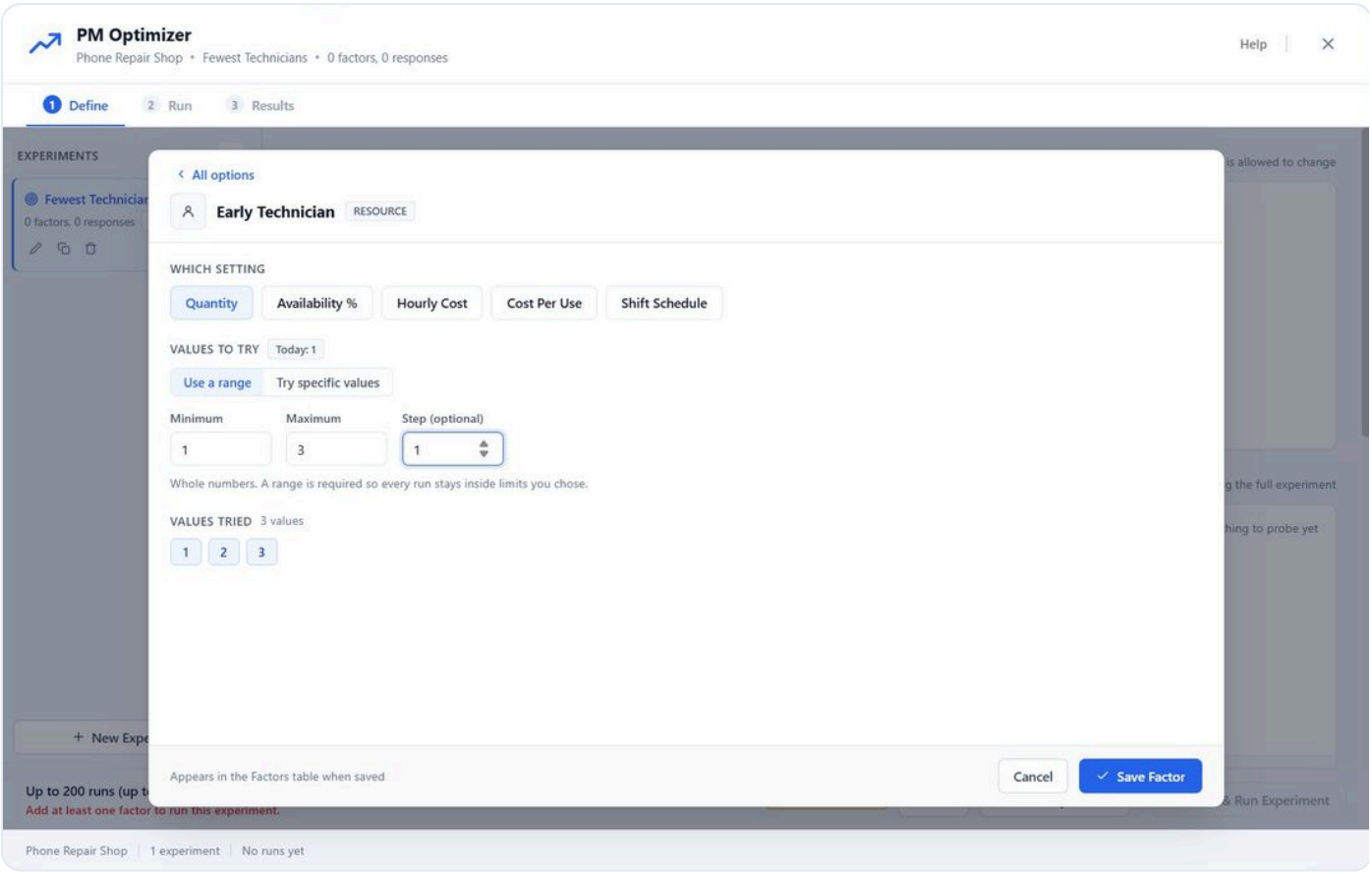



Figure 2. The first factor: Early Technician, 1 to 3.

> **What you should see:** under **Values tried**, the three values the optimizer will use: 1, 2 and 3.

3. Click **Save Factor**.

---

Step 3.

## Add the second factor: the box size

1. Click **Add Factor** again. Scroll down to **Inventory** and pick **Parts Shelf: Part**.
2. Click **Reorder Qty**, the size of the parts box. Today it is 10.
3. Try a **Minimum** of **10**, a **Maximum** of **20** and a **Step** of **10**: the box will be 10 or 20.
4. Click **Save Factor**.

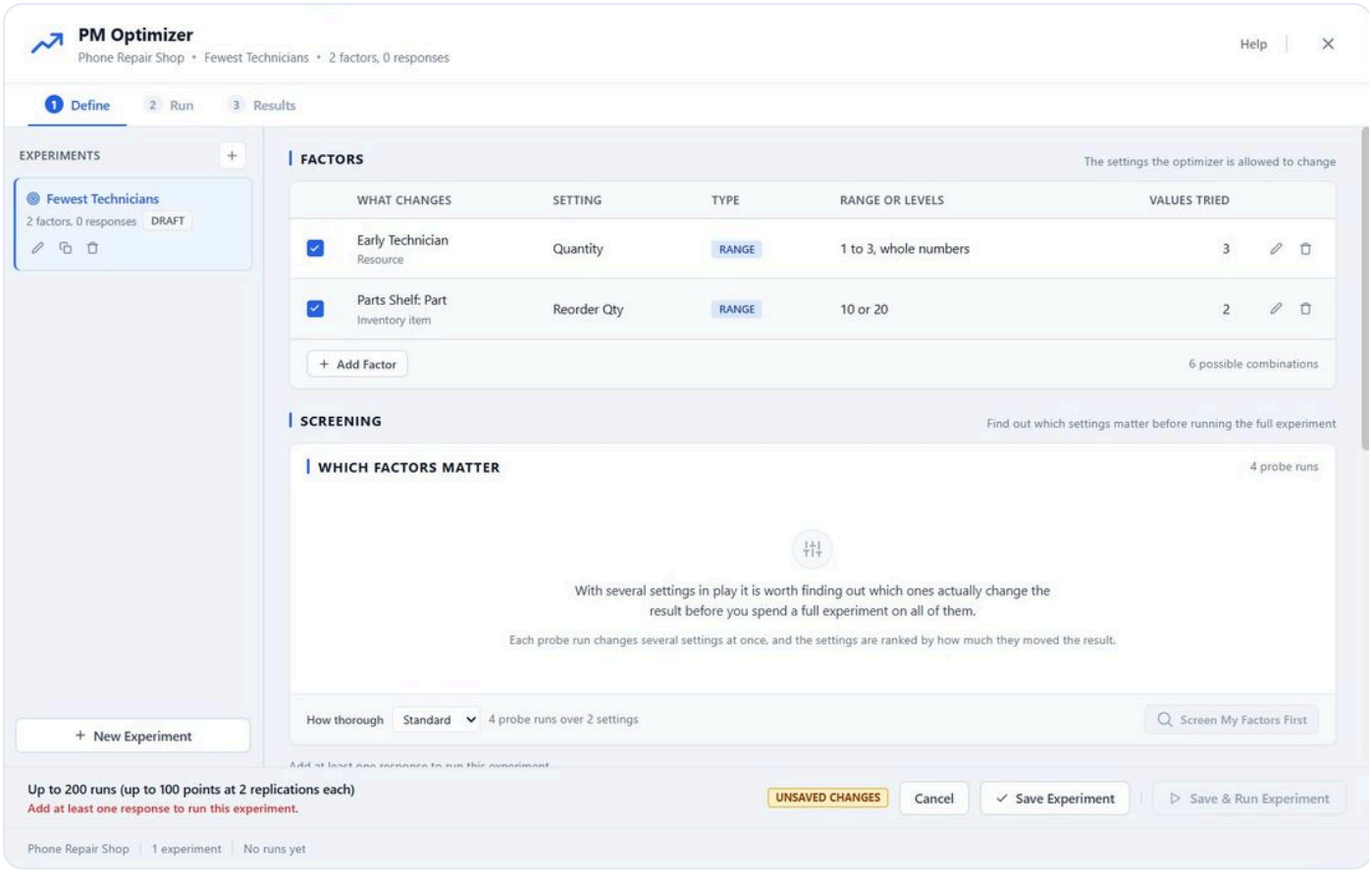



Figure 3. Two factors: 3 times 2, six possible combinations.

> **What you should see:** both factors in the table, and under them, **6 possible combinations**.

Step 4.

## Say what to measure, and the limit

1. Scroll down and click **Add Response**. Pick **Waiting time**, and the step, **Check In**. Three more responses come with it: Total run cost, Average time in system and Resources used.
2. Untick the two you do not need, **Total run cost** and **Average time in system**.
3. Keep **Resources used**: it counts the Technicians, and **Lower is better**. That is your second goal: fewer Technicians.
4. For **Check In: Waiting time**, lower is better too. Tick **Limit** and type **30**: any combination where customers wait more than 30 minutes cannot win.

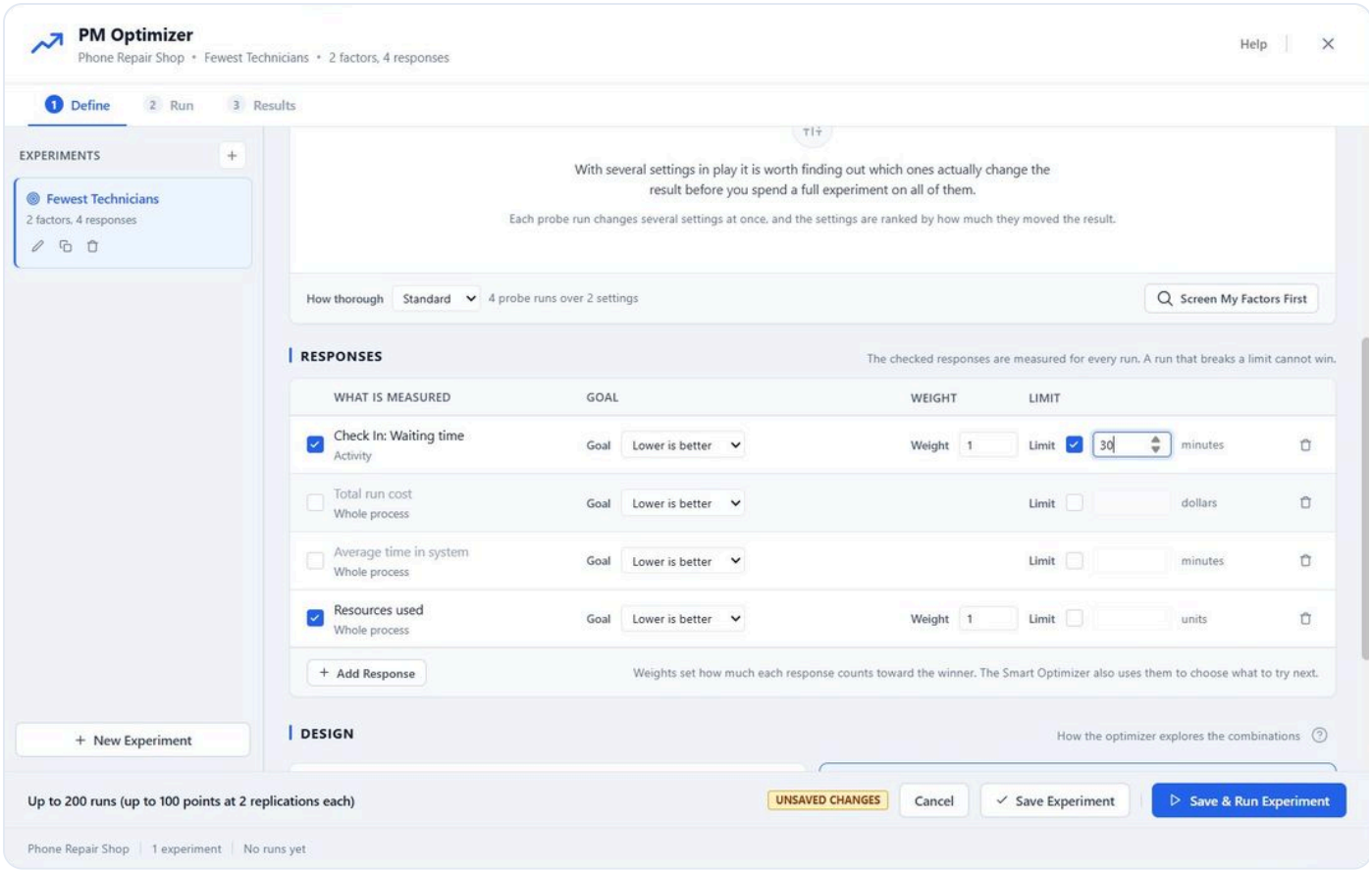



Figure 4. Two goals, a short wait and few Technicians, and a limit of 30 minutes on the wait.

Think about what you just set up, because it is the heart of optimization: **two goals that pull against each other**. More Technicians shorten the wait; fewer Technicians cost less. The optimizer weighs them, with the **Weight** each response carries (1 each here), and the limit rules out any setup the shop could not accept.

---

Step 5.

## Choose the design and run it

1. Scroll down to **Design** and click **Full Factorial**: it tries every combination.
2. Set **Replications** to **10**: ten Mondays for each combination, as in Chapter 10.

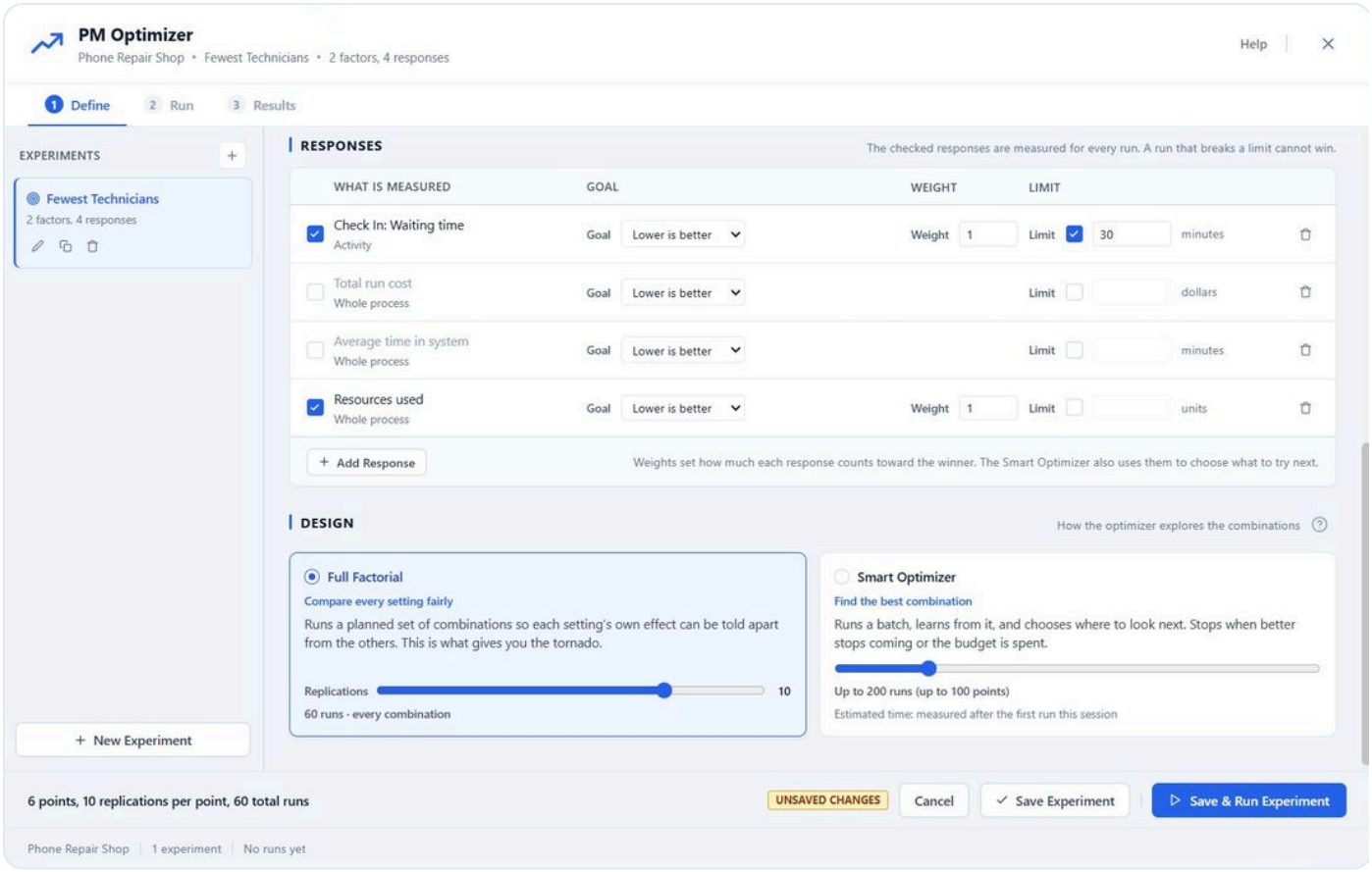



Figure 5. Full Factorial, ten replications each: 6 combinations, 60 runs.

> **What you should see:** the foot reads **6 points, 10 replications per point, 60 total runs**.

3 Click **Save & Run Experiment**. It runs all 60, and the results open by themselves on the **Results** step.

Step 6.

## Read the answer, and why it won

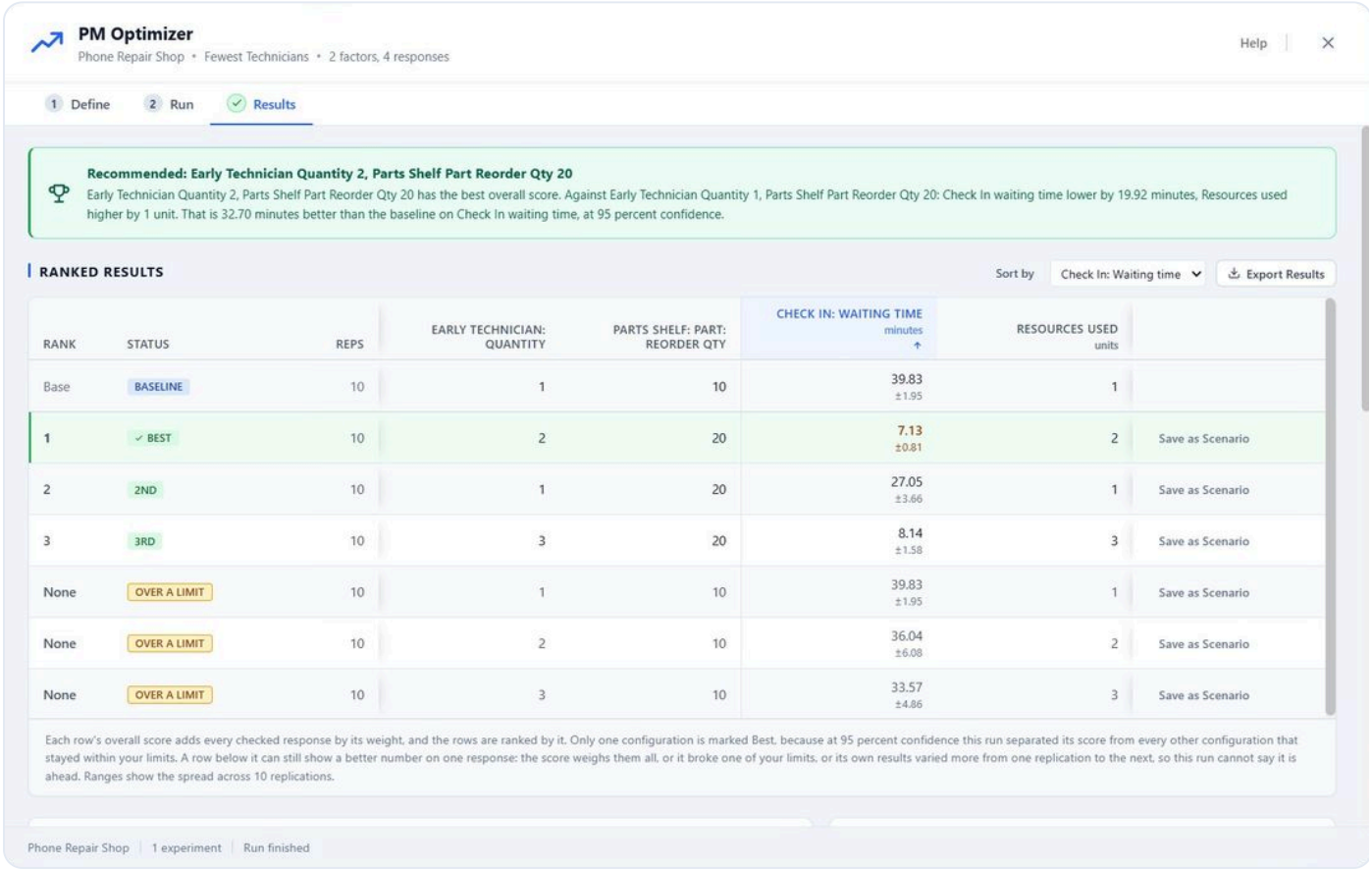



Figure 6. The answer: two early Technicians and a box of 20. The table shows why.

The answer is at the top. **Recommended: Early Technician Quantity 2, and a Reorder Qty of 20.** Under it, the table ranks every combination. Read it as the optimizer did:

- **Baseline**, today's shop (1 Technician, box of 10): customers wait about **40 minutes**.
- **With a box of 10, nothing gets under 30**, however many Technicians: the three rows at the bottom are **Over a limit**. The parts are the problem, as Chapter 11 found, so no number of people fixes it.
- With a box of 20: **one** Technician waits about **27 minutes**, under the limit; **two** wait about **7**; **three** wait about **8**, no better than two.

So why two, and not one? One Technician with a box of 20 meets the limit with the fewest people. But both goals count, and the recommendation says what one more Technician buys: about **20 minutes less waiting**. With the weights you set, that is worth one more person. Three would add a person and save nothing.

This is the value of letting the software search: it tried every combination, threw out the ones over your limit, and weighed your two goals the way you told it to. If you weighted fewer Technicians more heavily, the answer could change, and that would be a decision about what your shop values, not about the model.

Step 7.

## Keep the answer as a scenario

1. On the best row, click **Save as Scenario**. It joins the shop as a scenario, turned on for your next simulation, just like those you built by hand in Chapter 12.
2. Close PM Optimizer.
3. Click **File**, then **Save**. The experiment and its new scenario are saved with the model.

## Your practice model

START FROM

**Chapter 12. Try Two Fixes Side by Side**

The shop as Chapter 12 left it. Use your own saved Phone Repair Shop instead if you built it.

COMPARE WITH

**Chapter 13. Let the Software Find the Best Setup**

The shop as this chapter leaves it. Open it when you are done, and check your settings against it.

Both are in **File** ▸ **Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**Why are the three combinations with a box of 10 marked Over a limit, however many Technicians they have?**

A Because the optimizer only tries a box of 20

B Because the wait at Check In stays over the 30-minute limit: the parts run out, and more people cannot fix that

C Because they cost too much

The answer is B. A limit rules out any combination that breaks it. With a box of 10 the parts shortage keeps the wait over 30 minutes.

QUESTION 2

**One Technician with a box of 20 is under the limit and uses fewer people. Why is two recommended?**

A Because the optimizer always picks the middle value

B Because both goals count, and one more Technician buys about 20 minutes less waiting, which outweighs one more person with the weights set

C Because one Technician is over the limit

The answer is B. The optimizer weighs every checked response by its weight. Change the weights and the answer can change.

QUESTION 3

**What is a factor in PM Optimizer?**

A. A result the optimizer measures

B. A setting the optimizer may change, with the values to try

C. The number of replications

The answer is B. Factors are what changes; responses are what is measured; the design decides which combinations are run.

## Try it yourself

Change what the shop values, and see whether the answer changes.

1. Open **PM Optimizer**, select **Fewest Technicians**, and on the **Define** step raise the **Weight** of **Resources used** from 1 to **3**, so each Technician counts three times as much.
2. Click **Save & Run Experiment** and read the recommendation.

**What to expect.** With people weighted more heavily, the recommendation may move to **one** Technician with a box of 20: it still meets the 30-minute limit, and each extra person now counts for more against the minutes it saves. In one run of the practice model it did, with a wait at Check In of about 27 minutes. Whichever it recommends, read its sentence: it says what the winner gains against the runner-up. In that run: one Technician fewer, for about 20 minutes more waiting at Check In. The model did not change; your priorities did. Set the weight back to 1, or close without saving, before Chapter 14.

CHAPTER 14 OF 15

# Chapter 14. Let Your Own AI Build the Next Shop

By the end of this chapter you can

**Start faster, from a demo model or with your own AI.**

Watch the video (4 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 14. Let Your Own AI Build the Next Shop**
www.youtube.com/watch?v=Jer_7jXmMAM

## Before you start

For thirteen chapters you built one shop by hand, a shape at a time. That was the point: you now know what every shape, line and setting does, and you can read a report and judge a fix. Building by hand is how you learn. It is not always the fastest way to **start**.

This chapter shows the faster starts, from the one you already know to the newest:

- **A demo model**: a finished model of a real problem. Open the one closest to your process and change it.
- **Your own AI**, through **PM AI Bridge**: describe a process in plain words and let your AI build it, run it and read the results, while you watch and can take over at any time.
- **An industry palette** and **Import**, for two more special starts.

None of these replaces what you learned. They save you the typing, and what you learned is how you check the result.

**You need** about twenty minutes. For the AI part you need PM AI Bridge, which is included with Professional (the 14-day free trial includes the full app, with PM AI Bridge available to turn on), and an AI app installed on your computer, such as Claude, ChatGPT (Codex), Cursor, Google Antigravity or Grok Build, with your own account for it. Your phone repair shop is not needed: this chapter starts something new.

Step 1.

# Start from a demo model

You opened a demo in Chapter 1 to learn the window. Now open one as a **starting point**.

1. Close the Welcome screen, then click **Demo Models** on the toolbar.
2. Click **Services**, and scroll down to **The Pile Behind the Counter**.
3. Click **Read the brief**: the story behind the model, the problem it answers and how it is built.

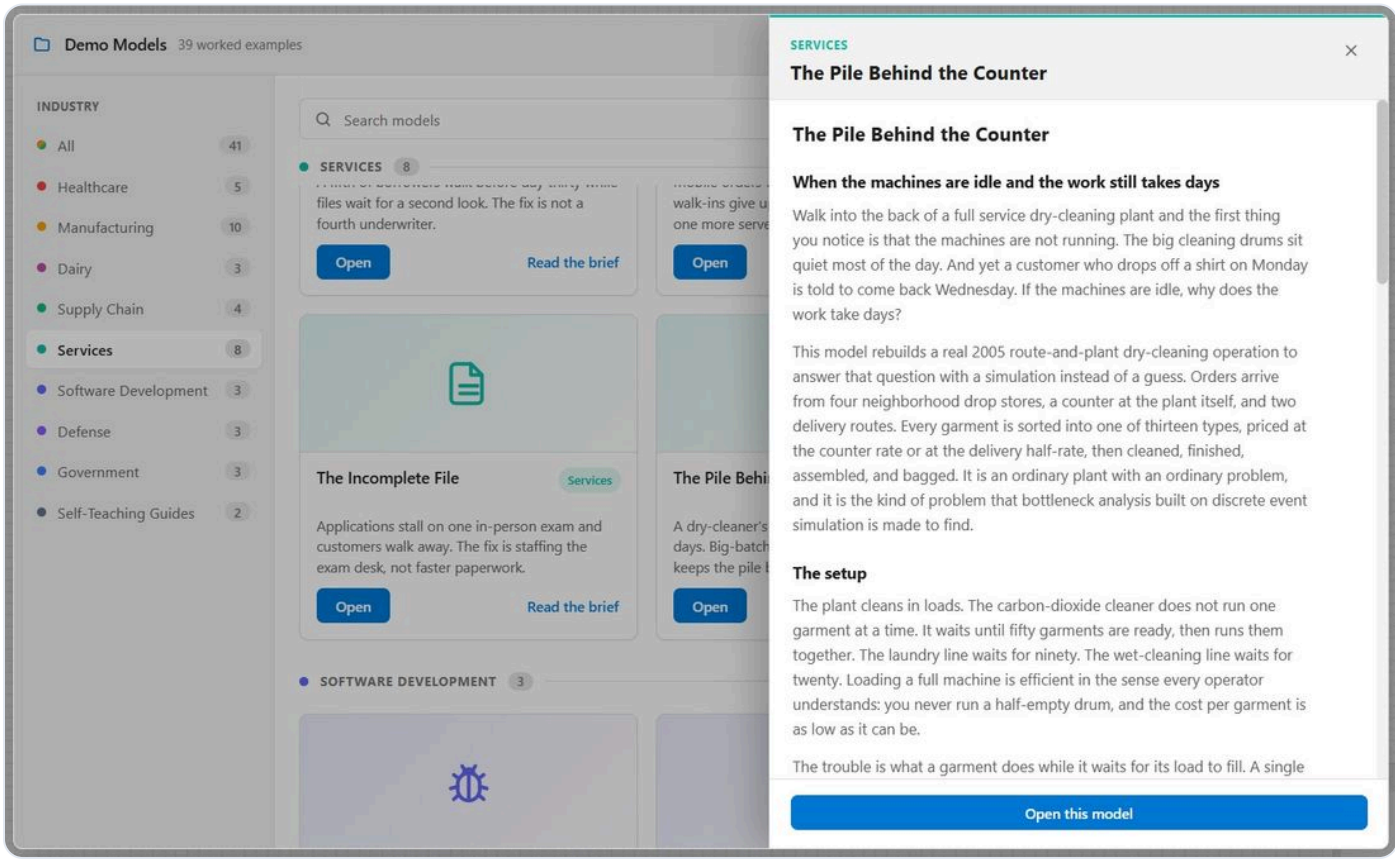



Figure 1. A demo's brief: read the story first, then open the model.

4. Click **Open this model**. Run it, change it, make it your own: the original never changes.

**When to start from a demo:** when one is close to your process. Changing the numbers and names of a working model is far faster than drawing one, and the demo already shows good practice. To get back to the Welcome screen, click **Open Model** on the toolbar.

Step 2.

## Turn on PM AI Bridge

**PM AI Bridge** lets your own AI app on this computer work in ProcessModel: read your model, change it, run it and read the results. ProcessModel does not provide the AI: you bring the AI app and its account. An AI you only use in a web browser cannot reach ProcessModel.

1. On the Welcome screen, click **Settings**.
2. Under **AI Integration**, find the **PM AI Bridge** card, "Let your own AI on this computer control ProcessModel", and turn its switch on.

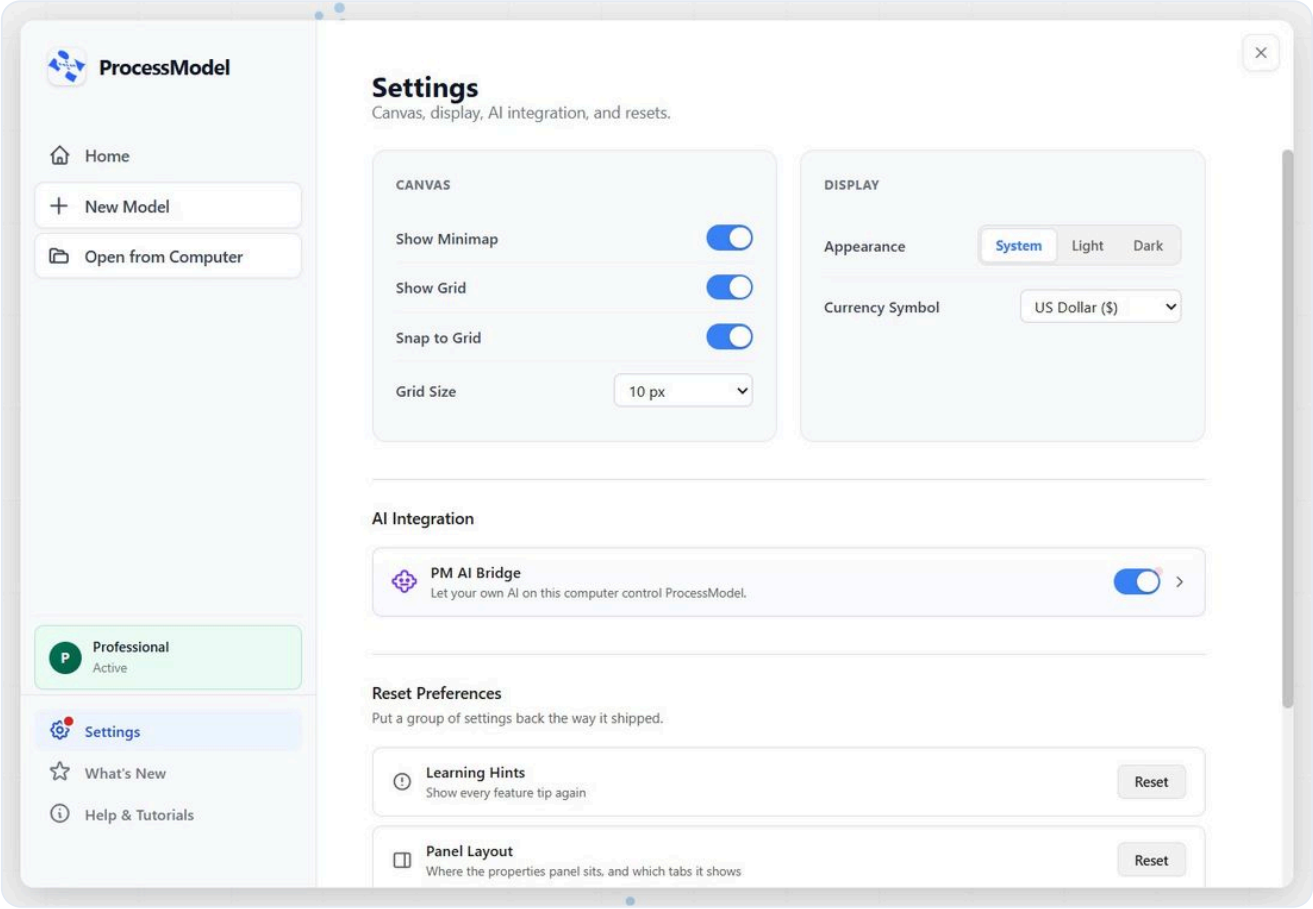



Figure 2. Settings: the PM AI Bridge switch, on.

The first time, your AI app needs to be connected to ProcessModel: open the card and use **Set up your AI**. The PM AI Bridge help walks you through it.

Step 3.

## Describe the shop in plain words

1. Click **New Model**. An empty canvas opens.
2. In your AI app, describe what you want, as you would to a colleague. In the video, the request was:

> Build a model of a small phone repair kiosk at the mall. Customers walk in about every 12 minutes from 10 AM to 8 PM. One technician checks each phone in, in 2 to 4 minutes. Seven in ten are fixed on the spot in 5 to 10 minutes; send the rest to a step called Main Shop. Run the day ten times and tell me how long customers wait.

Read that request again with what you now know. It names the entity, each step and its time as a range, a percentage split, a resource, an arrival window, and ten replications. It is exactly the information you typed by hand in Chapters 2 to 10. The clearer your words, the closer the first model. Your knowledge is what makes the request good.

Step 4.

## Watch your AI build it

Your AI adds the steps, connects them and sets the times, and you watch it happen.

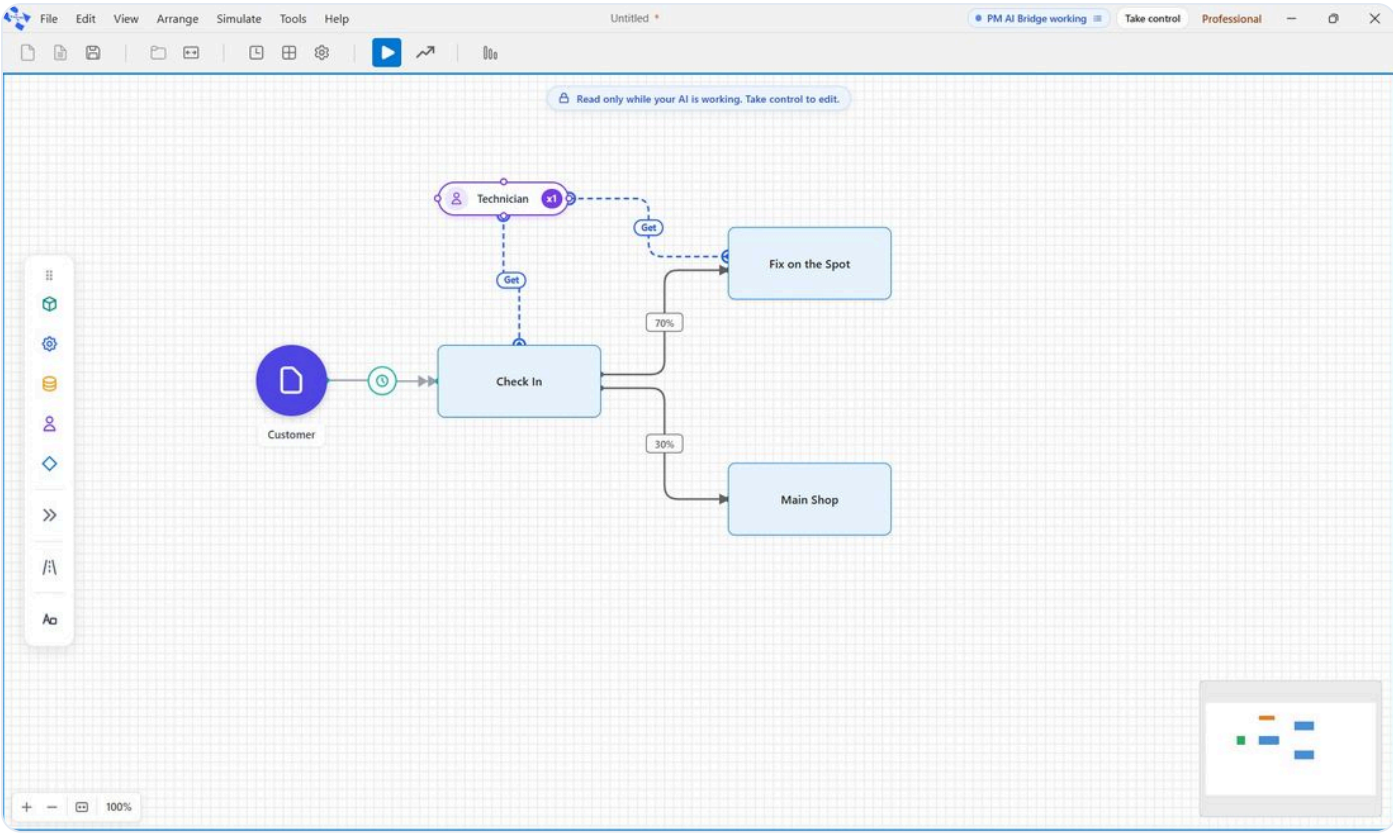



Figure 3. Your AI at work: the canvas is read only, and the title bar offers Take control.

**What you should see:**

- Each new shape is marked **Just added** as it appears.
- While your AI works, the canvas is **read only**, with a blue edge and the note “Read only while your AI is working. Take control to edit.”
- The title bar shows **PM AI Bridge working** and a **Take control** button. Press it at any moment and the model comes back to you.

Then your AI checks its own work, runs the day ten times, and opens the report. In the video its answer was that customers wait about 6 minutes to be checked in.

---

Step 5.

## See exactly what your AI did

You are responsible for any model you present, whoever built it. So check it.

1 Click the **PM AI Bridge** label in the title bar while your AI is working: it shows only then, as Take control does. A panel shows what your AI just did, and what it did before that. If your AI has finished, ask it to check the model, and click the label while it does.

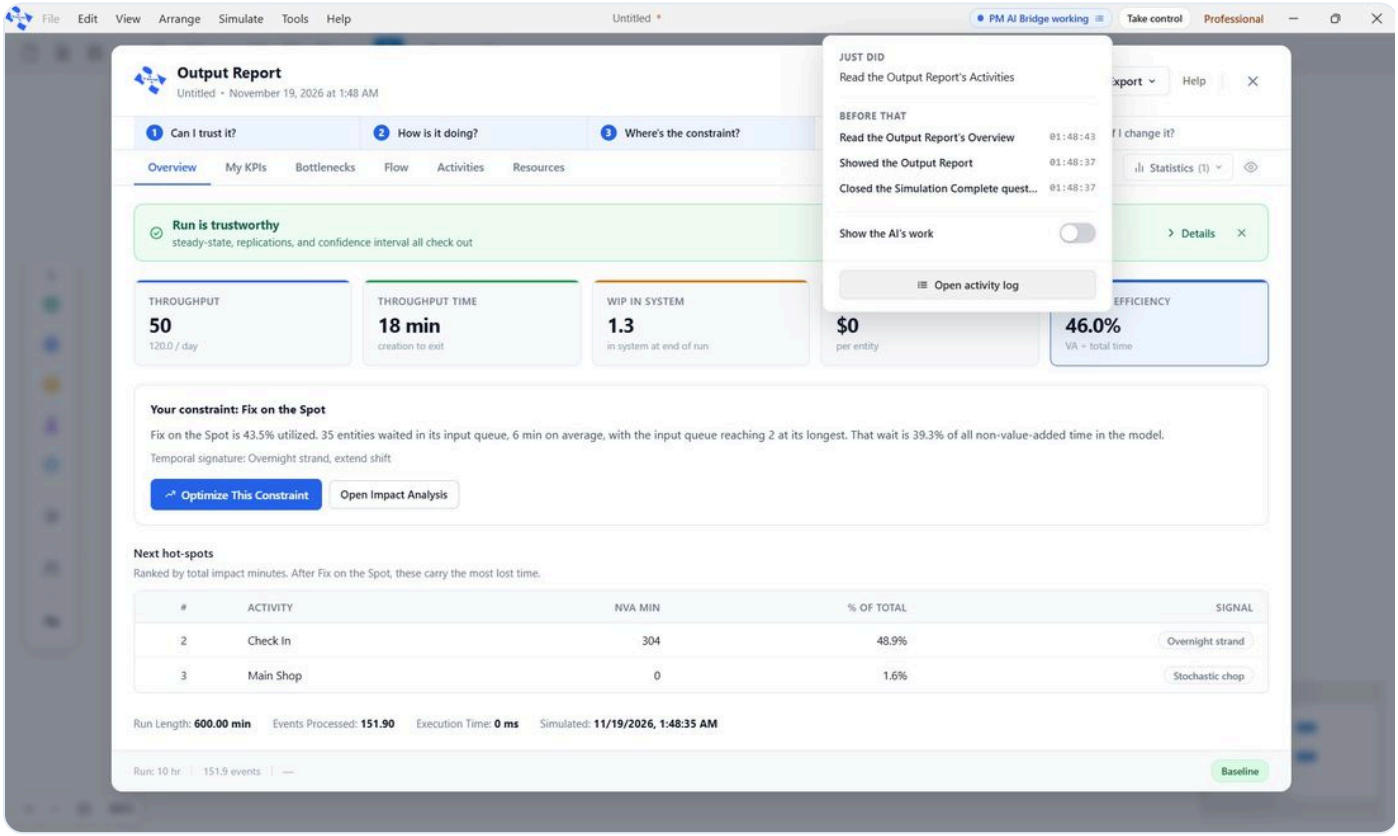



Figure 4. The PM AI Bridge panel: what your AI just did, and the way to the full log.

2 Click **Open activity log**.

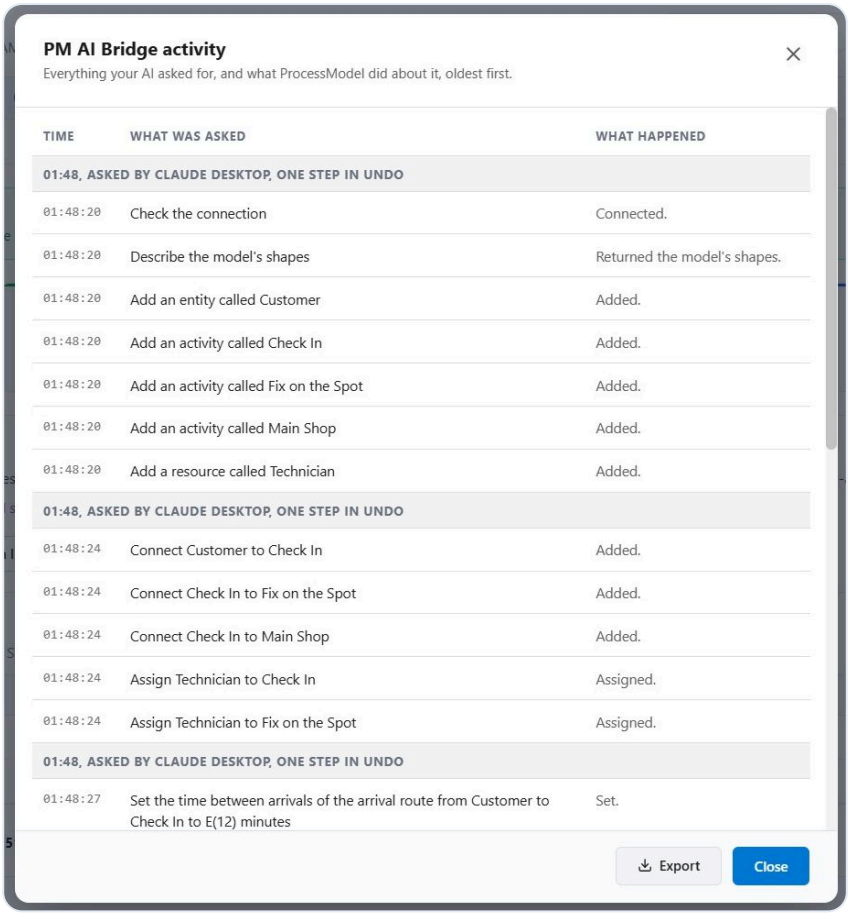



Figure 5. The activity log: every request, and what ProcessModel did about it, oldest first.

Everything your AI asked for is here, with what ProcessModel did about it, oldest first. Each stretch of work is one step in **Undo**, so you can take back a whole stretch at once.

3 Click **Close**, then close the report.

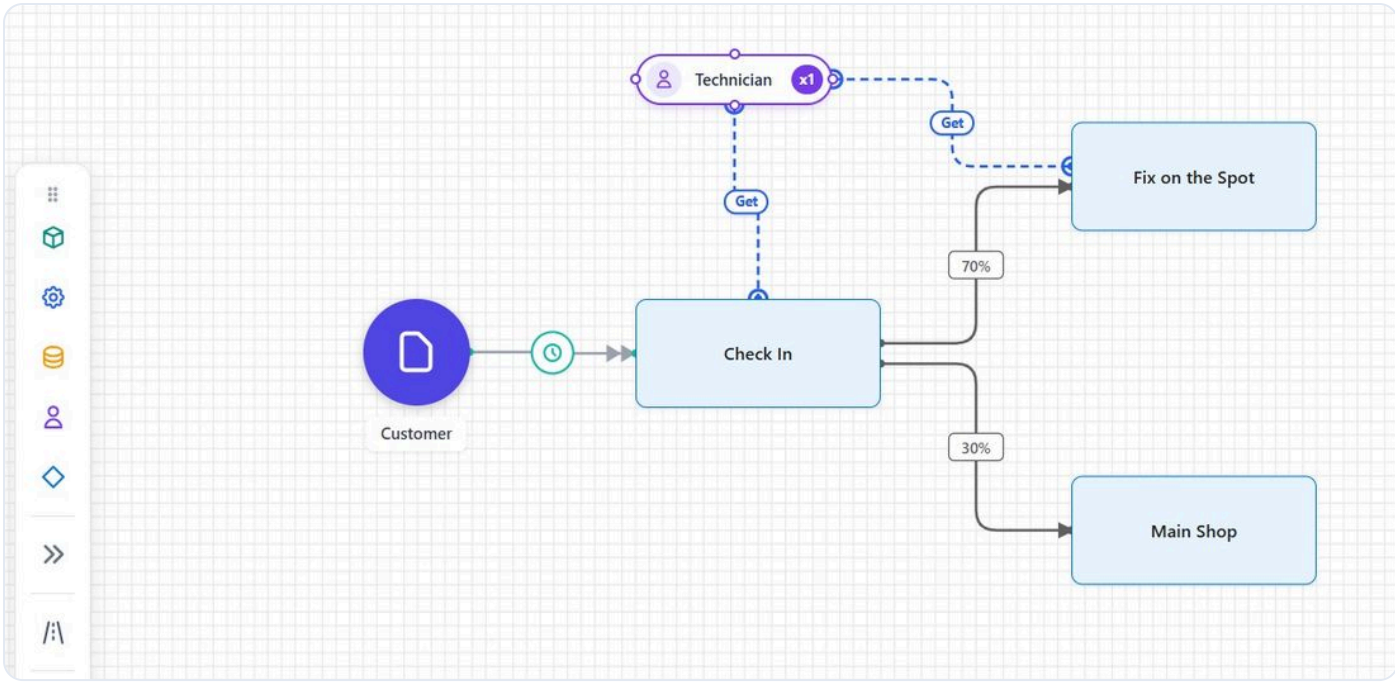



Figure 6. The shop your AI built: the same shapes and lines you would have drawn.

**Check it with your own eyes.** The entity, the steps, the 70 and 30, the Technician on Check In and Fix on the Spot: everything is a shape and a setting you know how to open and read. Double-click a few and check the times against your request.

---

Step 6.

## Save it

1. Click **File**, then **Save As**.
2. Type the name **Phone Repair Kiosk**, pick a folder, and click **Save Model**.

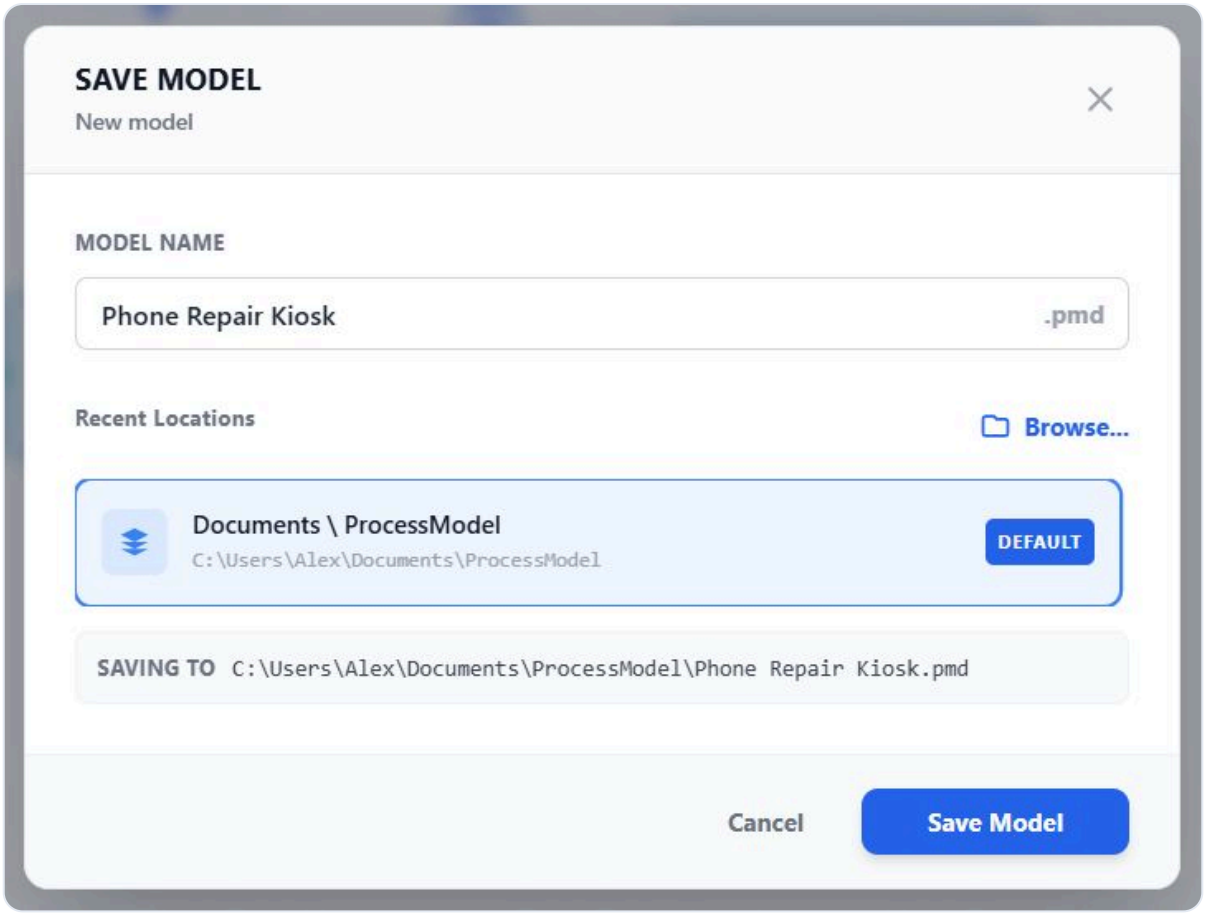



Figure 7. Save As for the kiosk: its name, and the full path under Saving to.

It is saved like any model, and it is yours.

---

Step 7.

## Two more faster starts

**An industry palette.** Click **View**, then **Show Industry Palette**. It docks a panel on the right that gathers the elements for one industry: Process plant, Software delivery or Healthcare, so a model in that field starts from its own vocabulary. See Industry palettes.

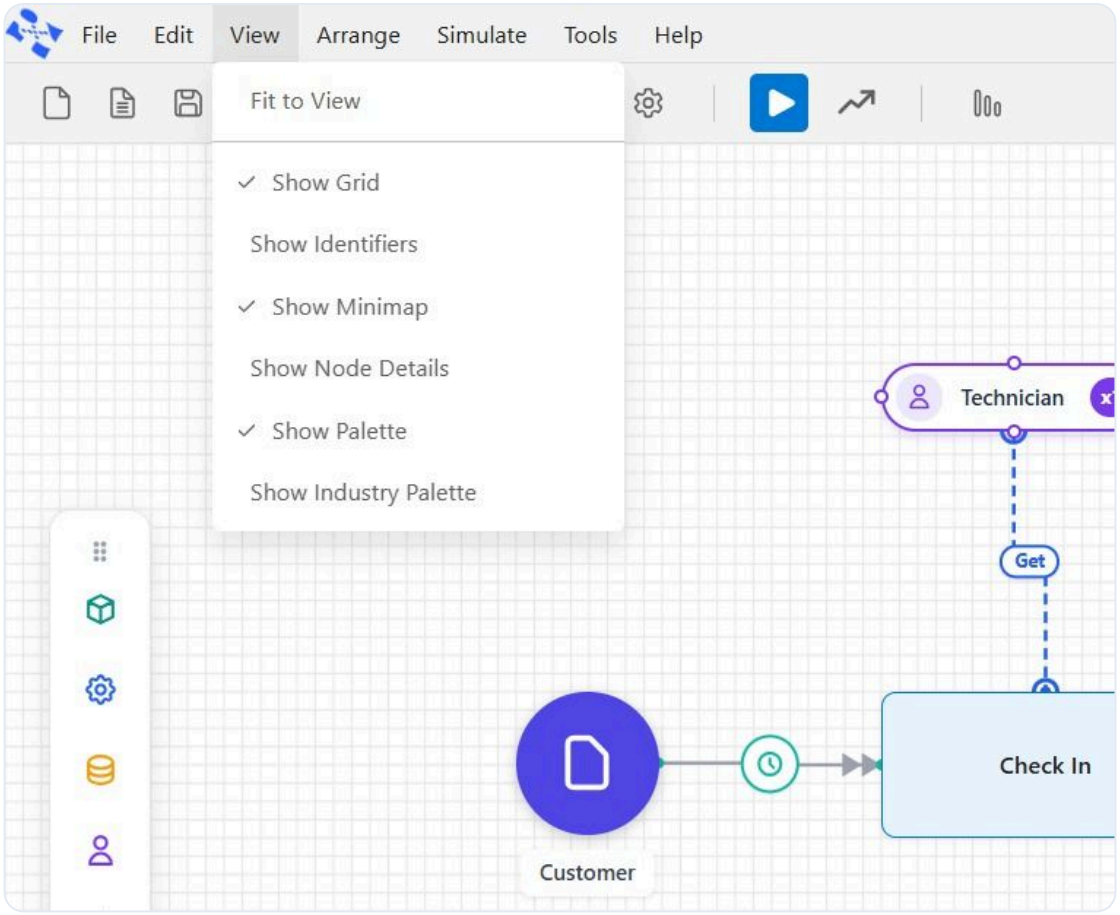



Figure 8. View ▸ Show Industry Palette.

**Import.** Click **File**, then **Import**, to bring in a model made in the earlier version of ProcessModel. See Import legacy models.

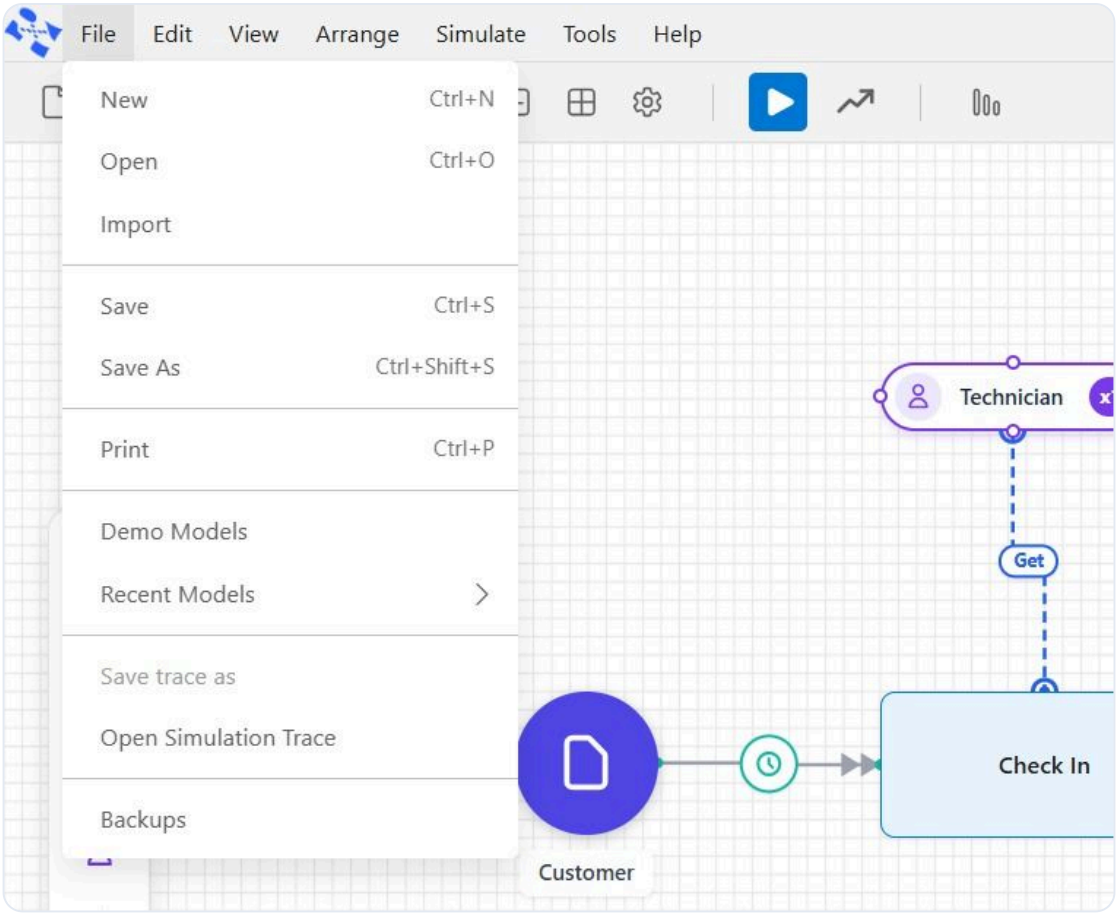



Figure 9. File ▸ Import, for a model from the earlier version.

## Your practice model

START FROM

**A new model**

New Model on the Welcome screen: your AI builds the shop on it.

COMPARE WITH

**Chapter 14. Let Your Own AI Build the Next Shop**

The phone repair kiosk the AI built in the video. Open it to compare with what your own AI builds.

The model to compare with is in **File ▸ Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**With PM AI Bridge, where does the AI come from?**

A ProcessModel includes its own AI

B It is your own AI app, installed on your computer, with your own account

C Any AI you use in a web browser

The answer is B. ProcessModel does not provide the AI. An AI app installed on the same computer works through PM AI Bridge; one you only use in a browser cannot reach it.

QUESTION 2

**Your AI is building and you spot a mistake. What do you do?**

A Wait until it finishes, then start over

B Press Take control in the title bar: the model comes back to you at once

C Close ProcessModel

The answer is B. Take control hands the model back at any moment. The activity log and Undo let you see and take back what the AI did.

QUESTION 3

**Why does what you learned in Chapters 2 to 13 still matter when an AI can build the model?**

A It does not: the AI makes it unnecessary

**B** Because a good request names what you learned to set, and you are the one who checks the model and judges its answer

C Because the AI only builds models you have already built

The answer is B. The request in this chapter is a list of the settings you learned. Checking shapes, times and reports is your job, whoever built the model.

## Try it yourself

Give your own AI a request of your own, then check its work.

1. With PM AI Bridge on and a new model open, ask your AI to build the Chapter 2 shop: customers every 10 minutes, Check In 3 minutes, Repair 12 minutes with one Technician, Pick Up 2 minutes, run for 8 hours once.
2. While it works, click the **PM AI Bridge** label and **Open activity log**, as in Step 5. When it is done, double-click Repair and the Technician to check their settings.

**What to expect.** A model like the one you built by hand in Chapter 2. When you run it, expect what Chapter 2 taught you: a line that keeps growing at Repair, because a repair takes 12 minutes and a customer comes every 10. In one run, the AI's shop gave exactly Chapter 2's first run: 39 customers, 55 minutes in the shop, 30.9% efficiency, and a count on Repair's corner that climbed to 8. If your AI's model does not show that, check its settings: you now know exactly what to look for. That is the real lesson of this chapter.

CHAPTER 15 OF 15

# Chapter 15. Build a Shop on Your Own, Then Take the Test

By the end of this chapter you can

**Build a shop on your own, check it, and take the certification test.**

Watch the video (5 minutes), then work through the lesson below it in ProcessModel, step by step.

**ProcessModel 7 Self Teach Guide, Chapter 15. Build a Shop on Your Own, Then Take the Test**
www.youtube.com/watch?v=Yu4VBA6UfIo

## Before you start

Fourteen chapters built one shop, a step at a time, with the steps written out for you. This last chapter takes the steps away. You get a **brief**, the way a real project starts, and you build the model **on your own**, run it, and answer three questions. Then you check your answer against a worked one, and take the certification test.

Do not skip ahead to the answer. The point of this chapter is the moment you get stuck and work it out: that is when what you learned becomes yours. Every part of the brief was taught in an earlier chapter, and each line says which, so you know where to look.

**You need** about an hour, and nothing else: you start from a Blank Canvas.

Step 1.

## Read the brief

**The Screen and Battery Bar.** The shop opens a second store that does two jobs only: new batteries and new screens. The numbers are made up for this course.

1. **Customers:** 50 on Monday, in business hours, busiest at noon: 30 percent of them come between noon and 1. *(Chapter 4)*
2. **Check In:** 2 to 5 minutes, mostly 3. No staff. *(Chapters 2 and 3)*

3. **The jobs:** 6 in 10 need a **New Battery**: 5 to 15 minutes, mostly 8. The rest need a **New Screen**: 20 to 40 minutes, mostly 30. Each needs a Technician. *(Chapters 3 and 5)*
4. **One screen bench:** New Screen works on one customer at a time. Screen customers wait in a **Screen Queue** that holds 3, and nowhere else. *(Chapter 6)*
5. **Staff:** two Technicians, on all day, doing both jobs. *(Chapter 2)*
6. **Pick Up:** 2 minutes. No staff. *(Chapter 2)*
7. **Run** the whole Monday, ten times. *(Chapters 4 and 10)*

**Then answer:**

- How long is a customer in the store, on average?
- Where do they wait longest, and why?
- Try a **second screen bench** as a what-if. How much time does it save? *(Chapters 11 and 12)*

---

Step 2.

## Build it on your own

Start from **Blank Canvas** on the Welcome screen. There is no single right order, but this one works, and each part is something you have done before:

1. The customer, then the five steps: Check In, New Battery, Screen Queue, New Screen and Pick Up, then the Technician with Quantity 2.
2. The times, as distributions where the brief gives a range.
3. The arrival as a Daily Pattern for Monday.
4. The routes: two Percent routes out of Check In, and the routes on to Pick Up.
5. The Technician's lines to the two jobs.
6. The two details that are easy to miss: the Screen Queue's capacity, and New Screen's own waiting room.
7. The run length and the replications.

When you get stuck, go back to the chapter the brief names, find the step, and come back. That is not cheating: it is exactly how people who model for a living work.

Step 3.

## Run it and answer the three questions

1. Run it, open the report, and check the banner before anything else (Chapter 11).
2. Write down your three answers before you look at Step 4: the average time in the store, where customers wait longest and why, and how much a second screen bench saves.

For the what-if, build a scenario that changes one setting, as in Chapter 12. Which setting gives the shop a second screen bench is part of the exercise.

Step 4.

## Check your answer

Here is one good answer. Yours may look different: a different layout, other names or another look all count as right.

1. Open the worked model, **Screen and Battery Bar**, from the practice models below.

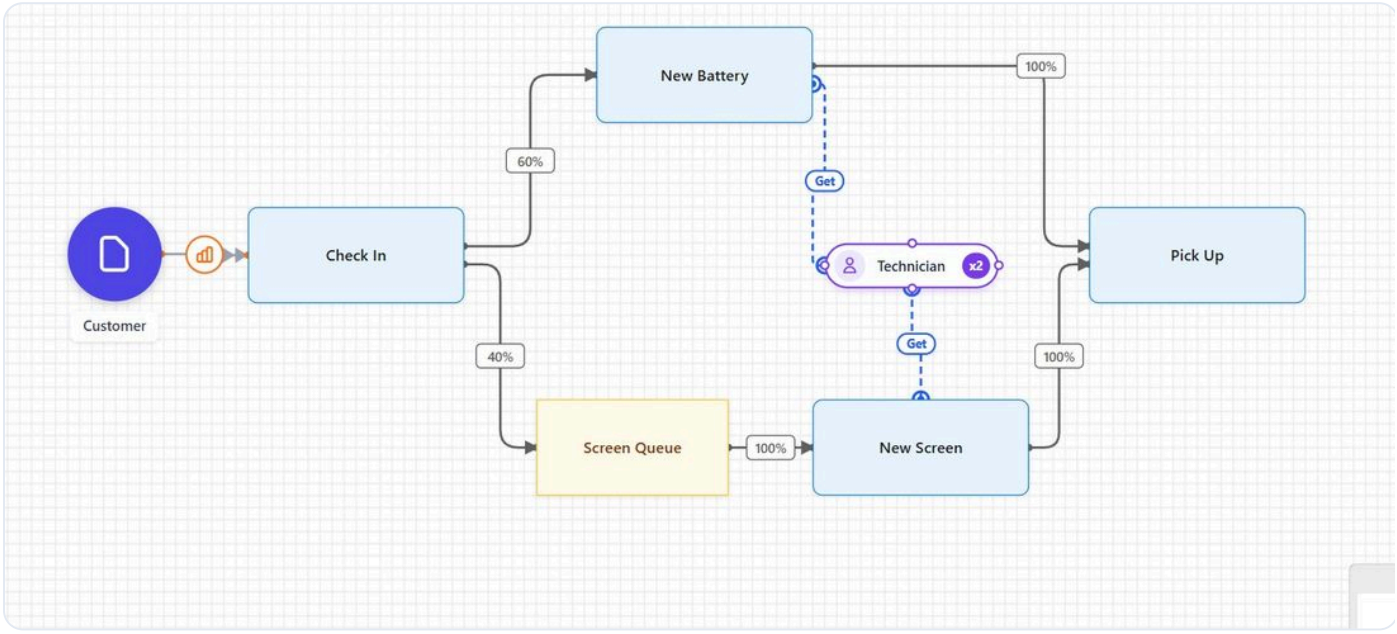



Figure 1. One good answer: the steps, the 60 and 40, and two Technicians on both jobs.

Check that yours has the same steps, routes and numbers. Customers arrive on a daily pattern at Check In, which sends 60 percent to New Battery and 40 percent to the Screen Queue. Screen customers wait in the

queue for New Screen. Both jobs lead to Pick Up, and the two Technicians do both jobs.

2 Run it, and open the report.

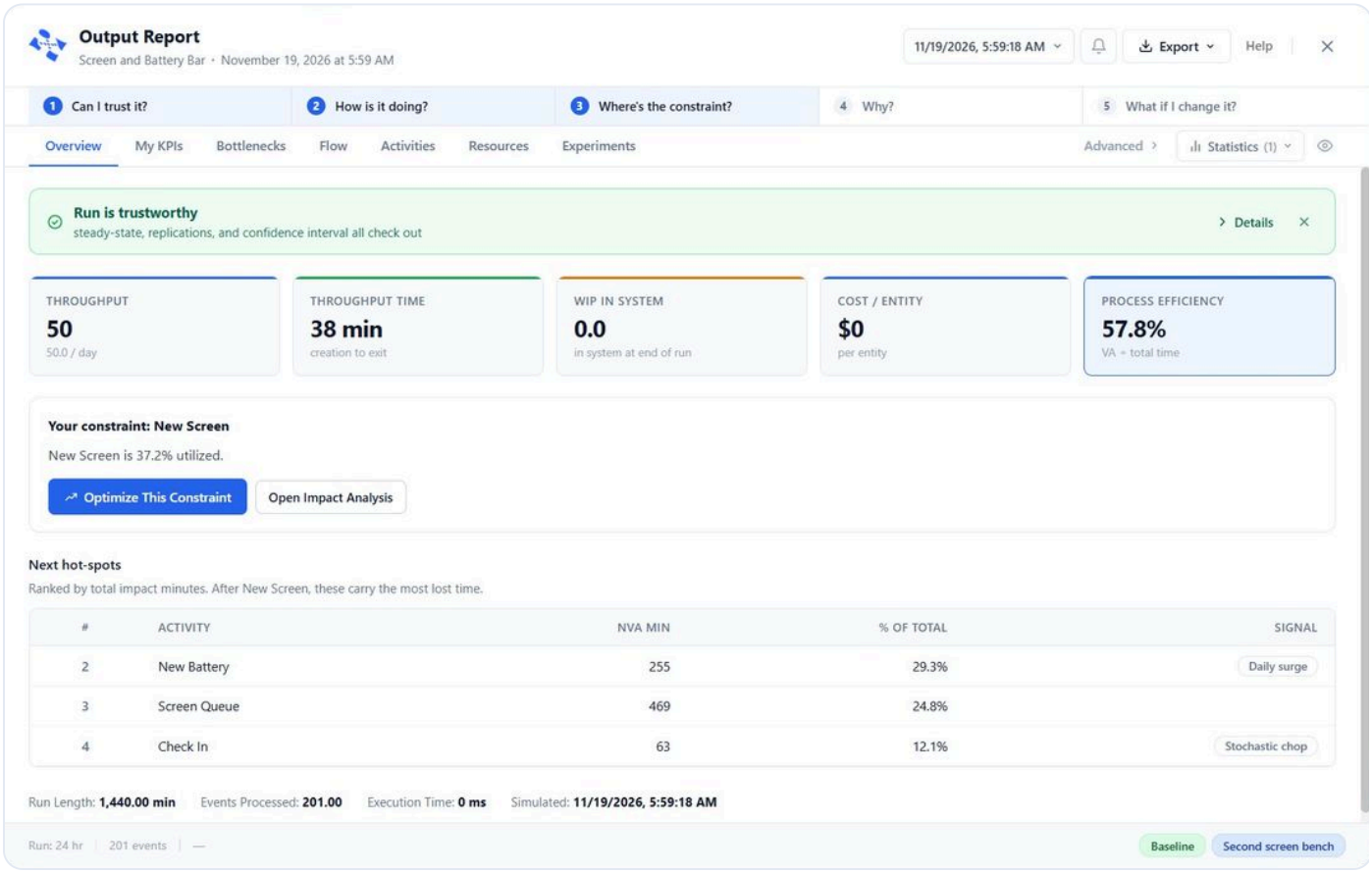



Figure 2. The worked answer: every customer served, 38 minutes in the store, and the constraint is New Screen.

**How long is a customer in the store?** In the worked model, **38 minutes**. Throughput is **50** and WIP is **0**: every customer is served. Each model draws its own random numbers, so **anywhere from about 38 to 43 minutes is right**.

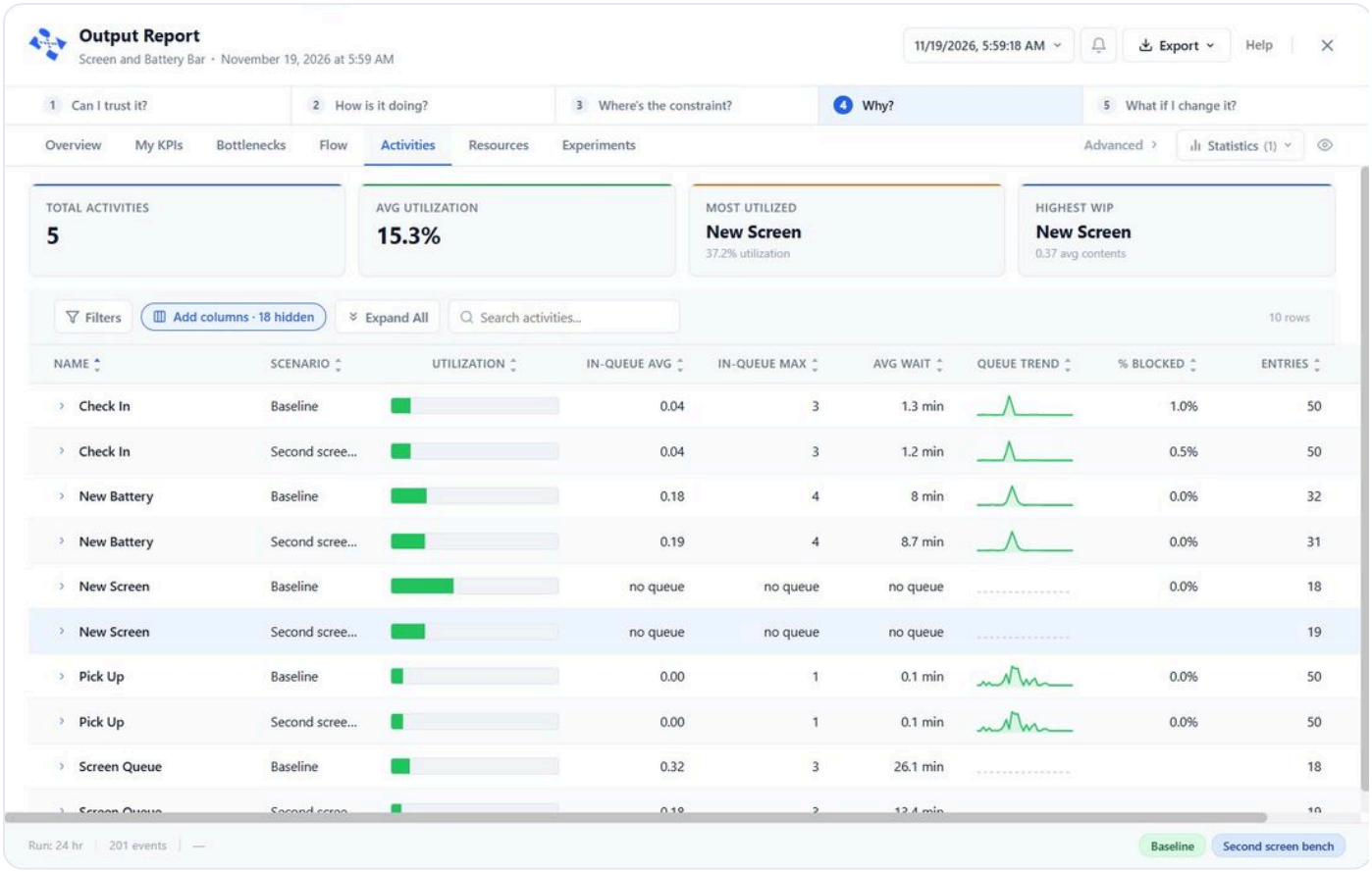



Figure 3. Why: New Screen is the most utilized step, and screen customers wait about 26 minutes in the Screen Queue.

**Where do they wait longest, and why?** At the **Screen Queue**, about **26 minutes**, for **New Screen**, the constraint. There is one screen bench, and a screen takes 20 to 40 minutes, so screen customers queue for it. The limit is the bench, not the number of people, just as it was the parts, not the people, in Chapter 11.

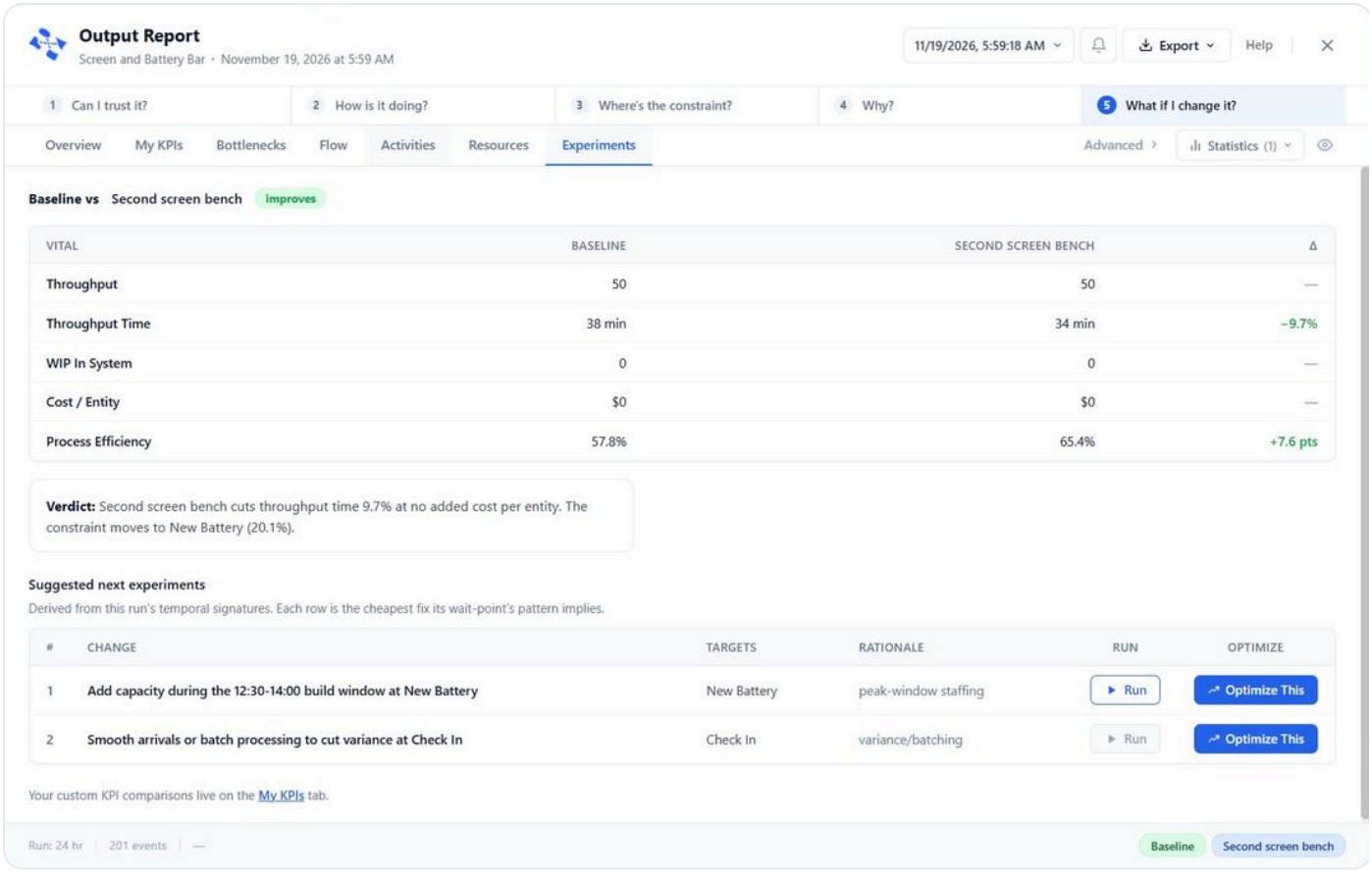



Figure 4. The fix: a second screen bench cuts the time in the store from 38 to 34 minutes.

**How much does a second screen bench save?** In the worked model the scenario sets **New Screen's Capacity to 2**, and the time in the store falls from **38 to 34 minutes**, about 10 percent. Across different random numbers, a cut of about 9 to 15 percent is right. Notice the verdict's last words: the constraint **moves** to New Battery. Fix one bottleneck and the next one shows itself. That is how improvement really goes, one constraint at a time.

If your model lands near these numbers, you built it right.

---

## Go further

Three extras, unchecked, for when you want more. Each uses a later chapter:

- Tag each customer with the phone's age, and send older phones' customers straight to New Screen (Chapters 5 and 8).
- Give New Screen a shelf of screens that runs low and reorders (Chapter 9).
- Let PM Optimizer find the fewest screen benches that keep the wait under 30 minutes (Chapter 13).

The two industry guides are electives too: Building a Software Development Model and Building a Fluid Model.

## Take the certification test

You have finished the ProcessModel 7 Self Teach Guide. Now show it.

**All registered ProcessModel users can take the test.** The certification page explains how it works, and the test starts there.

## Your practice model

START FROM

**An empty canvas**

Blank Canvas on the Welcome screen: you build the shop on your own, from the brief.

COMPARE WITH

**Chapter 15. Build a Shop on Your Own**

One good answer to the brief. Open it only once yours is built and run.

The model to compare with is in **File** ▸ **Demo Models**, in the **Self Teach Guide** group. A demo model opens as a new model, so the original never changes.

## Check yourself

QUESTION 1

**Your model gives 41 minutes in the store; the worked answer says 38. Did you build it wrong?**

A. Yes: any difference means a mistake

**B.** Not necessarily: each model draws its own random numbers, and anywhere from about 38 to 43 minutes is right

C. Yes, unless you used the same layout

The answer is B. Two correct models of the same shop can differ a little, because their random numbers differ. Check the steps, routes and settings, then the band.

QUESTION 2

**A second screen bench helps, and the verdict says the constraint moves to New Battery. What does that tell you?**

A. That the second bench was a mistake

**B.** That fixing one bottleneck reveals the next: improvement goes one constraint at a time

C. That New Battery was always the real problem

The answer is B. Once the screen bench keeps up, the next slowest part of the process becomes the limit. The report tells you where to look next.

QUESTION 3

**Where do customers wait longest in the Screen and Battery Bar, and why?**

A At Check In, because it has no staff

B In the Screen Queue, because there is one screen bench and a screen takes 20 to 40 minutes

C At Pick Up, because everyone passes through it

The answer is B. New Screen is the constraint: one bench, and long jobs, so screen customers queue for it.

## Try it yourself

Do one of the **Go further** extras above on your own model, then compare it with the brief's answer.

**What to expect.** There is no worked answer for the extras, on purpose. Judge your own result the way this guide taught you: trust the run first, read where customers wait and why, then try a fix side by side before you believe it. If you can do that unaided, you can model a real process.