Chapter 1 – Why Anyone Pays for a Digital Twin: Value, Markets, and Real Deployments

1.0 Before you start

What you are assumed to know. You write software for a living. You are comfortable with HTTP APIs, JSON, a time-series or relational database, and reading Python you did not write. You have seen a system that talks to hardware, even if you did not build it.

What you are *not* assumed to know. Nothing about control theory, numerical simulation, soil physics, or the digital-twin literature. Where this chapter needs a term from those fields, it defines it at first use.

What this chapter covers. Why organisations spend money on digital twins; how that spending is justified before a line of code exists; what the market numbers do and do not tell you; what four real deployment families look like when you read them for their *value* argument rather than their technology; and how to build the value case for this book's running example, a pot of soil with a moisture sensor and a pump.

What this chapter deliberately does not cover. What a digital twin *is*, technically. That sounds like the wrong order, and it is a deliberate choice. Chapter 2 draws the line between a twin, a shadow, a model and a simulation, and Chapter 3 gives the architecture. Both are easier to follow once you know what the thing is for. Until Chapter 2 arrives, this chapter runs on one working sentence:

> **Digital twin (working definition, Part I only).** Software that keeps a model of one specific physical system in step with that system's actual state, and uses it to control that system and answer questions about it that measurement alone cannot answer.

That sentence is deliberately loose. Chapter 2 will tighten it, and the tightening matters – the literature does not agree on where a *digital twin* ends and a *digital shadow* begins, and that boundary carries real cost consequences [1]. For now, the loose version is enough to talk about money.

Two more terms, pinned now and used unchanged for the rest of the book. The **Physical Twin (PT)** is the real system being twinned – in our running example, the pot, the plant, the soil, the pump, the tubing and the Raspberry Pi that samples the sensors. The **Digital Twin (DT)** is the software defined above. From here on this chapter writes PT and DT rather than spelling either out, and so does the rest of the book. When this book says "the twin" without qualification, it means the digital one.

Learning objectives

By the end of this chapter, you will be able to:

1. **State** the decision a proposed DT is supposed to improve, and **express** it as a single measurable value metric with a baseline and a measurement method.
2. **Classify** a proposed twin into one of five value patterns – *monitor, diagnose, predict, decide, certify-and-train* – and **predict** which engineering obligations that classification imposes on your team.

3. **Estimate** the cost structure of a twin project and **identify** which cost dominates for a given deployment.
4. **Critique** a DT proposal by naming the conditions under which it will not pay back, and **state** the measurement that would falsify the proposal before it is built.
5. **Derive** a complete value case for the plant/pot/pump demonstrator, including metric, baseline, value pattern, cost estimate and payback period.

Everything in this chapter serves one of those five.

1.1 The meeting you are about to be in

Here is the situation this chapter is written for.

You are a software engineer. Someone senior has decided that your organisation needs a DT. There is a kickoff meeting. Around the table are a modelling expert who talks about solvers and boundary conditions, a business sponsor who talks about uptime and margin, and a customer who talks about the thing they actually own and worry about. You are there because someone has to build it.

The failure mode of that meeting is not technical. It is that everyone agrees enthusiastically while meaning four different things, and nobody says out loud which decision the twin is supposed to improve. Six months later the project has a beautiful three-dimensional model, a live data feed, a dashboard, and no answer to the question "what did we do differently because of it?"

This is not a hypothetical failure. Practitioners writing about supply-chain twins put it plainly: what DTs can and cannot deliver has to be set out explicitly, because organisational confusion on the point is common, and expectations about return on investment and payback speed need to stay realistic [2]. A study of pilot projects across European manufacturing found that the pilots which went well were the ones where the company had clearly envisioned *why* it wanted the twin and what the use of it was, before starting [3].

So the first skill this book teaches is not modelling and not architecture. It is the ability to sit in that meeting and ask, without embarrassment: **which decision gets better, who makes it, and how will we know?**

Where the idea came from, in one paragraph

You will hear the origin story in meetings, so it is worth having straight. The concept traces to a 2002 presentation at the University of Michigan, in which Michael Grieves proposed what he then called a *Mirrored Spaces Model* for product lifecycle management: a real space, a virtual space, and a data link between them [4]. The name "digital twin" attached later, and the 2010 technology roadmap of the National Aeronautics and Space Administration (NASA) gave the first detailed definition in the aerospace context [5]. The folklore version – that NASA built two identical Apollo vehicles, flew one and kept the other on the ground to rehearse fixes – is real enough as history and is routinely cited as the ancestor of the idea [6]. It is also a useful reminder that the *value* came first: the

ground vehicle existed so that engineers could try a repair somewhere it was safe to be wrong.

Note what the Apollo story is really about. It is not about fidelity for its own sake. It is about moving a decision – "will this fix work?" – from a place where being wrong is fatal to a place where being wrong is cheap. Nearly every DT value argument in this chapter is a variant of that one move.

1.2 The question behind every twin project

Start from a claim that will look reductive and then earn its keep:

> **A DT is bought to improve a decision. If you cannot name the decision, there is no value case – only a budget.**

A decision has four parts you can write down in a meeting:

1. **The decision itself.** "How much water does pot 7 get this evening?" "Do we take turbine 12 offline this week or next month?" "Do we accept this batch?"
2. **The decider, and the cadence.** A human once a quarter, a human every morning, or a control loop every 200 milliseconds. These are three completely different systems, and the cadence drives most of your architecture.
3. **What is wrong with the decision today.** It is made too late, or on incomplete information, or by a person who has to be physically present, or it is not made at all because nobody noticed there was a decision to make.
4. **What it costs to be wrong.** In currency, in downtime, in scrapped product, in a ruined six-week experiment, in a life.

Point 4 is the one that gets skipped, and it is the one that determines whether the project is worth doing. A twin that improves a decision whose error cost is EUR 200 a year is a hobby.

The value metric

From those four parts you extract exactly one **value metric**: a measurable quantity, with a stated baseline and a stated way of measuring it, that the twin is supposed to move.

Insisting on *one* metric is the core of what practitioners call the **lean digital twin**, whose stated aim is a single business-value metric carrying a financial benefit that, in the authors' words, "can be verified in a short period of time". The increment that verification pays for is then what funds the one after it [7]. That is a familiar shape to any engineer who has shipped a minimum viable product, and the analogy is deliberate on the authors' part.

Bad value metrics are abstract nouns: *efficiency*, *visibility*, *optimisation*, *digital transformation*. Good value metrics have units and a denominator:

Weak	Strong
"Improve maintenance"	"Unplanned downtime hours per turbine per year"

Weak	Strong
"Better watering"	"Experiment-weeks lost to undetected watering faults per year"
"More visibility"	"Hours between a fault occurring and an engineer being told"
"Optimise the line"	"Scrapped units per 10,000 produced"

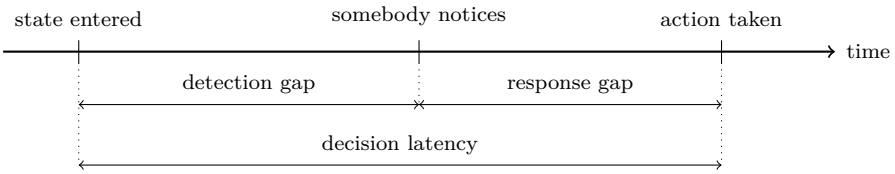
The right-hand column has a property the left does not: someone can go and measure it *today*, before you build anything, and get the **baseline**. A value claim with no measured baseline is subtraction with one operand missing.

> **Terminology note.** This book uses **baseline** for the measured present-day value of the metric, and **payback period** for (build cost + one year of running cost) / (annual benefit), expressed in months. Both are used with these meanings throughout.

Decision latency: what twins usually actually sell

Look across the deployments in Sec. 1.6 and one quantity keeps reappearing: **decision latency**, the time between the PT entering some state and someone (or something) acting on it. Sometimes, twins can make a previously improbable decision to a probable one by helping create a new service suited for the required decision making metric. However, more often they make an already-possible decision happen *hours or weeks earlier*, or *without a human standing next to the equipment.*

Decision latency splits into two gaps, and a DT does not touch both of them equally. Figure 1.1 separates them.



Plant demonstrator today : ∼4 days + ∼1 day
With a DT that diagnoses : **∼8 hours** + ∼1 day

the gap a DT actually closes; the response gap belongs to the organisation, not to you

Figure 1.1 Decision latency, split into the gap a twin closes and the gap it does not.

The detection gap is an engineering problem and yours to solve. The response gap is an organisational one – how fast the lab, the crew or the maintenance schedule can act once told – and no amount of fidelity shortens it. This is why Sec. 1.3's *predict* pattern carries an honest warning, and why Sec. 1.7 lists "nobody can act on the output" as a reason to stop.

That split is worth drawing in the kickoff meeting, because it reframes the project from "build a virtual replica" – unbounded, expensive, un-finishable – to "cut the time from

event to action from four days to four hours" – bounded, estimable, testable.

1.3 Five value patterns

Across the corpus of deployment reports, the *shapes* of value repeat. This book uses five names for them, and will use them consistently from here to Chapter 15. They form a rough ladder: each rung typically needs everything below it, and each rung costs more than the one below. Figure 1.2 is the whole section on one page; the subsections that follow are its rungs, in order.

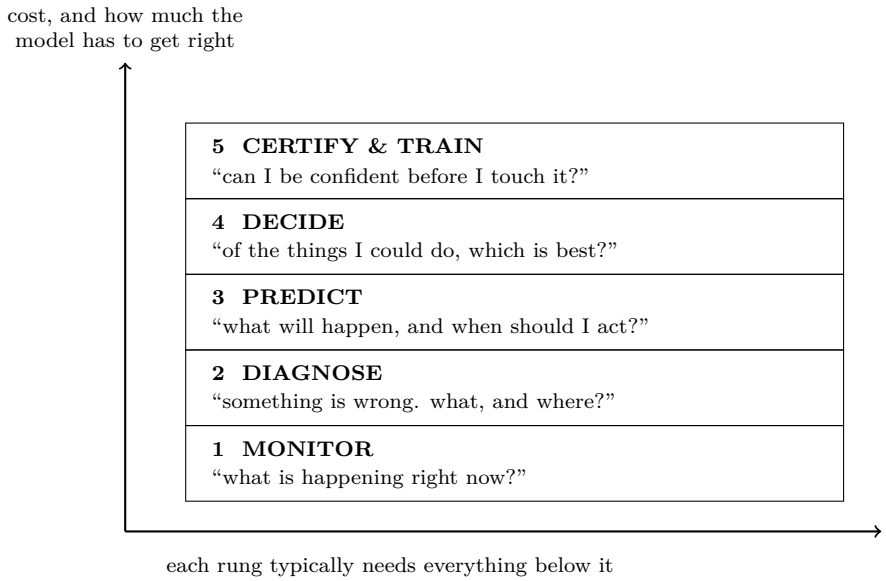


Figure 1.2 The five value patterns as a ladder. It is a vocabulary, not a maturity model: stop at the rung that pays for itself.

The ladder is a cost ordering, not a quality ordering. Rung 2 delivered is worth more than rung 4 proposed.

Pattern 1 – Monitor

"What is happening right now, and what happened last Tuesday?"

The twin ingests live measurements, stores them with context, and presents current and historical state. It may fill in quantities that are not directly measured – a **virtual sensor** is a model-derived estimate of a quantity you have no physical sensor for, such as a stress at a weld you cannot instrument [8].

- **Buys you:** lower decision latency, remote access, an audit trail.
- **Costs you:** connectivity, data engineering, and the unglamorous work of keeping timestamps and units correct (Chapters 9 and 10).
- **Honest warning:** many systems sold as DTs stop here.

Pattern 2 – Diagnose

"Something is wrong. What, and where?"

The twin compares what the model says *should* be happening with what the sensors say *is* happening. The difference – the **residual** – is the signal. A large, structured residual localises a fault.

This is the pattern behind diagnostic twins for floating offshore wind, where the twin monitors and analyses asset condition to detect anomalies on equipment that is expensive and slow to visit [9], and behind fault-diagnosis twins for rotating equipment [10].

- **Buys you:** faults caught in hours instead of days; fewer truck rolls to find out nothing is wrong.
- **Costs you:** a model good enough that a residual means something, plus the calibration work of Chapter 7.
- **Note for the meeting:** diagnosis is the cheapest rung with real engineering value, and it is very often the right first increment. It needs a model that gets the *direction and rough size* of a change right. It does not need a model that is accurate to three decimal places.

Pattern 3 – Predict

"What will happen, and when should I act?"

The twin runs the model forward from the current estimated state to forecast future condition – remaining useful life, time-to-failure, time until the soil dries below a threshold. This is by far the most-reported family in the literature: predictive maintenance using DTs has its own systematic literature reviews [11], [12].

- **Buys you:** maintenance scheduled by condition instead of by calendar; interventions before the failure rather than after.
- **Costs you:** everything in *diagnose*, plus a model that stays accurate over a forecast horizon, plus a serious answer to "how wrong might this be?" Uncertainty quantification stops being optional here [13].
- **Honest warning:** a prediction nobody acts on has zero value. If the maintenance crew's schedule is fixed six months ahead, a two-week-ahead prediction changes nothing. Check the *organisation's* ability to respond before you sell a prediction.

Pattern 4 – Decide

"Of these five things I could do, which is best?"

The twin runs many alternative futures and compares them: **what-if analysis**, design-space exploration, fault injection, and optimisation over a control setting [14]. In the strongest form the twin closes the loop and changes the PT's behaviour directly.

A fleet is a set of similar twins like a fleet of twins for steam turbines. At the fleet level this is where large operators claim their money back. In the reported analysis of General Electric's D11 steam-turbine twins, per-component predictive cost evaluation is what makes plant-wide cost control approachable – balancing revenue against maintenance cost across a fleet rather than one machine at a time [15].

- **Buys you:** better decisions, and decisions a human could not have enumerated by hand.

- **Costs you:** simulation you can run *fast* and *many times*. This is the first rung where compute cost is a design constraint; documented case studies specify things like "hundreds of simulations within a few seconds" as a hard requirement [16]. Chapter 6 is about how that is made possible.

Pattern 5 – Certify and train

"Can I be confident before I touch the real thing?"

The twin substitutes for the physical system when using the physical system is expensive, slow, dangerous, or not yet built. Two sub-cases matter to you specifically as a software engineer:

- **Testing the software that controls the PT.** A twin of the vehicle lets you build and exercise the control software – communication protocols included – with no physical hardware present, which cuts cost and removes the risk of destroying hardware during development [17]. Interview studies of continuous integration for cyber-physical systems show teams doing exactly this in production pipelines, sometimes with customer-mandated simulators [18].
- **Training people.** Industry treats trained operators as a direct benefit, and the training cost of sending staff away is itself a line item the twin displaces [19].
- **Buys you:** a test environment, a rehearsal space, and – in regulated domains – evidence.
- **Costs you:** credibility. If you will make decisions on the twin's output, you must be able to say why it should be believed. That is what most projects underestimate.

Using the patterns

The patterns are a vocabulary, not a maturity model to be climbed for its own sake. Two practical uses:

- **Diagnosing a request.** When a customer says "we want a digital twin", ask which of the five they mean. The answers imply wildly different budgets.
- **Sizing the first increment.** *Monitor* and *diagnose* are usually reachable in a single quarter. *Decide* rarely is. Deliver a rung that pays for itself, then climb [7].

1.4 Where the money actually goes

Twins are not free, but the return can be large – that is the industrial sales pitch in one line, and it is a fair summary of the pitch [20]. Your job is to know which half of that sentence you are being sold. Here is where the cost sits, roughly ordered by how often it is the surprise.

1. Getting the data out, and keeping it correct. Connectivity, protocol translation, timestamp discipline, units, and the context that says *which* pot a moisture reading belongs to. This is your home turf and it is consistently underestimated. Reported pilots identify data curation and labelling as a cost that persists even after the technical setup is done [3].

2. Building and – worse – maintaining the model. The first version of a model is a project. Keeping it valid as the PT wears, gets repaired, or has a component swapped is a *permanent* cost. The twin has to track both continuous change such as wear and discontinuous change such as a replaced component [21]. It is the cost most business cases omit entirely.

3. People who can do this. Surveys of process-industry adoption repeatedly name the shortage of specialists and expertise as a barrier, and name workforce upskilling as the corresponding enabler [22]. In healthcare, barriers to adoption are high.

4. Compute and licences. Real, and the least alarming of the four: hardware cost per unit of computation has been falling and is generally treated as a mitigating factor rather than a blocker [22]. Operating expenditure often shows up as software licensing rather than machines [3]. Simulator reuse – third-party or customer-supplied simulators instead of writing your own – is a documented cost-reduction tactic [18].

5. Bespoke integration, over and over. The structural problem of the field: most twins are hand-built one-offs. The explicit call to move "from the artisanal to the industrial" argues that scale requires a drastic reduction in the technical barriers to adoption [23]. Every hour you spend re-inventing an interface that a standard already defines is cost with no value attached; standardised reference architectures such as ISO 23247 exist precisely to stop that [24].

The shape of the estimate

For a first-increment twin, a serviceable rule for the kickoff meeting is this: **expect the integration and data work to be the largest single line, expect model maintenance to be the largest line you forgot, and expect compute to be smaller than everyone fears.** Then go and measure, because a rule of thumb is not an estimate.

1.5 The market, and how to read it

You will be shown a market forecast. Here is how to read one without either swallowing it or sneering at it.

The published figures are large and mutually inconsistent. A 2020 state-of-the-art survey cites a forecast of the DT market growing to about USD 27 billion by 2025 from about USD 2.3 billion in 2017, and notes that Gartner named DTs a top-ten strategic technology trend for 2019 [25]. A 2022 survey cites a different analyst's figure: USD 3.2 billion in 2020 to USD 48.3 billion by 2026, a compound annual growth rate of 58% [26]. A review published the same year takes its figures from yet another house: USD 48.2 billion by 2026, up from USD 3.1 billion in 2020 [27]. A 2025 paper cites USD 8.6 billion in 2022 growing to USD 137.7 billion by 2030 at 42.6% annually [28].

Four observations, in order of usefulness to you:

1. **These are analyst forecasts, not measurements.** Every number above is quoted by an academic paper *from* a commercial market-research report. None of

the papers measured a market. Neither did this chapter. Cite them as evidence of expectation, not of size.

2. **The disagreement is the information.** Estimates for the same year differ by more than a factor of two. That spread tells you the category boundary is contested – which it is, and which is why Chapter 2 exists. When "digital twin" can mean a dashboard or a coupled multi-physics simulation, any total is a total over an ill-defined set.
3. **Adoption lags the forecasts, and the adoption numbers also disagree.** The same 2022 survey that quotes 58% annual growth also relays an engineering-institution report finding industry-agnostic adoption at around 5% of enterprises [26], while a 2025 paper relays industry surveys in which only about 5% of companies place twins *outside* their digital-transformation strategy altogether [28]. Both can be true – "we have a strategy" and "we have a twin in production" are different claims – and holding both in mind is the correct posture.
4. **Sector reports are more useful than totals.** Oil and gas [29], health [30], smart cities [31] and net-zero manufacturing [32] each have their own literature with concrete application scenarios. A sector report tells you what people in *your* customer's industry are actually doing. A global total tells you what a research firm sold a report about.

Two further pieces of context worth carrying into the meeting. The COVID-19 period is repeatedly credited with accelerating digital transformation and shifting executive focus from cost-saving toward digital investment [26] – so some of the growth in these curves is a step change in attention, not a smooth technology trend. And vendor material is part of the evidence base: overviews of the field cite vendor product pages alongside analyst reports [33]. That is not disqualifying, but it means the loudest numbers come from parties with an interest in them.

What to say in the meeting. Not "the market will be USD 137 billion". Say: "forecasts are large and inconsistent, adoption is early, and our justification has to come from our own baseline, not from a growth curve." That sentence is both more defensible and more useful.

1.6 Four deployment families, read for their value argument

The point of this section is not the technology. It is to show you four *different value arguments*, each with a different reason the money is there. Read each one asking: what is the decision, and what does being wrong cost?

Aerospace and defence – the cost of being wrong is enormous

A position paper from the American Institute of Aeronautics and Astronautics (AIAA) and the Aerospace Industries Association (AIA) sets out the aerospace industry's rationale for accelerating DT adoption, and is candid that progress so far has been uneven, with efforts often siloed and uncoordinated within each organisation's view of the system [34]. Related defence-side work frames the same territory as *digital engineering*: an integrated digital approach using authoritative models across the lifecycle, with a lineage explicitly traced back to Apollo-era simulation for risk reduction [35].

- **Decision:** whether this airframe or engine can safely fly another N cycles; whether a design change is acceptable.
- **Why it pays:** the cost of being wrong includes lives and airframes. Structural-health work in this family uses twin approaches for life prediction under unpredictable operational conditions [36], and monitoring engine wear and pressure tolerance is a standard cited application [37].
- **What it demands of you:** credibility and traceability above all (Chapter 7). Not speed.

Rotating machinery and offshore assets – the cost of *going there* is enormous

Offshore wind and offshore platforms have a distinctive economics: the asset is remote, visiting it is expensive and weather-dependent, and a failure found early is worth far more than the same failure found late. That is why this sector is a heavy adopter of condition-based and predictive maintenance, with twins as the enabler [9]. Fatigue monitoring of North Sea platforms has been done on an industrial basis with a twin approach [36], and wind-turbine prognostics is among the standard cited applications [37].

- **Decision:** send a vessel and a crew, or don't.
- **Why it pays:** each avoided unnecessary trip, and each avoided catastrophic-but-preventable failure, is directly costed.
- **What it demands of you:** intermittent, lossy connectivity; edge processing; long-horizon data (Chapters 9 and 10).

Process and discrete manufacturing – the cost of being *slightly* wrong, multiplied

Here no single error is catastrophic, but tiny per-unit losses multiply across enormous volumes. The literature is correspondingly focused on enablers and barriers to getting twins into the plant at all [22], and on where value shows up: predictive cost evaluation per component and plant-wide cost control in power generation [15], reference architectures for maintenance [38], and platform selection for specific goals such as net-zero targets [32].

It is not only for giants. A case study of a small-to-medium roll-to-roll label-printing manufacturer reports a deliberate strategy of protecting return on investment while adopting modern technology with minimal risk and investment [39] – which is the lean-increment argument of Sec. 1.2 showing up in the field.

- **Decision:** run, adjust, or stop; accept or reject a batch.
- **Why it pays:** volume. A 0.3% scrap reduction is a real number when the denominator is millions.
- **What it demands of you:** integration with systems that predate you, and uptime.

Health, cities and infrastructure – the cost is societal, and the twin is contested

DTs for health have proliferated fast enough to need scoping reviews [30], and the field is organised enough to have published position papers on the barriers slowing adoption and on what a shared ecosystem would need [40]. Smart-city work maps twin value onto concrete municipal scenarios – energy management, sustainable mobility, water resources – and onto evidence-based decision making for a community [31].

- **Decision:** a clinical or a policy one.
- **Why it pays:** outcomes, and cost avoidance at population scale.
- **What it demands of you:** governance, privacy, and an unusually high standard for explaining what the model does. This is the family where "the model said so" is least acceptable as an answer.

The common thread

In all four, the twin is paid for because **a decision was being made too late, too blind, or too expensively** – and the twin moves it. None of the four is paid for because a virtual replica is interesting.

1.7 When a twin is the wrong answer

A chapter that only argues for twins would be an advertisement. Here is the checklist for the other direction. Any single one of these is a reason to stop, or to shrink the proposal drastically.

1. **You cannot name the decision.** Sec. 1.2. This is the disqualifier.
2. **The decision is already good enough.** If the plant operator gets it right 99% of the time and the 1% costs little, there is no room.
3. **Nobody can act on the output.** A prediction that arrives faster than the organisation can respond changes nothing. Check the response process before building the predictor.
4. **The measurement does not exist and cannot be added.** No sensor, no twin. The instrumentation cost belongs in the estimate, not in a later phase (Chapter 9).
5. **The PT changes faster than you can re-validate the model.** If the plant is reconfigured monthly, model maintenance may exceed the benefit [21].
6. **The value metric is water in a lab.** That is the trap the worked example in Sec. 1.8 walks into deliberately – an obvious-looking metric whose total possible value is smaller than one week of your salary.
7. **The only justification offered is a market forecast.** Sec. 1.5.
8. **A simpler thing would do.** A threshold alarm, a checklist, or a better-placed sensor sometimes captures most of the value at a fraction of the cost. Proposing that instead is a professionally strong move, and it makes you more credible the next time you say a twin *is* warranted.

1.8 Worked example: the value case for the plant demonstrator

This book's running example is a real, documented PT: the INTO-CPS plant controller. From Part III onward you will build a DT for it. In this chapter you build only its **value case** – and you build it the way you would build it for a paying customer, including discarding the first metric you think of.

The scenario is not exotic. Plants in pots, sensors, and pumps are a recognised exemplar for DT work precisely because the physics is approachable while the systems problems

are real [41], and closely related greenhouse setups appear in the literature with the same ingredients – soil-moisture sensors, pumps, a camera [42].

1.8.1 The physical twin, as it exists today

These facts come from the demonstrator's own documentation and source code, not from an idealisation. Figure 1.3 is the whole system on one page; the bullets below name its parts.

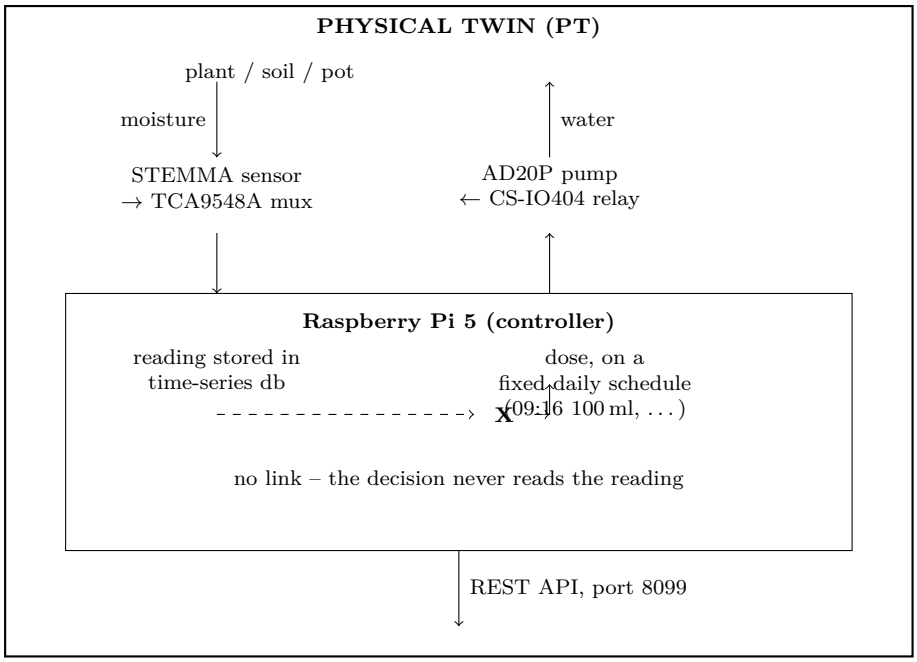


Figure 1.3 The demonstrator as it exists today. Sensing works and actuation works; the watering decision simply never reads the measurement. That missing link, marked X, is what this section's value case is about.

- A **Raspberry Pi 5** runs the controller software and connects to everything else.
- **Sensors.** An Adafruit STEMMA soil-moisture sensor per plant, reached over I2C through a TCA9548A multiplexer (so several sensors with the same address can coexist); an SHT45 for greenhouse air temperature and humidity; an AS7341 for ambient light spectrum; optionally a MODBUS soil probe reporting soil temperature, electrical conductivity and moisture.
- **Actuation.** An AD20P 12 V pump per plant, switched by a CS-IO404 MODBUS relay module.
- **Storage.** Measurements are written to a time-series database on the Pi.
- **Interface.** A REST API on port 8099. `GET /sensing/{unit}/{parameter}` returns measurements with optional `limit` and `since_timestamp`; `GET /actuation/{unit}/watering_events` returns watering history; `PUT /actuation/{unit}/updat` replaces the watering schedule. The documentation names the two actors explicitly: an *Implementer* with physical access, and a *Remote Connecter* – "either a person or a computer running a Digital Twin".

That last phrase matters. The PT was built with a socket for a DT to plug into, and that

socket is an HTTP API. Recognising the seam is the software engineer's contribution to the kickoff meeting.

- **Control, today.** Watering is **open loop**. The schedule is a fixed daily list of times and doses. The shipped example is:

  ```
  { "type": "daily",
    "schedule": [ { "time": "09:16:00", "dose": 100 },
                  { "time": "17:05:30", "dose": 120 } ] }
  ```

  A **dose** is millilitres. The pump driver converts it to a run-time with a calibrated linear model, `t = slope * dose + offset` seconds; the shipped example configuration uses `slope = 0.02`, `offset = 1.5`. So a 100 ml dose runs the pump for `0.02 x 100 + 1.5 = 3.5` seconds.

Read those two bullets together and the gap is visible. **The controller measures moisture and it stores moisture, but the watering decision never reads it.** The pump fires at 09:16 whether the soil is saturated, bone dry, or the tube has fallen out of the pot.

1.8.2 Step 1 – Name the decision

Two candidate decisions live in this system, and they are not the same project:

- **Decision A:** *How much water should pot 7 receive at 17:05 today?* Decider: the controller. Cadence: twice daily.
- **Decision B:** *Is the watering system for pot 7 actually working?* Decider: a human. Cadence: currently whenever someone happens to look.

Most people reach for A, because it is the flashy one – a closed-loop irrigation controller. Hold on to B; Sec. 1.8.4 will show why it wins.

1.8.3 Step 2 – Propose a value metric, then try to kill it

The first metric anyone proposes for irrigation is **water saved**. Test it with arithmetic before falling in love with it.

Take a university lab running four controllers with six plants each: 24 pots, each on the shipped schedule of 100 ml + 120 ml per day.

```
24 pots x 220 ml/day          = 5.28 L/day
5.28 L/day x 365              ~ 1 927 L/year  ~ 1.93 m3/year
```

At a metered water price of roughly EUR 5/m3, the lab's **entire annual water bill for this experiment is under EUR 10.** Even a twin that eliminated 100% of watering would save under EUR 10 per year.

> **This is the single most useful piece of arithmetic in the chapter.** The obvious metric was not merely small – it was three orders of magnitude away from the cost of building anything. Two lines of multiplication, done in the kickoff meeting, prevent a year of misdirected work. Do this to every metric before you accept it.

Note also that the arithmetic is *scale-dependent*, not wrong in principle. Change the denominator to a 9-hectare irrigated field – as in the OpenTwins agricultural case built around an irrigation pivot and soil-moisture data [43] – and water volume becomes a serious metric again. The lesson is not "water never matters". It is "run the numbers for *your* deployment".

1.8.4 Step 3 – Find the metric that carries the cost

If the water saving is not the goal, where is the cost of being wrong? Ask what actually goes wrong, and what it costs when it does. For a research lab, the answer is the *experiment*, not the water.

Suppose that over the past 12 months the lab's own records – this is the baseline, and it is measured, not assumed – show:

- **3 watering-system faults**: one blocked delivery tube, one empty reservoir over a long weekend, one relay coil that stopped switching.
- **Mean time from fault to discovery: 4 days**, because discovery happens when someone notices wilting.
- **2 of the 3 faults ruined a batch** of plants, each batch representing **6 experiment-weeks** of work.
- A **manual inspection routine**: about half a person-day per week spent walking the bench, checking pots, and eyeballing charts.

Now the value metric writes itself:

> **Value metric:** experiment-weeks lost per year to undetected watering faults. **Baseline: 12** (2 batches x 6 weeks), measured from lab records. **Secondary metric:** person-days per year of manual inspection. **Baseline: 26.**

Put costs on them. Use your own organisation's numbers; these are illustrative placeholders and are labelled as such:

```
Experiment-week   ~ EUR 1,600   (loaded researcher time + consumables + bench)
Person-day        ~ EUR   400

Annual cost of the problem
  12 experiment-weeks x EUR 1 600   =  EUR 19,200
  26 person-days      x EUR   400   =  EUR 10,400
                                       ----------
                                       EUR 29,600 / year
```

That is a number worth a project. It was invisible while the metric was "water saved".

1.8.5 Step 4 – Choose the value pattern

Which of the five patterns from Sec. 1.3 delivers that EUR 29 600?

- *Monitor* alone: partial. Charts exist today and the faults still took four days to find, because nobody was watching the charts.
- **Diagnose: yes.** The twin holds a model of how moisture *should* evolve – it decays as the plant transpires and the soil dries, and it steps up by a predictable amount when a 100 ml dose lands. Compare predicted with measured. A dose that produces

no moisture step means water is not arriving: blocked tube, empty reservoir, or dead relay. That residual is detectable within **one watering cycle**, roughly 8 hours instead of 4 days.

- *Predict*: attractive, and it serves Decision A (schedule the next dose from forecast dryness). But it is a larger project, and the EUR 29 600 is already captured by diagnosis. Defer it.
- *Decide* and *certify-and-train*: not where this money is.

Figure 1.4 is that residual, drawn. It is worth studying, because it is the smallest complete picture of what a DT does that measurement alone cannot: the model supplies the expectation, the sensor supplies the fact, and the *difference between them* is the product.

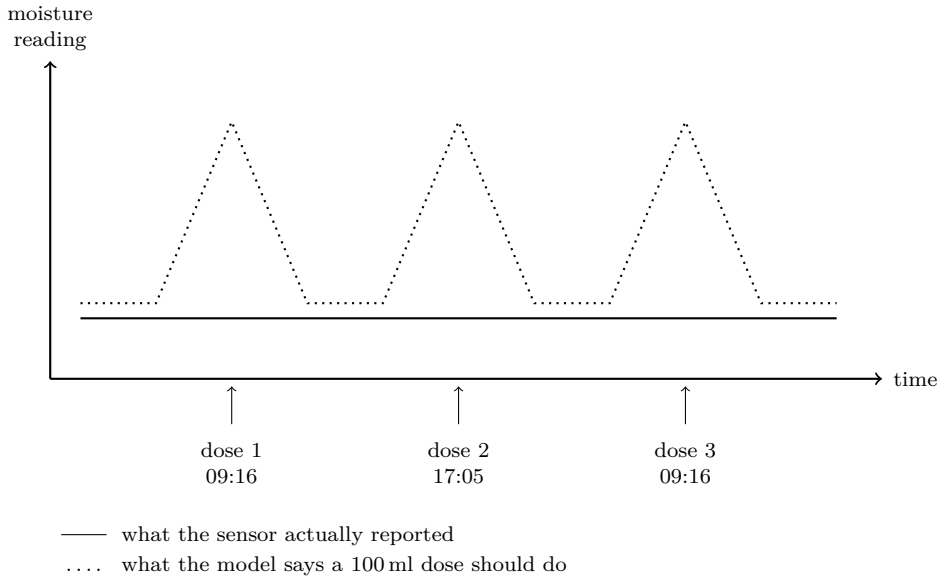


Figure 1.4 Why *diagnose* pays here. The fault is visible one watering cycle after it starts (∼8 h), not when a human notices wilting (∼4 days).

Note what Figure 1.4 does *not* require. The two curves do not have to coincide; they only have to differ by more than the sensor's noise when a dose fails to land. That is the whole fidelity requirement, and Sec. 1.8.6 derives it properly.

Chosen pattern: diagnose. Chosen decision: B, not A. The flashy closed-loop controller does not lose to the fault detector on elegance – it loses on the baseline arithmetic. That is what a value case is for.

1.8.6 Step 5 – Derive the fidelity requirement from the metric

Here is where being able to reason about models pays off. One term first, because it is about to do all the work:

> **Fidelity** – how closely a model reproduces the PT's behaviour *on the quantities the value metric depends on.* Note the second half. A model can be crude

about everything else and still be high-fidelity in the sense that matters.

The twin must decide: *did this dose reach the soil?* That requires a model that predicts the **sign and rough magnitude** of the moisture step following a 100 ml dose, and the approximate **decay rate** between waterings. A two-parameter water-balance model for one pot – in goes the dose, out goes evaporation and transpiration – is enough. It is a first-order model you could sketch on a napkin.

The twin does **not** need three-dimensional soil physics, root-zone modelling, or a plant growth model. Those would be needed for Decision A, where the question is "how many millilitres, exactly?" and the answer's accuracy is the product.

> **Generalise this.** *Fidelity is a requirement derived from the value metric, not a virtue pursued for itself.* Every hour spent on fidelity that the value metric does not need is unpaid work. It is the single most useful thing to say when a modelling expert and a business sponsor are talking past each other.

There is one more requirement, and it comes from the PT's own documentation rather than from the model: the twin must know **which sensor stream belongs to which plant**, and cope when plants are moved between shelves or a pot is replaced. The GreenhouseDT exemplar names exactly this – mapping a plant to the correct sensor stream, and reconfiguring when the arrangement changes [41]. In the demonstrator this surfaces as a per-plant JSON file binding a plant to a multiplexer port and a relay coil. Miss it, and your beautiful residual is computed against the wrong pot.

1.8.7 Step 6 – Estimate cost and payback

```
BUILD (one-off)
  Data path: poll the REST API, land readings in a time-series store   1.5 wk
  Water-balance model + parameter fitting from historical data         1.5 wk
  Residual computation, thresholds, alerting                           1.5 wk
  Deployment, documentation, handover                                  1.5 wk
                                                                     ---------
                                                                        6 wk
  6 engineer-weeks x EUR 2 000/wk                                =  EUR 12,000

RUN (annual)
  Small server / hosting                                         =  EUR    600
  Model + threshold maintenance, 0.5 day/month x EUR 400         =  EUR  2,400
                                                                     ---------
                                                                    EUR  3,000


BENEFIT (annual, deliberately conservative)
  Faults caught in ~8 h instead of ~4 days.
  Assume 2 of 3 faults are caught in time to save the batch,
  i.e. 1.5 of the 2 historically ruined batches survive.
    1.5 batches x 6 experiment-weeks x EUR 1 600                  =  EUR 14,400
  Manual inspection halved: 13 person-days x EUR 400              =  EUR  5,200
                                                                     ---------
                                                                    EUR 19,600
```

```
PAYBACK = (12 000 + 3 000) / 19 600 = 0.77 years ~ 9 months
```

Nine months. That is a project a sponsor will approve, and it is defensible line by line – which matters more than the number, because every line is a place a sceptical customer can push back, and you have an answer.

1.8.8 Step 7 – State what would falsify it

A value case you cannot falsify is a sales pitch. Write the falsifier down:

> **The case depends on the fault rate.** At 3 faults per year the payback is 9 months. At 1 fault per year the benefit falls to roughly EUR 10 000 and payback stretches to about 18 months, at which point the manual inspection saving – the softest number here – is carrying the case, and the honest recommendation is a cheaper alarm instead of a twin (Sec. 1.7, item 8).
>
> **Cheapest test:** before writing any twin code, extract 12 months of watering events and moisture readings through `GET /actuation/{unit}/watering_events` and `GET /sensing/{unit}/moisture`, and count how many doses produced no moisture step. That single query, against data the PT already stores, either confirms the fault rate or kills the project for the price of an afternoon.

That last move – spending an afternoon to avoid spending six weeks – is what separates an engineer who balances the cost of building a twin against the benefits derived from it.

1.8.9 The value case canvas

Collecting the worked example into a reusable form. Six lines. If you cannot fill all six, you do not yet have a project.

Line	Plant demonstrator
Decision	Is the watering system for this pot actually delivering water?
Decider & cadence	Lab technician, alerted per watering cycle (~8 h)
Value metric	Experiment-weeks lost per year to undetected watering faults
Baseline	12 per year, from 12 months of lab records
Value pattern	Diagnose
Falsifier	Fault rate below ~2/year; testable from stored history in one afternoon

---

1.9 Faded example: the commercial nursery

The situation. A commercial nursery grows ornamental plants in 4,000 pots across three greenhouses. The same controller design is installed, one controller per bench, all on fixed daily schedules. The nursery's own records show:

- Water is metered and charged: about **340 m3/year** for irrigation, at **EUR 3.20/m3**.
- **Plant losses to watering problems: about 2% of stock per year.** Average wholesale value EUR 4.50 per plant.
- Two staff spend a combined **4 hours per day** walking benches and spot-checking pots.
- Peak-season overwatering causes an estimated **1.5% of stock** to be downgraded from A-grade to B-grade, a EUR 1.10 per plant price difference.

Step 1 – the decision. Two again, and this time both are live: *how much water does bench 12 get* (an optimisation decision, made per bench per day) and *is bench 12's irrigation working* (a fault decision).

Step 2 – kill or keep the water metric. Run it:

```
340 m3/year x EUR 3.20/m3 ~ EUR 1,088/year
```

Under a thousand euros. Larger than the lab's EUR 10 by two orders of magnitude, and still too small to fund a twin on its own. Keep it as a secondary metric; do not build the case on it. (Notice this is the *same* calculation as Sec. 1.8.3, giving a *different* verdict about how seriously to take the number – which is why you run it rather than remembering last time's answer.)

Step 3 – size the other candidates. Two are yours to finish.

```
Plant loss:      4,000 x 2%   x EUR 4.50          =  EUR    360/year  ... check this
Grade downgrade: 4,000 x 1.5% x EUR 1.10          =  EUR     66/year  ... check this
Labour:          4 h/day x 365 / 8 h x EUR 400/day =  EUR 73,000/year
                 (= 182.5 person-days/year)
```

Now it is your turn. Two of those three lines are computed for a *single crop cycle*, and a nursery runs several cycles a year – the throughput figure, not the standing stock, is what the loss percentage applies to.

- **(a)** If the nursery turns its stock over **five times a year**, recompute the plant-loss and grade-downgrade lines. Which line now dominates?
- **(b)** Given your answer, which of the five value patterns does this nursery need – and is it the same one the lab needed?
- **(c)** Assume a build cost of 14 engineer-weeks at EUR 2,000 and running costs of EUR 6,000/year. Compute the payback period.
- **(d)** State the falsifier, in the form of Sec. 1.8.8: which single number, if the nursery has mis-remembered it, collapses the case?

Hint for (b): look at where the money landed in (a). If most of it is avoidable loss from watering the wrong amount rather than from watering not happening at all, you are no longer buying *diagnose*.

1.10 Posed problem: the municipal park

No scaffolding this time. A city parks department irrigates 120 zones across 14 parks. Zones are watered on fixed timers. The department's pressures are: a drought ordinance

that caps total seasonal water use, public complaints about visible sprinkler operation during rain, a maintenance crew of four covering all 14 parks, and a political requirement to demonstrate compliance.

Produce a complete value case canvas (Sec. 1.8.9) for a DT of the irrigation network. Then answer: **which stakeholder's metric should the project be justified on, and what changes about the engineering if you pick the compliance metric instead of the water metric?**

There is no single right answer. There are wrong ones, and they are the ones with no baseline. Smart-city twin work provides real municipal framings for scenarios of exactly this kind – energy, mobility, water – and evidence-based decision making as the stated goal [31]; architecture-pattern catalogues for agriculture and food systems offer a structured way to think about the systems side once the value case is settled [44].

1.11 Summary

Back to the objectives this chapter opened with.

1. **Decision first.** A twin is bought to improve a decision. Name the decision, the decider, the cadence, and the cost of being wrong; compress them into one value metric with a measured baseline (Sec. 1.2). Verify a defined financial benefit quickly and let it fund the next increment [7].
2. **Five value patterns.** *Monitor, diagnose, predict, decide, certify-and-train* (Sec. 1.3). Each rung typically needs the ones below and costs more. *Diagnose* is very often the right first increment: real engineering value, modest fidelity requirement.
3. **Cost structure.** Data integration is the biggest line, model maintenance is the biggest forgotten line, expertise is the most-reported barrier [22], and compute is smaller than people fear (Sec. 1.4). Bespoke one-off integration is the field's structural cost problem [23].
4. **Reading the evidence.** Market forecasts are large, inconsistent and analyst-sourced; adoption lags them; sector reports beat global totals (Sec. 1.5). Four deployment families pay for twins for four different reasons, and all four reduce to moving a decision earlier, cheaper or safer (Sec. 1.6). Eight conditions say *don't build it* (Sec. 1.7).
5. **A complete value case.** For the plant demonstrator: the obvious metric (water) was worth under EUR 10 a year; the real metric (lost experiment-weeks) was worth EUR 29 600; the right pattern was *diagnose*, not the flashier closed-loop controller; the fidelity requirement fell out of the metric; payback was about 9 months; and one afternoon of querying stored history can falsify the whole thing before you write any code (Sec. 1.8).

The one sentence to carry forward: **fidelity is a requirement derived from a value metric, not a virtue pursued for its own sake.**

1.12 Exercises

Objectives are noted in brackets. Solutions and hints follow.

1.12.1 (Objective 1, easy). For each, rewrite as a value metric with a unit, a denominator and a stated measurement method: (a) "improve building energy performance"; (b) "reduce risk in the substation"; (c) "make the production line smarter".

1.12.2 (Objective 2, easy). Classify each into one of the five patterns, and state the *next* pattern up the ladder and what it would additionally require: (a) a dashboard showing live bearing temperature across 40 pumps; (b) a system that flags a pump whose vibration deviates from its model; (c) a system that estimates each pump's remaining useful life; (d) a system that chooses which of 40 pumps to service during a six-hour maintenance window.

1.12.3 (Objective 1, medium). In Sec. 1.8.3 the water metric was killed by two lines of arithmetic. Construct a plausible deployment of the *same* demonstrator hardware in which water volume **is** the dominant value metric. State the scale at which the verdict flips, and show the arithmetic.

1.12.4 (Objective 3, medium). You are quoted EUR 40,000 to build a twin. The vendor's estimate lists sensors, cloud compute, and licences. Name three cost lines the estimate is likely missing, and for each say who in your organisation will end up paying it.

1.12.5 (Objective 5, medium). Complete the faded example of Sec. 1.9, parts (a) through (d).

1.12.6 (Objective 4, medium). A customer asks for a twin that predicts equipment failure two weeks ahead. Their maintenance schedule is fixed quarterly and cannot be changed within a quarter. Write the two-sentence reply you would give in the meeting, and state what you would propose instead.

1.12.7 (Objective 4, hard). Take the plant-demonstrator case of Sec. 1.8 and attack it as a hostile reviewer. Find three assumptions that, if wrong, break the payback. Rank them by how cheaply each could be checked before the project starts.

1.12.8 (Objectives 1 and 5, hard). Do Sec. 1.10, the municipal park, in full. Then write the single paragraph you would put at the top of the proposal for a reader who will read only that paragraph.

1.12.9 (Objective 3, hard, open-ended). Sec. 1.4 claims model *maintenance* is the most-forgotten cost. Take a software system you have personally maintained, and estimate the ratio of its total maintenance cost to its original build cost. Then argue whether a physics model's ratio should be higher or lower, and why. There is no correct answer; there is a correct *form* of answer, which names the drivers of change on both sides [21].

Solutions and hints

1.12.1. (a) "kWh per square metre per heating-degree-day, from the building meter, measured over one heating season." (b) "Unplanned outage minutes per year, from the Supervisory Control and Data Acquisition (SCADA) event log" – note that "risk" is not a metric until you say risk *of what*, measured *how*. (c) There is no answer. That is the point: "smarter" names no decision. The correct response is the questions from Sec. 1.2.

1.12.2. (a) monitor -> diagnose, which additionally requires a model of expected temperature and a residual. (b) diagnose -> predict, which additionally requires a forecast horizon and an uncertainty estimate [13]. (c) predict -> decide, which additionally requires running many scenarios fast [14]. (d) decide.

1.12.3. *Hint:* keep the price per cubic metre and increase the irrigated area until the annual water bill exceeds a plausible build cost. The literature offers a real anchor: an irrigation pivot covering 9 hectares, with soil moisture among the collected data [43]. *Sketch:* the lab's 24 pots consume ~1.93 m3/year. To reach EUR 12,000 of water at EUR 5/m3 you need ~2,400 m3/year – around 1,200 times the lab's volume. Somewhere between those two scales the verdict flips, and the flip point is where your build cost crosses your water bill. Stating that crossing point *is* the answer.

1.12.4. *Hint:* re-read Sec. 1.4. Strong answers name (i) integration with your existing systems and the data-quality work behind it; (ii) model maintenance as the physical asset changes [21]; (iii) the people cost – hiring or upskilling someone who can own this [22]. Weak answers name only compute. The second half of the question is the important half: these costs do not disappear when they are absent from the quote, they land on your team's roadmap.

1.12.5. *Partial solution.* (a) With five turnovers, the throughput is 20 000 plants: plant loss becomes 20,000 x 2% x EUR 4.50 = EUR 1,800, and grade downgrade becomes 20,000 x 1.5% x EUR 1.10 = EUR 330. Labour, at EUR 73,000, still dominates by an order of magnitude – which is a genuinely interesting result and a common one: the twin's biggest customer here is the wage bill, not the crop. (b) Since the dominant line is *staff walking benches to check things*, the nursery is buying a reduction in inspection effort: *monitor* plus *diagnose* covers most of it, and the closed-loop *decide* case has to be justified on the plant-loss and water lines alone (about EUR 3,200 combined) – which will not fund it. (c) and (d) are left to you; for (c) note that build plus first-year running is EUR 34 000, and for (d) look hard at the labour line, since it is both the largest and the one based on an estimate rather than a meter reading.

1.12.6. *Hint:* Sec. 1.7, item 3, and Sec. 1.3's warning under *predict*. A good reply names the mismatch without blaming the customer, and proposes the rung below: diagnosis, which is actionable within their existing quarterly cycle because it tells them what to put *on* the next quarter's list.

1.12.7. *Hint:* the three softest numbers in Sec. 1.8 are the fault rate (3/yr – checkable from stored history in an afternoon), the "1.5 of 2 batches saved" detection-effectiveness assumption (checkable only by running the detector against historical data, a few days), and the EUR 1,600 per experiment-week loading (checkable by asking the lab's finance office, an hour). Ranking by cost of checking is the actual exercise.

1.12.8. *Hint:* there is no solution, but there is a test. If your paragraph contains a number without a source, it is not finished.

1.12.9. Open. Assessed on whether both sides' change drivers are named: for software, requirements change and dependency churn; for a physics model, wear, repair, component replacement and reconfiguration of the PT [21].

1.13 Where to go next

In the literature, if you want more. These are the sources consulted for this chapter, grouped by what they are good for:

- *Book-length treatments of engineering twins for cyber-physical systems,* the closest fit to this book's angle: [45], with its chapter of worked case studies [16] and its chapter on advanced services and what-if analysis [14], [46].
- *Where the concept came from:* [4] for the 2002 origin, [5] for the naming history and the NASA definition, [6] for the Apollo lineage, and [47] for the question of whether twins are an evolution of modelling and simulation or something new.
- *Surveys with breadth:* [26] on enabling technologies and trends, [37] on tools, [25] on the business-model angle, [48] on the networking and IoT side, and [28] on moving from twins to twinning systems.
- *Value and adoption specifically:* [7] on the lean approach this chapter's method is built on, [22] on enablers and barriers, [2] on being realistic about return on investment, [3] and [20] on pilot experience, and [39] for a small-manufacturer case.
- *Sector deep-dives:* [29] for oil and gas, [30] and [40] for health, [31] for smart cities, [34] and [35] for aerospace and defence, [36] for offshore structural health, and [32] for net-zero manufacturing platforms.
- *Closest to our running example:* [41] presents a greenhouse twin as a community exemplar, [42] covers lifecycle management on a comparable set-up, [43] is a field-scale agricultural twin on an open platform, [44] catalogues architecture patterns using agriculture cases, and [17] is a small-scale reproducible exemplar in the same spirit as this book's demonstrator.
- *Consulted but not drawn on above,* and worth knowing they exist: [24] on ISO 23247 (returns in Chapter 13), [11], [12] and [38] on predictive maintenance (Chapter 11), [8] on the executable twin and model order reduction (Chapter 6), [9] and [10] on diagnostic twins, [49] on what a twin has to *know* to be trusted, [15] on industrial applications, [13] and [27] on modelling and uncertainty, [33] as an accessible overview, [18] on continuous integration for cyber-physical systems (Chapter 14), [23] on scaling (Chapter 15), and [19] on twins in engineering education.

In the demonstrator. Before Chapter 2, spend twenty minutes with `https://github.com/INTO-CPS-A` specifically `docs/pt/controller_3/index.md` for the hardware and `pt/controller_3/src/plant_cont` for the REST interface. You will build against that API from Part III onward, and the value case in Sec. 1.8 becomes much more concrete once you have seen the endpoints it depends on.

1.14 References

[1] T. Bergs, S. Gierlings, T. Auerbach, A. Klink, D. Schraknepper, and T. Augspurger, "The concept of digital twin and digital shadow in manufacturing," *Procedia CIRP*, vol. 101, pp. 81–84, 2021, Accessed: Jan. 21, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827121006612

[2] R. Bhandal, "Conceptualising the Application of Digital Twins in Supply Chain Management: A Path Towards Supply Chain Resilience," in *Digital Twin: Fundamentals and Applications*, S. Sabri, K. Alexandridis, and N. Lee, Eds., Cham: Springer Nature Switzerland, 2024, pp. 173–189. doi: 10.1007/978-3-031-67778-6_8.

[3] "Case studies of digitalization for creating digital twins." Change2Twin project. Available: https://www.change2twin.eu/about/deliverables/

[4] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems," in *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, F.-J. Kahlen, S. Flumerfelt, and A. Alves, Eds., Cham: Springer International Publishing, 2017, pp. 85–113. doi: 10.1007/978-3-319-38756-7_4.

[5] O. J. Pinon Fischer, S. Sabri, and Y. Chen, "Fundamentals of Digital Twins, Modeling Approaches, and Governance," in *Digital Twin: Fundamentals and Applications*, S. Sabri, K. Alexandridis, and N. Lee, Eds., Cham: Springer Nature Switzerland, 2024, pp. 13–29. doi: 10.1007/978-3-031-67778-6_2.

[6] F. Tao, M. Zhang, and A. Y. C. Nee, Eds., "Digital Twin Driven Smart Manufacturing," in *Digital Twin Driven Smart Manufacturing*, Academic Press, 2019, pp. i–iii. doi: 10.1016/B978-0-12-817630-6.00013-8.

[7] P. van Schalkwyk and D. Isaacs, "Achieving Scale Through Composable and Lean Digital Twins," in *The Digital Twin*, N. Crespi, A. T. Drobot, and R. Minerva, Eds., Cham: Springer International Publishing, 2023, pp. 153–180. doi: 10.1007/978-3-031-21343-4_6.

[8] D. Hartmann and H. Van der Auweraer, *The Executable Digital Twin: Merging the digital and the physics worlds*. 2022.

[9] F. Stadtmann and A. Rasheed, "Diagnostic Digital Twin for Anomaly Detection in Floating Offshore Wind Energy." arXiv, Jun. 2024. doi: 10.48550/arXiv.2406.02775.

[10] H. Zhang *et al.*, "Digital-Triplet: A new three entities digital-twin paradigm for equipment fault diagnosis," *Journal of Intelligent Manufacturing*, Aug. 2024, doi: 10.1007/s10845-024-02471-7.

[11] R. van Dinter, B. Tekinerdogan, and C. Catal, "Predictive maintenance using digital twins: A systematic literature review," *Information and Software Technology*, vol. 151, p. 107008, Nov. 2022, doi: 10.1016/j.infsof.2022.107008.

[12] D. Zhong, Z. Xia, Y. Zhu, and J. Duan, "Overview of predictive maintenance based on digital twin technology," *Heliyon*, vol. 9, no. 4, p. e14534, Apr. 2023, doi: 10.1016/j.heliyon.2023.e14534.

[13] A. Thelen *et al.*, "A comprehensive review of digital twin — part 1: Modeling and twinning enabling technologies," *Structural and Multidisciplinary Optimization*, vol. 65, no. 12, p. 354, Nov. 2022, doi: 10.1007/s00158-022-03425-4.

[14] M. Frasheri, T. Böttjer, P. G. Larsen, L. Esterle, and C. Gomes, "Advanced Digital Twin Services," in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds., Cham: Springer International Publishing, 2024, pp. 209–222. doi: 10.1007/978-3-031-66719-0_10.

[15] Y. Jiang, S. Yin, K. Li, H. Luo, and O. Kaynak, "Industrial applications of digital twins," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2207, p. 20200360, Aug. 2021, doi: 10.1098/rsta.2020.0360.

[16] B. J. Oakes *et al.*, "Case Studies in Digital Twins," in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds., Cham: Springer International Publishing, 2024, pp. 257–310. doi: 10.1007/978-3-031-66719-0_12.

[17] A. Barbie and W. Hasselbring, "Toward Reproducibility of Digital Twin Research: Exemplified with the PiCar-X." arXiv, Aug. 2024. doi: 10.48550/arXiv.2408.13866.

[18] F. Zampetti, D. Tamburri, S. Panichella, A. Panichella, G. Canfora, and M. Di Penta, "Continuous Integration and Delivery Practices for Cyber-Physical Systems: An Interview-Based Study," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 3, pp. 73:1–73:44, Apr. 2023, doi: 10.1145/3571854.

[19] M. Grieves, "Digital Twins and Their Role in Reengineering Engineering Education," in *Digital Twin: Fundamentals and Applications*, S. Sabri, K. Alexandridis, and N. Lee, Eds., Cham: Springer Nature Switzerland, 2024, pp. 237–261. doi: 10.1007/978-3-031-67778-6_11.

[20] "Change2Twin," *Change2Twin Project*. Accessed: Nov. 29, 2024. [Online]. Available: https://www.change2twin.eu/about/download/

[21] T. Alskaif *et al.*, "Evolution at the Core of Digital Twin Engineering," Grand Rapids, USA: IEEE, Jul. 2025. doi: 10.5283/epub.77656.

[22] M. Perno, L. Hvam, and A. Haug, "Implementation of digital twins in the process industry: A systematic literature review of enablers and barriers," *Computers in Industry*, vol. 134, p. 103558, Jan. 2022, doi: 10.1016/j.compind.2021.103558.

[23] S. A. Niederer, M. S. Sacks, M. Girolami, and K. Willcox, "Scaling digital twins from the artisanal to the industrial," *Nature Computational Science*, vol. 1, no. 5, pp. 313–320, May 2021, doi: 10.1038/s43588-021-00072-5.

[24] E. Ferko, A. Bucaioni, P. Pelliccione, and M. Behnam, "Standardisation in Digital Twin Architectures in Manufacturing," in *2023 IEEE 20th International Conference on Software Architecture (ICSA)*, Mar. 2023, pp. 70–81. doi: 10.1109/ICSA56044.2023.00015.

[25] K. Y. H. Lim, P. Zheng, and C.-H. Chen, "A state-of-the-art survey of Digital Twin: Techniques, engineering product lifecycle management and business innovation perspectives," *Journal of Intelligent Manufacturing*, vol. 31, no. 6, pp. 1313–1337, Aug. 2020, doi: 10.1007/s10845-019-01512-w.

[26] S. Mihai *et al.*, "Digital Twins: A Survey on Enabling Technologies, Challenges, Trends and Future Prospects," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2255–2291, 2022, doi: 10.1109/COMST.2022.3208773.

[27] A. Thelen *et al.*, "A comprehensive review of digital twin—part 2: Roles of uncertainty quantification and optimization, a battery digital twin, and perspectives," *Structural and Multidisciplinary Optimization*, vol. 66, no. 1, p. 1, Dec. 2022, doi: 10.1007/s00158-022-03410-x.

[28] G. Lugaresi and H. Vangheluwe, *From Digital Twins to Twinning Systems*. 2025.

[29] T. R. Wanasinghe *et al.*, "Digital Twin for the Oil and Gas Industry: Overview, Research Trends, Opportunities, and Challenges," *IEEE Access*, vol. 8, pp. 104175–104197, 2020, doi: 10.1109/ACCESS.2020.2998723.

[30] E. Katsoulakis *et al.*, "Digital twins for health: A scoping review," *npj Digital Medicine*, vol. 7, no. 1, pp. 1–11, Mar. 2024, doi: 10.1038/s41746-024-01073-0.

[31] D. McKee and D. Dokter, "DISCS: An Approach for Accelerating the Development of Digital Twins for Smart Cities," in *Digital Twin: Fundamentals and Applications*, S. Sabri, K. Alexandridis, and N. Lee, Eds., Cham: Springer Nature Switzerland, 2024, pp. 31–58. doi: 10.1007/978-3-031-67778-6_3.

[32] D. Parle, G. Sharma, N. Anand, N. Padgaonkar, D. Stoddart, and D. Malley, "A Comparative Analysis for Harnessing Digital Twin Platforms for Net-Zero Manufacturing," *IEEE Access*, vol. PP, Aug. 2024, doi: 10.1109/ACCESS.2024.3447475.

[33] R. Saracco, "Digital Twins: Bridging Physical Space and Cyberspace," *Computer*, vol. 52, no. 12, pp. 58–64, Dec. 2019, doi: 10.1109/MC.2019.2942803.

[34] "Digital Twin: Definition & Value – An AIAA and AIA Position Paper," *Aerospace Industries Association*. Accessed: Apr. 23, 2024. [Online]. Available: https://www.aia-aerospace.org/publications/digital-twin-definition-value-an-aiaa-and-aia-position-paper/

[35] "Digital Engineering Effectiveness." May 2022. Accessed: Mar. 13, 2024. [Online]. Available: https://insights.sei.cmu.edu/library/digital-engineering-effectiveness/

[36] H. Pezeshki, H. Adeli, D. Pavlou, and S. C. Siriwardane, "State of the art in structural health monitoring of offshore and marine structures," *Proceedings of the Institution of Civil Engineers - Maritime Engineering*, vol. 176, no. 2, pp. 89–108, Apr. 2023, doi: 10.1680/jmaen.2022.027.

[37] Q. Qi *et al.*, "Enabling technologies and tools for digital twin," *Journal of Manufacturing Systems*, vol. 58, pp. 3–21, Jan. 2021, doi: 10.1016/j.jmsy.2019.10.001.

[38] R. van Dinter, B. Tekinerdogan, and C. Catal, "Reference architecture for digital twin-based predictive maintenance systems," *Computers & Industrial Engineering*, vol. 177, p. 109099, Mar. 2023, doi: 10.1016/j.cie.2023.109099.

[39] P. Singh, N. Nidhi, J. Karpavice, M. Beliatis, and M. Presser, *Digital Dataspace and Business Ecosystem Growth for Industrial Roll-to-Roll Label Printing Manufacturing: A Case Study*. 2023.

[40] M. Viceconti, M. De Vos, S. Mellone, and L. Geris, "Position Paper From the Digital Twins in Healthcare to the Virtual Human Twin: A Moon-Shot Project for Digital Health Research," *IEEE Journal of Biomedical and Health Informatics*, vol. 28, no. 1, pp. 491–501, Jan. 2024, doi: 10.1109/JBHI.2023.3323688.

[41] E. Kamburjan *et al.*, "GreenhouseDT: An Exemplar for Digital Twins," in *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, in SEAMS '24. New York, NY, USA: Association for Computing Machinery, Jun. 2024, pp. 175–181. doi: 10.1145/3643915.3644108.

[42] E. Kamburjan, N. Bencomo, S. L. Tapia Tarifa, and E. B. Johnsen, "Declarative Lifecycle Management in Digital Twins," in *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, Linz Austria: ACM, Sep. 2024, pp. 353–363. doi: 10.1145/3652620.3688248.

[43] S. Infante *et al.*, "Integrating FMI and ML/AI models on the open-source digital twin framework OpenTwins," *Software Practice and Experience*, Mar. 2024, doi: 10.1002/spe.3322.

[44] B. Tekinerdogan and C. Verdouw, "Systems Architecture Design Pattern Catalog for Developing Digital Twins," *Sensors*, vol. 20, no. 18, p. 5103, Jan. 2020, doi: 10.3390/s20185103.

[45] P. G. Larsen, J. Fitzgerald, and C. Gomes, "Engineering Digital Twins for Cyber-Physical Systems," in *The Engineering of Digital Twins*, J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds., Cham: Springer International Publishing, 2024, pp. 3–17. doi: 10.1007/978-3-031-66719-0_1.

[46] J. Fitzgerald, C. Gomes, and P. G. Larsen, Eds., *The Engineering of Digital Twins*. Cham: Springer International Publishing, 2024. doi: 10.1007/978-3-031-66719-0.

[47] Z. Ali, R. Biglari, J. Denil, J. Mertens, M. Poursoltan, and M. K. Traoré, "From modeling and simulation to Digital Twin: Evolution or revolution?" *SIMULATION*, vol. 100, no. 7, pp. 751–769, Jul. 2024, doi: 10.1177/00375497241234680.

[48] A. Hakiri, A. Gokhale, S. B. Yahia, and N. Mellouli, "A comprehensive survey on digital twin for future networks and emerging Internet of Things industry," *Computer Networks*, vol. 244, p. 110350, May 2024, doi: 10.1016/j.comnet.2024.110350.

[49] N. Zhang, R. Bahsoon, N. Tziritas, and G. Theodoropoulos, "Knowledge Equivalence in Digital Twins of Intelligent Systems," *ACM Transactions on Modeling and Computer Simulation*, vol. 34, no. 1, pp. 1–37, Jan. 2024, doi: 10.1145/3635306.