

Brief for a crash course

Customer Development, the Validation Board, Scrum of Scrums, Large Scale Scrum, Kanban, and test-driven development. Written from the photographed exam, not from a full 60-item key.

PASTE THIS WITH THE PDF

Build me a crash course from the attached brief. Teach the ideas in the concept sections. Treat the answer key as ground truth.

Do not invent the missing exam questions, and do not present practice items as the original 60-item exam. The only known item numbers are 11, 14, 15, 16, 17, 18, 21, 22, 23, 24, 25, 27, 29, and 30.

For each known item, show why the right choice is right and why every wrong choice fails. Write for an IT student who needs to pass the exam and also understand the ideas.

Produce, in this order: a one-page map, five short lessons with one worked example each, an exam-phrase card, a drill of the 14 known items, and a separate 15-question practice set labeled PRACTICE. Plain language. No filler.

If the user only says "make the crash course" or "follow the brief," follow these rules and do not ask them to restate the material.

1. What the exam is testing

The photographed paper has 60 items. These photos captured 14 of them. Every captured item asks the same kind of question: can you tell learning from building, an opinion from evidence, and one scaling rule from another?

The crash course should teach the frameworks those 14 items sit in. It should not pretend the other 46 items are known. Practice questions are allowed only when they are labeled PRACTICE and can be defended from this brief.

- Score opinions, page views, and compliments as weak evidence.
- Score access, money, time, and a real commitment as customer currency.
- When an experiment fails, change the hypothesis. Keep the team.
- Do not start with the solution, the architecture, or the model.

2. The ideas, in the order to teach them

A. Search first, then execute

Steve Blank splits a new product into two modes. Search mode is hypothesis testing: who the customer is, what problem they have, and whether a solution is worth building. You expect to be wrong. Execution mode is scaling production software for a model that already has evidence.

Exam 23 is true. *IT startups and innovative teams should search before they transition into execution.*

B. Get out of the building

Get Out of the Building (GOOTB) is a Customer Development rule associated with Steve Blank. Leave the lab. Talk to real users. Watch what they do. Evidence collected in a meeting room, from the team's own guesses, does not count.

Exam 30 is true. *The paper calls GOOTB a core Customer Development directive associated with Steve Blank, and it contrasts real users with sitting in the lab. Mark it true. Do not argue with the wording "co-founded."*

C. Fire the plan, not the team

When an experiment invalidates a core technical or problem assumption, Lean and Customer Discovery discard the hypothesis. They do not discard the engineers. A pivot changes at least one core hypothesis: customer, problem, or solution. Replacing the student lead is not the method.

Exam 14 is true. Fire the hypothesis or the plan. Do not fire the engineering team.

D. The Validation Board

The Validation Board is the canvas from the Lean Startup Machine tradition. It tracks learning, not lines of code. Use one short sticky per box.

The three primary hypothesis stickies are Customer, Problem, and Solution.

- **Customer.** A specific group that shares a pain, described so you could recognize them. Not a broad demographic.
- **Problem.** That group's pain, in their words. Not a societal trend.
- **Solution.** What you believe will remove the pain. Write it late.

A core assumption is any belief that would break the business if it were false. List several. Test only the single riskiest one. One failed core assumption is enough to drop the concept.

The riskiest assumption is the core unproven belief with the least evidence, whose failure would make the entire software concept fail. It is not the hardest SQL query, the button color people argue about, or the largest cloud bill.

The minimum success criterion is the weakest result the team will still accept as validation. Set it before the experiment, with the stakeholders in the room.

After the experiment: if the result misses the criterion, pivot at least one core hypothesis. If it meets the criterion, pick the next riskiest assumption and test that. Track the pivots across the top of the board.

Exam 27. The three sticky types are Customer, Problem, Solution. Agile/Scrum/Kanban, test stages, and frontend/backend/database are not the stickies.

Exam 24. Riskiest assumption: least evidence, and invalidating it kills the whole concept.

Exam 16, the order the paper scores. Customer Segment, then Problem Hypothesis, then Riskiest Assumption, then Solution Hypothesis. Do not start with the solution, the architecture, a Docker pipeline, or unit tests. The board's own warning is: do not define a solution until the problem is validated.

E. Three methods, in rising cost

The board names exactly three experiment methods. Run the cheapest one that can answer the riskiest assumption. For this exam, Concierge and Wizard of Oz are the same move.

- **Exploration.** Interviews and observation. Collect stories of real behavior. Learn the problem. Do not sell yet.
- **Pitch.** Ask the customer to give something up in exchange for a solution: money, an email, time, access, or a signed commitment. That gift is currency. It tests whether the problem is a must-have.
- **Concierge.** Also called Wizard of Oz on this paper. Deliver the service by hand to a few customers, with as little technology as possible. Build the system only after they value the result and act on it.

Exam 17. Static code inspection is not one of the three. Exploration, Pitch, and Concierge are.

F. Opinions are not evidence

Exam 11 is the worked case. A student team interviews 10 DevOps engineers about a cloud cost optimizer. Eight say AWS bill reduction is extremely important. Only one agrees to grant read-only IAM access so a sandbox script can audit a real cluster.

The right reading: stated interest did not become customer currency. Pivot, or re-evaluate the customer

and problem hypothesis. Do not persevere because 80 percent sounded keen. Do not fire the student lead and hire a consultant. Do not answer a failed commitment test by demanding administrative IAM. A bigger ask is not new evidence.

Exam 25 is the vanity-metric case. One thousand unique page views on an open-source tool's documentation do not prove that enterprise teams will pay a monthly license. The statement is false. Payment, or another real commitment, is the evidence. Traffic is not.

G. Concierge, in the AI example

Exam 21. A team reviews pull requests by hand and emails the feedback to 10 beta teams every evening. They postpone the machine-learning parser on purpose.

The reason is to learn whether developers value the feedback and act on it before anyone invests in the model. It is not about AES-256 being stronger than an automated pipeline. It is not a Waterfall document gate. It is not a trick about patent protection.

H. Scrum of Scrums is not LeSS

Both names show up in multi-team Scrum. They are not the same framework, and neither one is Waterfall or Kanban.

- **Scrum of Scrums.** A coordination meeting. Representatives of the teams meet, often like a scaled daily standup, and surface dependencies and blockers. It is a technique. It does not require 1,000 developers.
- **LeSS.** A framework for several teams building one product: one Product Owner, one Product Backlog, one shared sprint, feature teams, one Definition of Done.

Exam 15 is false. *LeSS does not give every feature team its own isolated Product Backlog.*

Exam 22. *SoS uses team representatives at a scaled standup. LeSS shares one Product Backlog and one sprint cadence. They are not identical. SoS is not Waterfall, LeSS is not Kanban, and the head-count options (1,000 developers, or a cap of 3) are not the difference.*

I. Kanban does not run on sprints

Scrum schedules work in fixed-length, timeboxed iterations called sprints. Kanban schedules by flow. People pull work when capacity frees up. A limit on work in progress is the control, not a sprint calendar.

Exam 29 is false. *Kanban does not rely on sprints to schedule releases.*

J. Test-driven development writes the test first

In Extreme Programming, test-driven development follows red, green, refactor. Red: write a unit test that fails because the behavior does not exist yet. Green: write the smallest production change that makes the test pass. Refactor: clean the code while the test stays green.

Exam 18 is false. *TDD does not say write the production code first and add the unit tests afterwards.*

3. Answer key

Use these verdicts even if another source phrases the framework differently. This is the paper that was photographed.

11

Pivot or re-evaluate the problem and customer hypothesis. Eight engineers called bill reduction extremely important, and only one granted read-only IAM access. Verbal interest is not currency.

Do not persevere on the 80 percent. Do not replace the student lead. Do not escalate the IAM request to admin.

- 14 True. Fire the hypothesis or the plan, not the engineering team.**
A failed experiment is evidence about the assumption.
- 15 False. LeSS uses one shared Product Backlog, not an isolated backlog per feature team.**
One Product Owner, one backlog, one sprint.
- 16 Customer Segment, then Problem Hypothesis, then Riskiest Assumption, then Solution Hypothesis.**
Do not start from the riskiest assumption, a Docker pipeline, or the solution architecture.
- 17 Static Code Inspection is not a validation method.**
The three methods are Exploration, Pitch, and Concierge.
- 18 False. TDD writes the failing test before the production code.**
The cycle is red, green, refactor.
- 21 The manual review checks that developers value and act on the feedback before the team builds the machine-learning pipeline.**
Not encryption, not a Waterfall gate, not a patent story.
- 22 SoS is a meeting of team representatives. LeSS shares one Product Backlog and one sprint cadence.**
Not identical. Not Waterfall versus Kanban. Not a 1,000-person rule versus a 3-person cap.
- 23 True. Search (hypothesis testing) comes before execution (scaling production software).**
Skipping search is how a team scales something nobody wants.
- 24 The riskiest assumption is the unproven belief with the least evidence, whose failure kills the whole concept.**
Not the hardest query, the disputed button color, or the largest invoice.
- 25 False. A thousand documentation page views do not prove enterprises will pay a license fee.**
Page views are a vanity metric. Payment is evidence.
- 27 Customer, Problem, Solution.**
Not Agile/Scrum/Kanban, not the test pyramid, not frontend/backend/database.
- 29 False. Kanban does not schedule releases with fixed sprints.**
Sprints belong to Scrum. Kanban is a flow with a work-in-progress limit.
- 30 True. Get Out of the Building means collect evidence from real users instead of staying in the lab.**
Mark the statement true, including its attribution to Steve Blank.

4. What the crash course must not do

- Do not fabricate items 1–10, 12, 13, 19, 20, 26, 28, or 31–60, and do not number a practice item with one of those slots.
- Do not teach SAFE, Nexus, or the Spotify model as if this paper used them. They were not in the photos.
- Do not teach LeSS as "each team owns a private backlog." The tested rule is one Product Backlog.
- Do not split Concierge and Wizard of Oz into two methods. This paper uses them as one method.
- Do not treat praise, a survey score, or page views as customer currency. Currency is something the customer gives up.
- Do not start a lesson with the solution, the model, or the deployment pipeline.

5. Shape of the crash course

Five lessons, then two drills. Each lesson gets one worked example and three sentences the student can say in an exam.

- **Lesson 1.** Evidence before building. Search versus execution, GOOTB, and firing the plan. Example: a team that stays in the lab.
- **Lesson 2.** The Validation Board, in the exam's order, including how to pick the riskiest assumption and how to set a minimum success criterion.
- **Lesson 3.** Exploration, Pitch, and Concierge. Work the cost-optimizer case and the handwritten code-review case.
- **Lesson 4.** Scrum of Scrums versus LeSS, then Kanban versus sprints. One comparison table is enough.
- **Lesson 5.** Test-driven development: red, green, refactor, with one tiny code example.

After the lessons, print an exam-phrase card: one line per known item, in the words of the answer key. Then drill the 14 known items in a shuffled order, with the wrong choices explained. Finish with 15 new PRACTICE questions drawn only from sections 2 and 3, each marked PRACTICE, each with an answer and a one-line reason.