Experience debt

A method to turn scattered knowledge into strategic action

Handout notes for the talk by Chiara Scarabotti, UX Lead, Springer Nature

THE TALK IN 30 SECONDS

Known user problems pile up like an ever-growing task list: invisible, scattered, then forgotten.
Experience debt names the condition: the gap between what users need and what our experience enables.
Capture, Connect, Resolve is the method for managing it.
The implementation can vary, but starting small is the best approach: a spreadsheet with four columns is enough; the habit matters more than the tool.

1. Why this started

Have you ever felt that your products are accumulating user problems over time, and that their complexity and dependencies are also hard to represent? I've experienced this feeling too, so I started to focus on how to make this information visible and actionable for everyone.

In conversations with teams and leadership, I kept hearing the same questions. Some were detailed (where is the evidence for this?), others strategic (what pattern are we missing?). They are questions about the past, present and future of a problem, and answering them meant digging every time, with answers that kept changing. We did not lack knowledge. We lacked control of it.

Three things were getting in the way:

- **Invisible.** We have portfolios for solutions, prototypes and mock-ups, but few ways to show problem understanding: the patterns, connections and evidence behind the first design.
- **Scattered.** Research, support, analytics, stakeholders and people's memory each hold part of the picture because the knowledge is organised around where it comes from. The cost: we rediscover what we knew, we miss connections (treating symptoms, not causes) and we ship without tracking user impact.
- **Forgotten.** A problem can leave the roadmap without leaving the experience. Once it is not visible, we lose its context and relationships, and we may assume it has been solved.

2. The concept: borrow the persistence of technical debt

Engineers preserve knowledge about technical problems as **technical debt**. The word sounds negative, but it makes the cost of ignoring a known issue explicit, like a bill waiting to be paid. More importantly, a known issue becomes an object worth maintaining even when attention moves elsewhere. The question I asked: could known user problems get the same persistence?

Experience debt can represent **the accumulated gap between what users need and what the current experience enables**. If a user is still experiencing the gap, it is part of the debt, regardless of how it originated: design compromises, usability and accessibility issues, unmet needs, and so on. Keeping this wide view matters, because removing things from the picture can remove connections we do not yet understand.

Debt defines the condition. It does not tell us how to manage it. That is the method.

3. The method: Capture, Connect, Resolve

	Capture	Connect	Resolve
Fights	Problems being invisible	Knowledge being scattered	Problems being forgotten
Gives a problem	A home and a persistent identity	A context and a strategic meaning	A history and a future
In practice	One framed problem, one unique ID, linked to the user goal (or Job To Be Done) it blocks and to its sources	Explicit relationships between problems: causes, blocks, shares a root cause	Every hypothesis and solution attached to the same problem; debt is paid only when the experience changed
You get	Self-explanatory problems you can interrogate, cite and use	Cross-journey, JTBD and business views	Product, design and leadership views (enriched with connections and previously captured metadata)

4. A worked example: UX-301

Imagine a hotel booking system. Problem UX-301 is: "Guests cannot tell if their hotel reservation was successful."

- **Capture.** UX-301 is one unique item. New evidence is added to it, not to a copy. It is linked to the job it blocks (Complete booking with confidence) and its sources (customer support, stakeholders, research). Anyone, not only UX, can understand it, and anyone can interrogate it (for example by product), cite it in a persona or document, and carry it into a workshop.
- **Connect.** UX-301 causes UX-444 (hotel staff struggle with the volume of email requests) and blocks UX-122 (guests struggle to complete online check-in). Making these relationships explicit gives us a holistic view of the system and helps us decide where to intervene. Rather than fixing the email system first, we ask what creates that volume, which connection is the strongest and what ripple effects we should avoid. This is systems thinking applied to our list: looking beyond the isolated issues to identify patterns and strategic themes.
- **Resolve.** Solution 1 changes nothing, Solution 2 helps some users, Solution 3 resolves it. Every attempt stays attached to UX-301, so another team can learn from the past. If you cannot measure the user impact, record the attempt and keep the problem in the debt.

> **THE RULE FOR PAYING THE DEBT**
> **A problem is crossed off from the debt only when we have evidence that the experience changed for the better, not only if we shipped something.**

5. One debt, different ways to work with it

Experience debt aims to answer all the questions that teams and leadership keep asking about user problems. We have experimented with different interfaces, so that more disciplines can interrogate it. These are examples of three different ways to work with the same experience debt, but the method is tool-agnostic.

Jira	**Structurally**	Navigate by fields, metadata, queries, and boards. Ask: what problems relate to a user type, JTBD or source? Which are active, dormant or resolved?
Miro	**Visually**	Navigate through journeys and connections. Ask: where does the problem live on the journey? What are the direct connections between experiences?
AI agent	**Conversationally**	Navigate by asking. What do we know about this issue? What are the biggest systemic problems? Is this evidence new, or does it relate to an existing issue?

6. Start small: four columns, then connections

You do not need Jira, AI or a complicated taxonomy. A spreadsheet with four columns is already a basic form of Capture and Resolve; add a fifth for relationships and you are beginning to Connect. Customise it as your knowledge grows.

Problem	Source	JTBD	Status	Connects with
User cannot confirm the action succeeded	User interviews; Stakeholders	Complete with confidence	Ideating	*Relates to #4*
Important information is easy to miss	CSAT comments; Marketing research	Make an informed decision	Validating	
The same details must be entered twice	Analytics; Support	Finish in one go	Backlog	*Blocks #11*

Problem, Source and JTBD = Capture · Status = Resolve · Connects with = Connect

Then build a habit, not just a database:

- **Add more information when it arises.** Strengthen an existing problem instead of creating a disconnected new one.
- **Bring your debt into conversations.** Ask colleagues where they see connections, or what can be prioritised. Share it with leadership: they appreciate the strategic view and can sponsor the scaling up.
- **Keep each problem alive until you know it has changed.** It is a debt we all must pay. UX can coordinate it and make sure real user problems are recorded, but maintenance and usage need to be shared so that the approach stays sustainable and valuable.

7. One last reflection, and what to take home

Shaping this, I incorporated ideas that already existed: atomic research, JTBD, systems thinking for relationships, and an outcome-driven, Lean UX approach for resolution. None is new on its own, but in isolation none of them served the main purpose.

What emerged was a way of organising them around the lifecycle of a user problem. A new method does not need brand new ingredients; it can emerge from combining existing ones.

> **NEXT TIME YOU FACE AN OVERWHELMING LIST OF KNOWN PROBLEMS**
> **Begin with one problem. Make it visible, connect it, and keep it alive.** If a problem is not captured, it stays invisible, so it cannot be connected and becomes harder to prioritise.

8. Reading list

The ideas and methods that inspired this work:

Capture

Jobs-to-be-Done: A Framework for Customer Needs by Tony Ulwick

What is Atomic UX Research? by Daniel Pidcock
Cards for discovery by Emily Heath

Connect
Closing the Loop: Systems Thinking for Designers by Sheryl Cababa
Thinking in Systems by Donella Meadows
Rethinking UX Debt as a System, Not a List by Chiara Scarabotti

Resolve
Why UX Outcomes Make Better Goals Than Business Outcomes by Jared Spool
UX Debt: How to Identify, Prioritize, and Resolve by Anna Kaley
The Double Diamond (Design Council)
Lean UX by Jeff Gothelf and Josh Seiden