SAST 30

My Essence of Testing: Understanding Relations

Rikard Edgren
rikard.edgren@inera.se

if testing is easy you are doing something wrong

Why this presentation

Testing happens in your head.

Your understanding determines the quality of your testing.

The most interesting understanding is in relations.

~~The Essence of Testing~~

My Essence of Testing

My Old Essence of Testing

First you understand what is important, then you test it

My New Essence of Testing

By understanding relations, you will among many things understand what is important, so you can do good testing (while also providing value in other ways)

The value of testing happens in the communication of qualities, problems, risks, and more.

About

Rikard Edgren

Lives in Karlstad, Värmland, Sweden.

Three children. Long distance runner.

Works with testing digital infrastructure in Swedish healthcare.

Author of The Little Black Book on Test Design and
Den Lilla Svarta Om Teststrategi.

Member of thetesteye.com think-tank.

rikard.edgren@nordicmedtest.se

Understanding is important

Testing is a sampling business, we are never finished.

How your sampling is designed, executed, evaluated and reported can be decided by

- other people
- tradition
- something you read in a book or on a web site
- your experience from other situations
- ***your understanding of this unique situation***

Πάντα ῥεῖ

Heraclitus

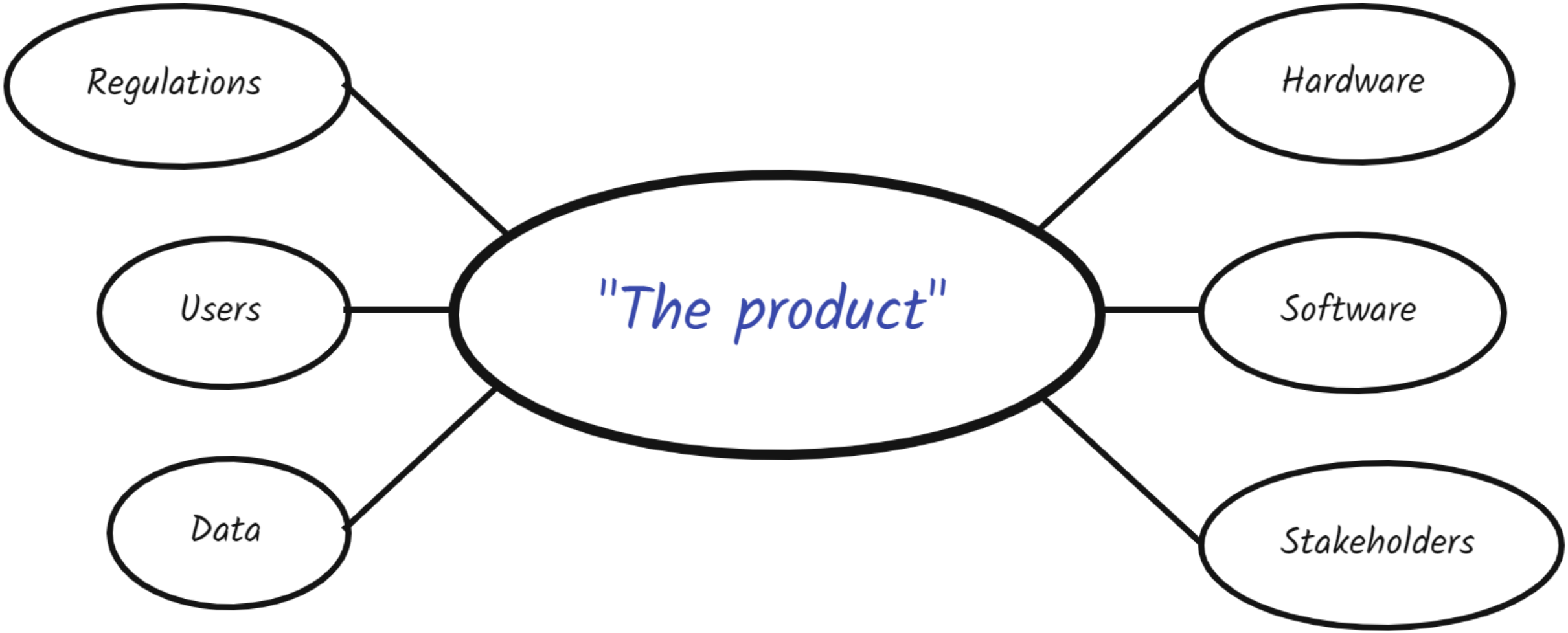
Relations are important

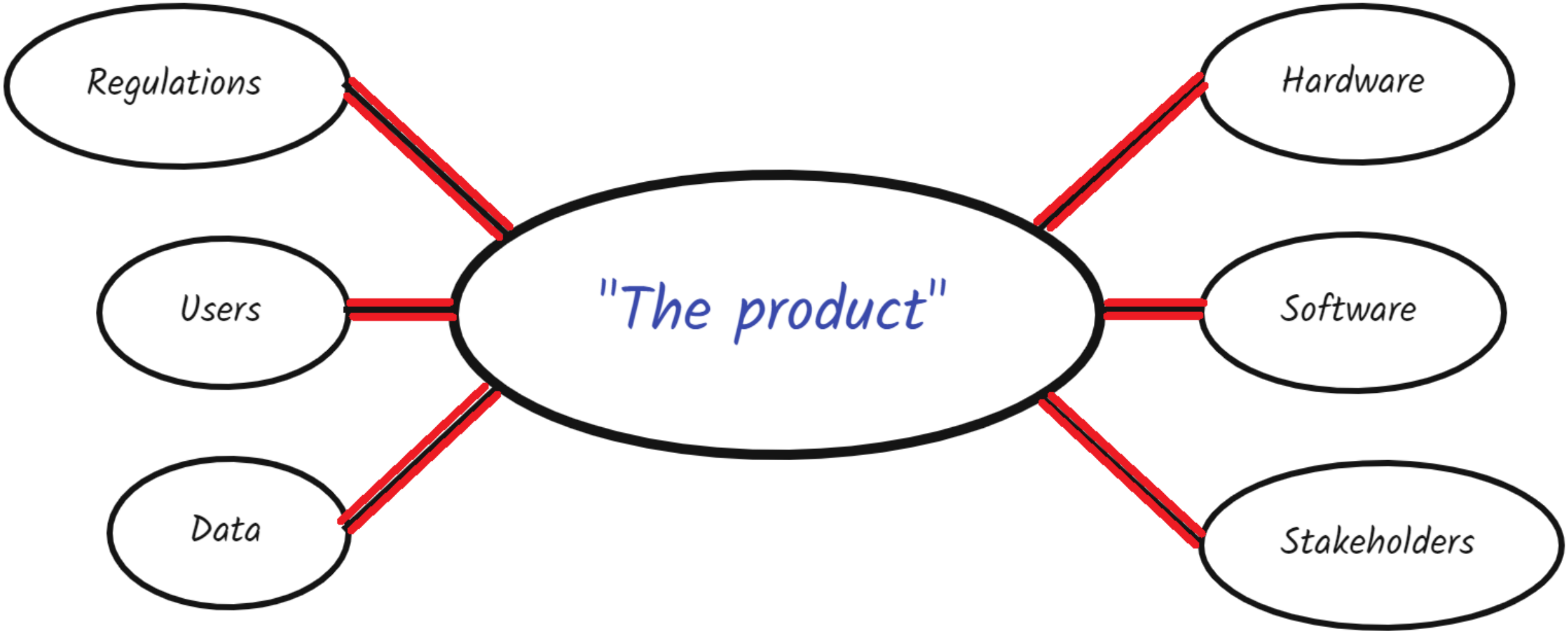
You can try to understand a Print button in isolation, how it is designed and what it does

But to test it I believe you need to understand the relationships with

- printers, operating system
- the data, the surrounding software
- users' needs, knowledge, privacy…
- ideas from developers, product owners et.al.

Relations define your true understanding in a complex world


Regulations
Users
Data
"The product"
Hardware
Software
Stakeholders


Regulations
Users
Data
"The product"
Hardware
Software
Stakeholders

Continuous learning

You must know things in order to be able to test a product well.

And testing the product is learning more about it.

And by doing that you will have more ideas about what to test…

And you will get new perspectives that bring value to the whole team, and better chances to find important information.

Being a fast-learner might be the tester's most important trait.

Learn how to be a fast learner might be my best advice

Experimenting?

Discussing?

Talking?

Trying?

Reading?

Thinking?

Writing?

Teaching?

sast

Testing is not an island

The importance of relations of course also applies to the testing, and to you

The **value** of testing happens outside of testing, and outside of you

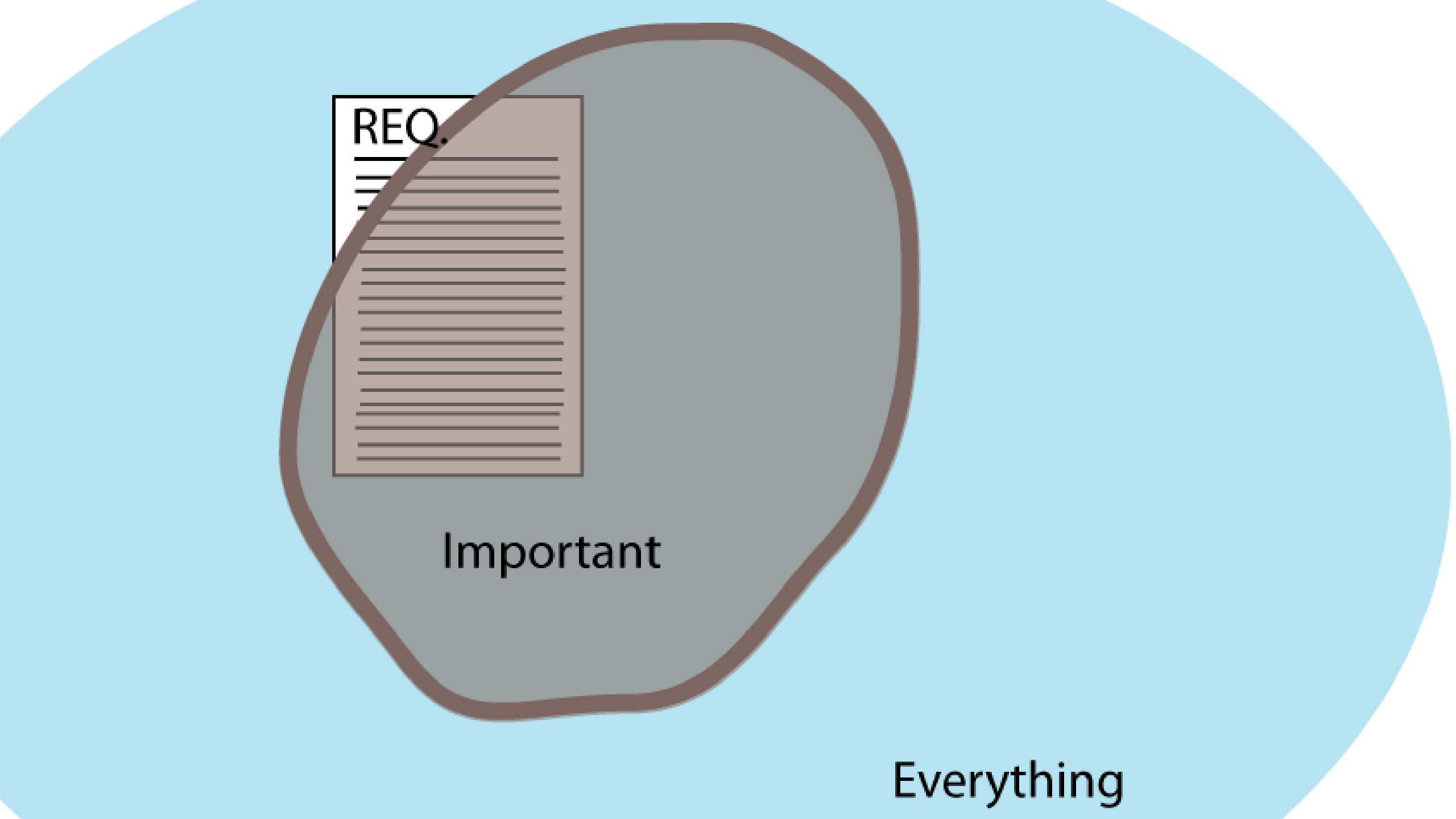
My best learnings and moments involve **people** I trust and like

Managing personal relations

Testing (and life) would be nothing without personal relations

But it is not enough to understand your **mood** and your friend's mood, you subconsciously also understand the relation of the moods. Together you can turn a **bad day** to a **good day**.

Respect takes time to earn, but can be destroyed fast


REQ.
Important
Everything

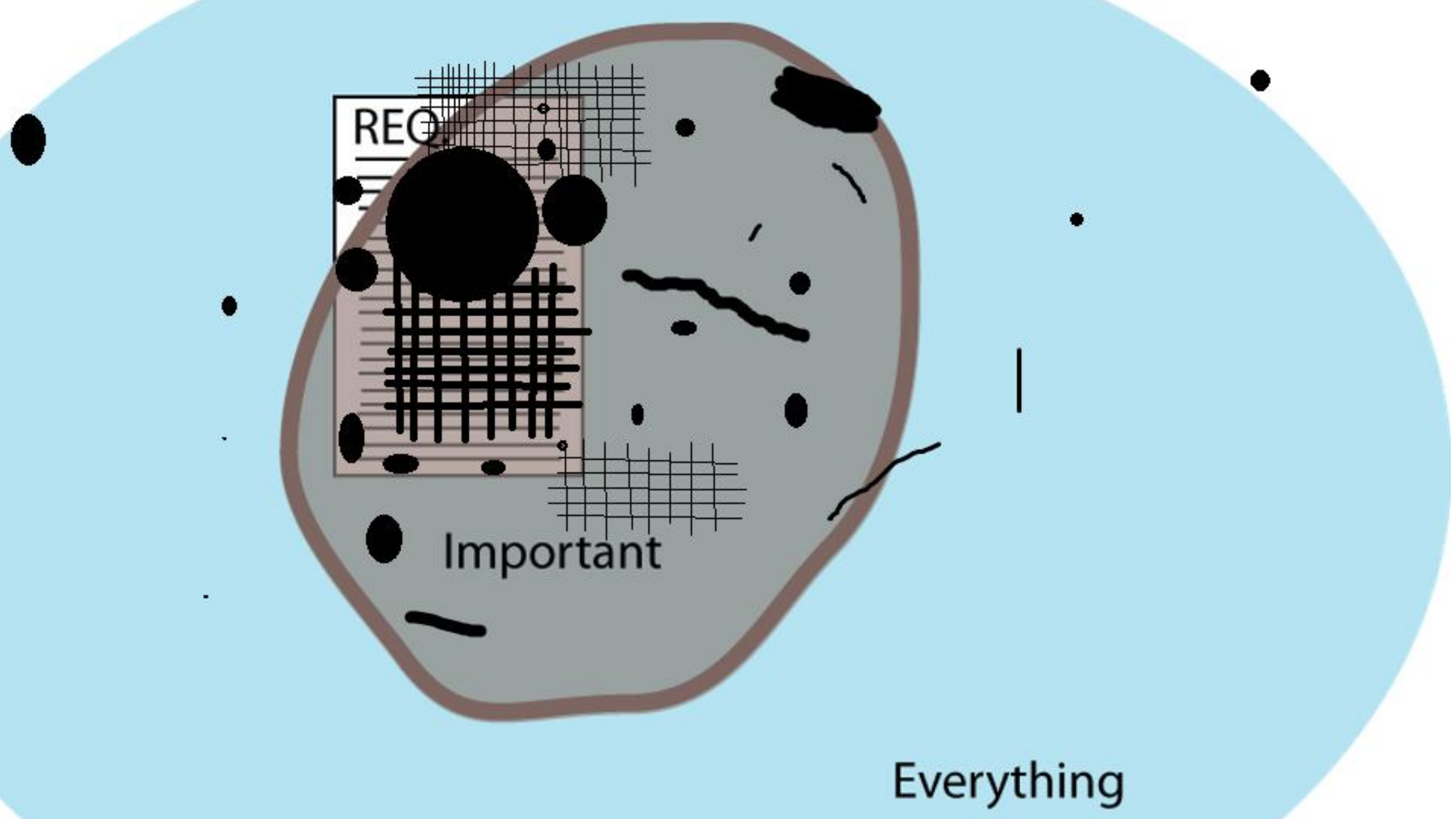
The Number One Pragmatic Testing Advice

This is from the great book Lessons Learned in Software Testing by Kaner, Bach, Pettichord, Lesson 283

Diverse Half-Measures

“it's better to do more different kinds of testing to a pretty good level, than to do one or two kinds of testing perfectly.”

Testing in different ways unveils more relations that matter


REQ
Important
Everything

Good testers are often lucky

In a sampling business, I recommend to always be ready to find valuable things you didn't know you were looking for - ***serendipity***

preparations – increase chances of luck

wide awareness – and observant mind comes with experience…

sagacity - by knowing a lot about the software and its usage, you can make wise judgments about what happens, and what more should be tested.

by understanding **similarities** and **anomalies**, you will find valuable things, often

Models

People have different ways of modeling, and that's a strength.

We need several models in order to capture what's important to different users.

Models can come from

- developers
- data
- surroundings
- usage
- quality characteristics
- your detailed knowledge about the product

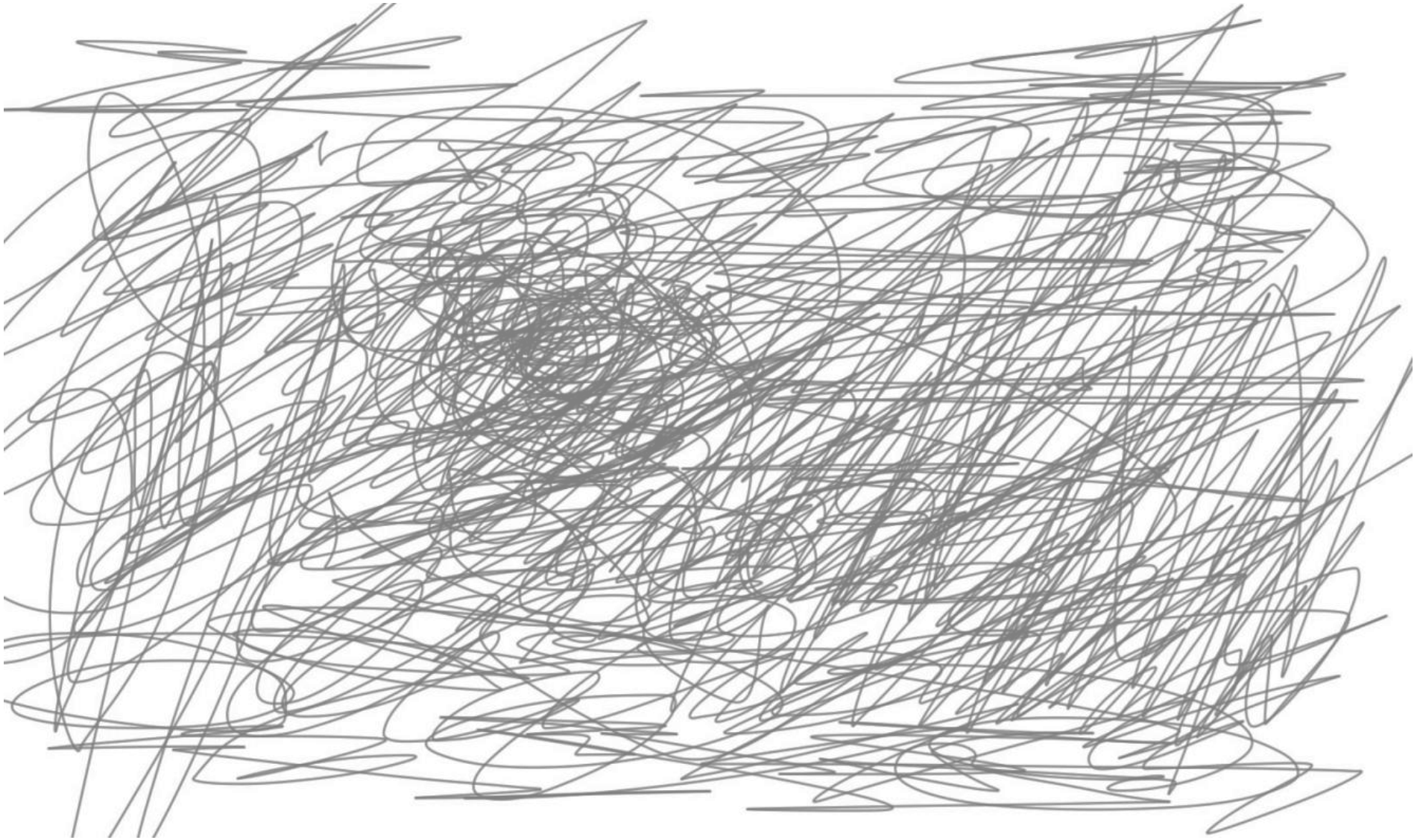
37 Sources for Test Ideas

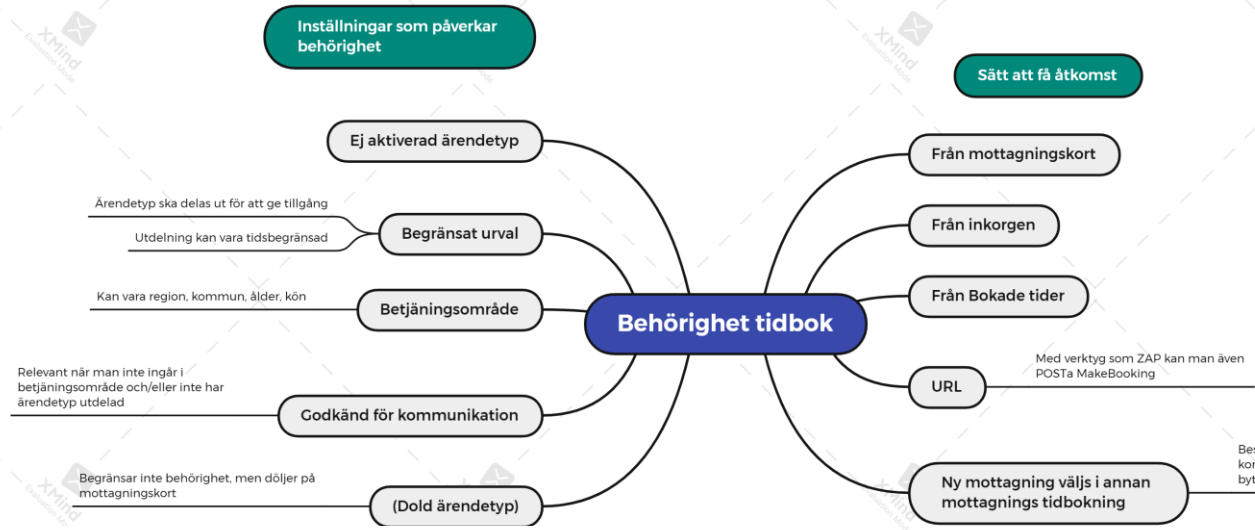
Invisible, overlapping models

Skilled testers make many, invisible models all the time.
Your understanding (regardless of origin) is a sort of model.

The diversity in models bring more, and richer test ideas.
These don’t need to be documented, but it’s a good exercise.

- Image: Ruud Cox

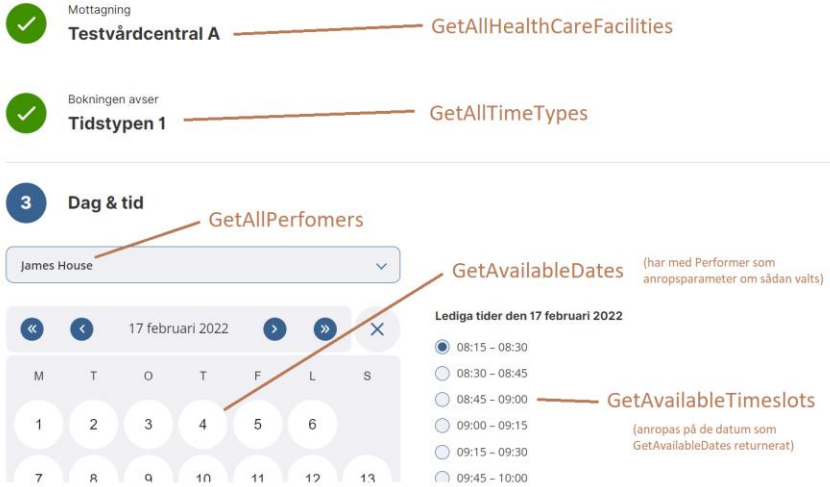




Tjänstekontraktskarta tidbokning

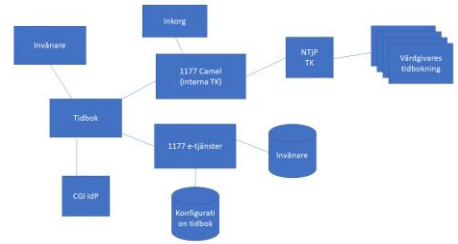
Ombudsläge	RUÖT	Formulärtjänsten	Inaktivitetsutloggning	Regionen rekommenderar
BankID QR	Ny ärende-editor	Skyddad folkbokföring	Uthoppstjänst	Bokade Tider
Långa scenarion	Omvänt ärendeflöde	In/utloggning	Listning	Stöd och Behandling
Webbtidbok	Väntrum	Egen Provhantering	Direktlänkar	Användarvänlighet
Byt användare	Mobiltelefon	Nya ärendetyper	Sök mottagning	Genvägar

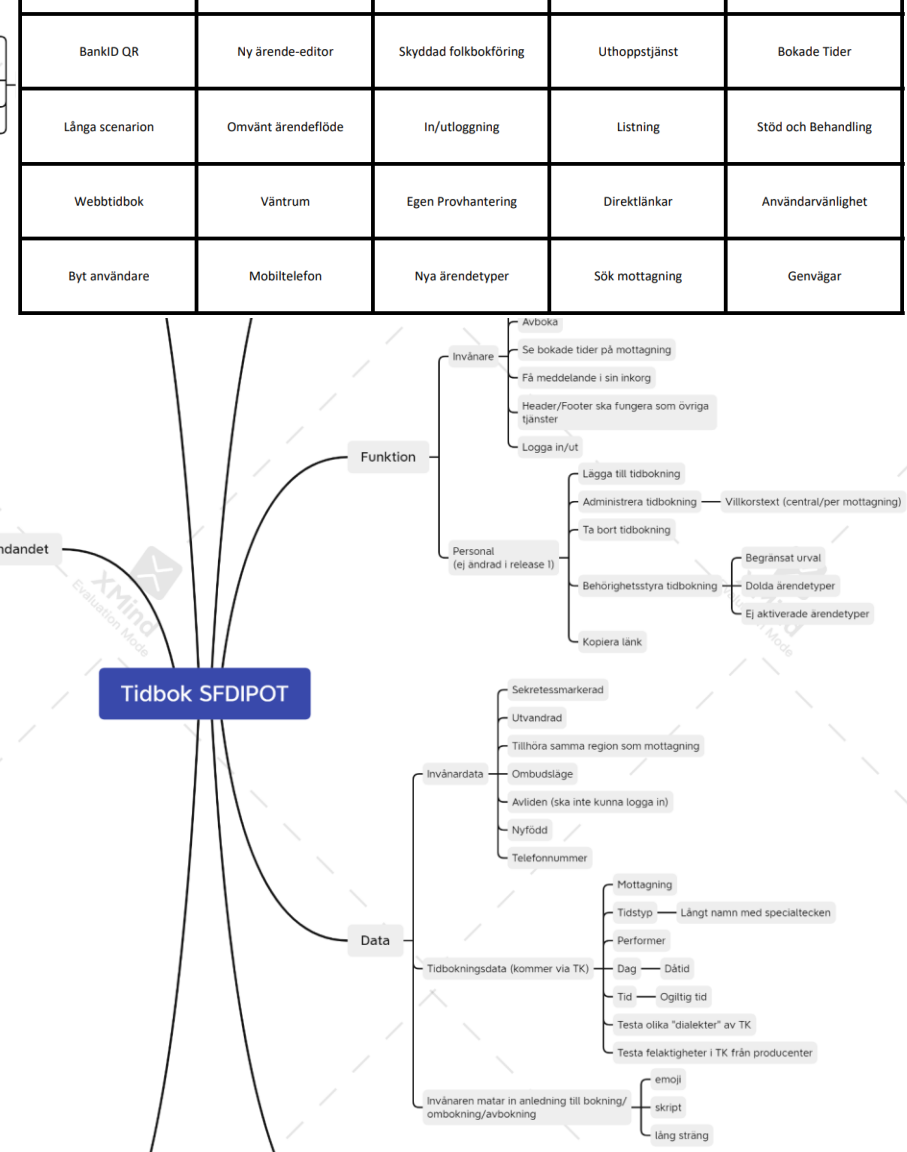
Ombudsläge
Rubriker Under Övriga Tjänster
Parent-child



Boka — MakeBooking (använder de detaljer man fått i GetAvailableTimeslots-svaret
Omboka — UpdateBooking
Avboka — CancelBooking

GetBookingDetails — används av Bokade Tider och sidan Av/Omboka tid





Modeling – Product Elements

A great framework for getting structure is to use SFDIPOT from James Bach's Heuristic Test Strategy Model.

Structure – what the product is

Functions – what the product does

Data – what the product operates on

Interfaces – how the product can be accessed or expressed

Platform – the environment the product depends on

Operations – what the users want to accomplish

Time – relations between the product and time

Software Quality Characteristics

Go through the list and think about your product/features. Add specifics for your context, and transform the list to your own.

Capability. *Can the product perform valuable functions?*

- *Completeness:* all important functions wanted by end users are available.
- *Accuracy:* any output or calculation in the product is correct and presented with significant digits.
- *Eff*
- *Int*
- *Co*
- *Da*
- *Extensibility:* ability for customers or 3rd parties to add features or change behavior.

Reliability. *Can you trust the product in many and difficul*

- *Stability:* the product shouldn't cause crashes, unhandled exceptions or scr
- *Robustness:* the product handles foreseen and unforeseen errors gracefully
- *Stress handling:* how does the system cope when exceeding various limits?
- *Recoverability:* it is possible to recover and continue using the product after a fatal error.
- *Da*
- *Sa*
- *Di*
- *Tr*

Usability. *Is the product easy to use?*

- *Affordance:* product invites to discover possibilities of the product.
- *Intuitiveness:* it is easy to understand and explain what the product can do.
- *Minimalism:* there is nothing redundant about the product's content or appearance.
- *Learnability:* it is fast and easy to learn how to use the product.
- *Memorability:* once you have learnt how to do something you don't forget it.
- *Discoverability:* the product's information and capabilities can be discovered by exploration of the user interface.
- *Op*
- *Int* acting with the application (via GUI or API).
- *Co*
- *Cl* understood, leaving no room for doubt?
- *Errors:* there are informative error messages, difficult to make mistakes and easy to repair after making them.
- *Consistency:* behavior is the same throughout the product, and there is one look & feel.
- *Tailorability:* default settings and behavior can be specified for flexibility.
- *Accessibility:* the product is possible to use for as many people as possible, and meets applicable accessibility standards.
- *Documentation:* there is a Help that helps, and matches the functionality.

Charisma. *Does the product have "it"?*

- *Un*
- *Sa*
- *Pr* el fit for purpose?
- *At* ses?
- *Curiosity:* will users get interested and try out what they can do with the product?
- *Entrancement:* do users get hooked, have fun, in a flow, and fully engaged when using the product?
- *Hype:* should the product use the latest and greatest technologies/ideas?
- *Expectancy:* the product exceeds expectations and meets the needs you didn't know you had.
- *Attitude:* do the product and its information have the right attitude and speak to you with the
- *Directness:* are (first) impressions impressive?
- *Story:* are there compelling stories about the product's inception, construction or usage?

Security. *Does the product protect against unwanted usage?*

- *Au*
- *Au*
- *Pr*
- *Se*
- *Secrecy:* the product should under no circumstances disclose information about the underlying systems.
- *Invulnerability:* ability to withstand penetration attempts.
- *Virus-free:* product will not transport virus, or appear as one.
- *Piracy Resistance:* no possibility to illegally copy and distribute the software or code.
- *Compliance:* security standards the product adheres to.

Performance. *Is the product fast enough?*

- *Capacity:* the many limits of the product, for different circumstances (e.g. slow network.)
- *Resource Utilization:* appropriate usage of memory, storage and other resources.
- *Re*
- *Av*
- *Th*
- *En*
- *Feedback:* is the feedback from the system on user actions appropriate?
- *Scalability:* how well does the product scale up, out or down?

IT-bility. *Is the product easy to install, maintain and support?*

- *System requirements:* ability to run on supported configurations, and handle different environments or missing components.
- *Deployability:* product can be rolled-out by IT department to different types of (restricted) users and environments.
- *Maintainability:* are the product and its artifacts easy to maintain and support for customers?
- *Testability:* how effectively can the deployed product be tested by the customer?

Compatibility. *How well does the product interact with software and environments?*

- *Hardware Compatibility:* the product can be used with applicable configurations of hardware components.
- *Forward Compatibility:* will the product be able to use artifacts or interfaces of future versions?
- *ts on the environment, e.g. energy efficiency, switch-offs, power-saving modes, telecommuting.*
- *nce:* the product conforms to applicable standards, regulations, laws or ethics.

oftware Quality Characteristics

These characteristics are not directly experienced by end users, but can be equally important for successful products.

- *Troubleshootable:* is it easy to pinpoint errors (e.g. log files) and get help?
- *Debugging:* can you observe the internal states of the software when needed?
- *Versatility:* ability to use the product in more ways than it was originally designed for.

Testability. *Is it easy to check and test the product?*

- *Traceability:* the product logs actions at appropriate levels and in usable format.
- *Controllability:* ability to independently set states, objects or variables.
- *Observabilit* ability to observe things that should be tested.
- *Monitorabil*
- *Isolateabilit*
- *Stability:* ch
- *Automation:* are there public or hidden programmatic interface that can be used.
- *Information:* ability for testers to learn what needs to be learned...
- *Auditability:* can the product and its creation be validated?

Maintainability. *Can the product be maintained and extended at low cost?*

- *Flexibility:* the ability to change the product as required by customers.
- *Extensibility:* will it be easy to add features in the future?
- *Simplicity:* the code is not more complex than needed, and does not obscure test design, execution and evaluation.
- *Readability:* the code is adequately documented and easy to read and understand.

Portability. *Is transferring of the product to different environments enabled?*

- *Reusability:* can parts of the product be re-used elsewhere?
- *aptability:* is it easy to change the product to support a different environment?
- *mpatibility:* does the product comply with common interfaces or official standards?
- *ternationalization:* it is easy to translate the product.
- *calization:* are all parts of the product adjusted to meet the needs of the targeted culture/country?
- *User Interface-robustness:* will the product look equally good when translated?

IT-bility. *Is the product easy to install, maintain and support?*

Capability. *Can the product perform valuable functions?*

Compatibility. *How well does the product interact with software and environments?*

Reliability. *Can you trust the product in many and difficult situations?*

Supportability. *Can customers' usage and problems be supported?*

Usability. *Is the product easy to use?*

Testability. *Is it easy to check and test the product?*

Charisma. *Does the product have "it"?*

Maintainability. *Can the product be maintained and extended at low cost?*

Security. *Does the product protect against unwanted usage?*

Portability. *Is transferring of the product to different environments and languages enabled?*

Performance. *Is the product fast enough?*

...where factors like security, performance, capabilities will be everywhere...

Dealing with reality

I, and you, will never be finished with our testing.

There are always more interesting tests to perform.

Time is limited.

So you have to skip things, but when you know more, that will be easier.

But it will still be difficult.

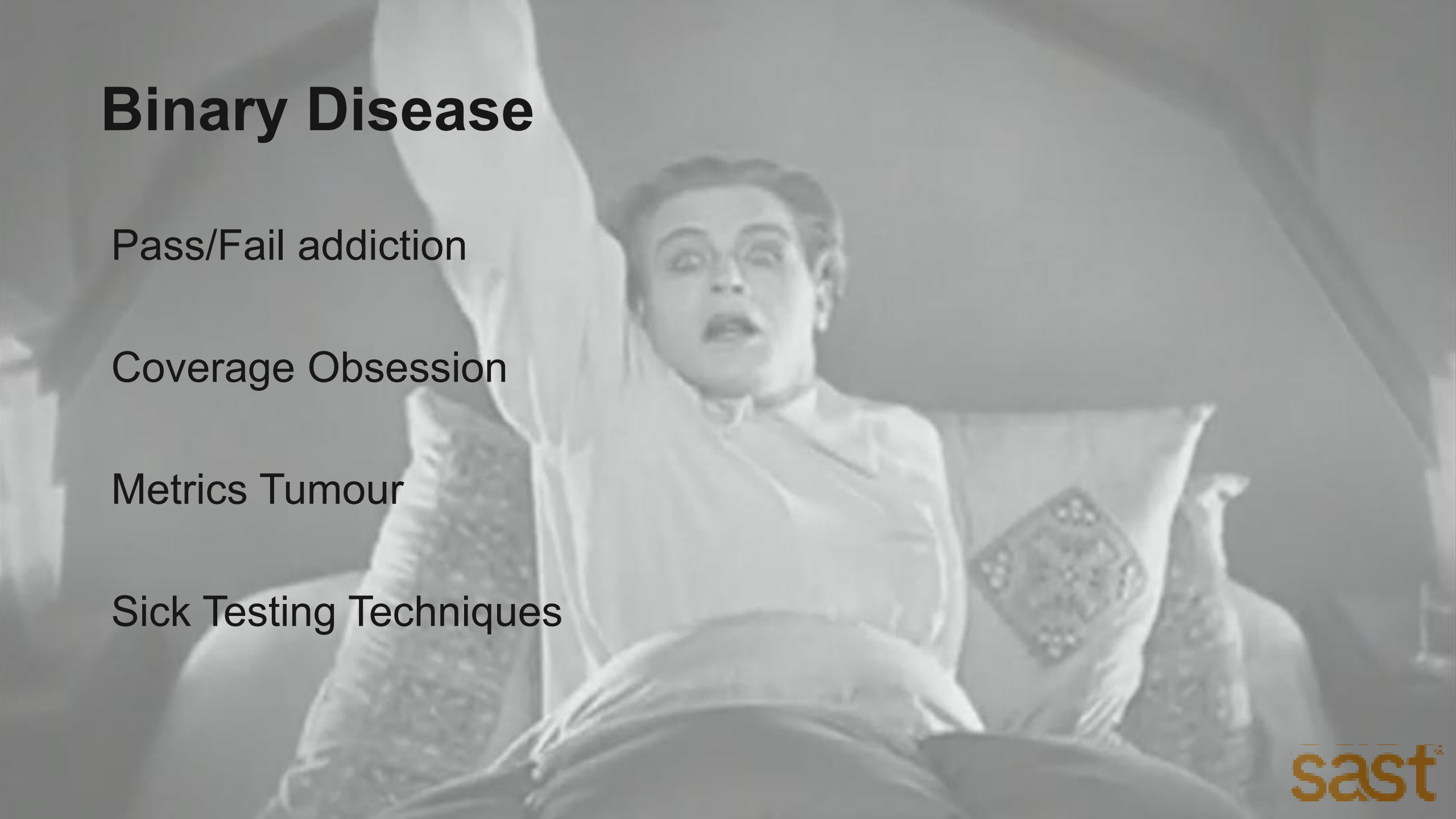
what if I don’t do anything of this?

Skipping the deeper understanding

You might be fine without understanding of relations.

But you might also be ill…

Binary Disease



Pass/Fail addiction

Coverage Obsession

Metrics Tumour

Sick Testing Techniques

sast

We are humans

It’s not only that software is made for humans, by humans

We are making new, unique things; providing value

Humans are superior to machines at:

* understanding what’s important
* judgment
* separating right from wrong
* dealing with the inevitable unknown

Finale

By understanding relations you will test better, and communicate better around qualities, problems, risks

Find YOUR best ways to understanding (relations)

Let testing be difficult

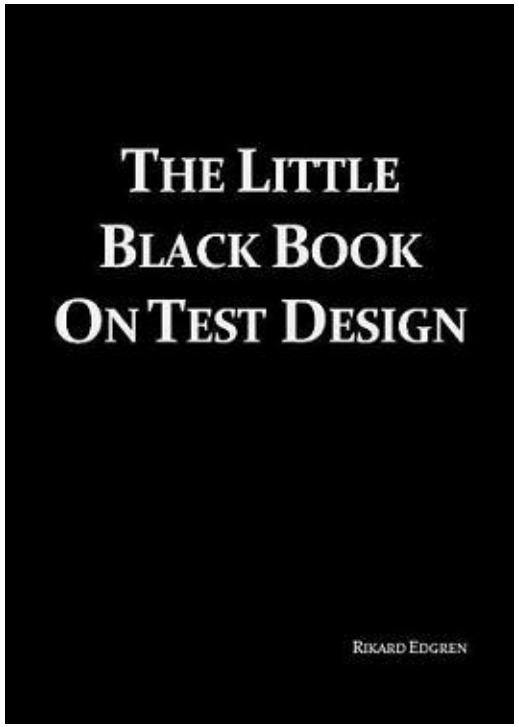
References

The Little Black Book on Test Design (Edgren)

Heuristic Test Strategy Model (Bach)

Product Ecology (Cox)

Intro: Mårten Ivert & Henrik Emilsson




Questions?
rikard.edgren@inera.se