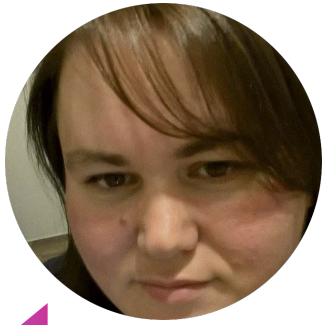
Hello!

Welcome at Visma

April 2017 #VismaAClabs @ligaac_labs

Who are we



Monica Iovan

Zoltan Sandor

Adriana Hazulea

Ionut Bradatan

Emil Craciun

Who are you?

1. Team up with your “papermate”
2. Draw each other:

   don’t look at the paper & don’t lift the pen!
3. Interview each other to answer the questions

10 min.

1.
Agenda

Let’s start our journey

Breakdown of our time together

	40h
Day 1	Project start-up & requirements
Day 2	Become a UX hacker
Day 3-6	Implement
Day 7-9	Test
Day 10	Deploy

Typical day

	4h
09:00 - 10:00	Work
10:00 - 10:10	Break
10:10 - 11:30	Work
11:30 - 12:00	Lunch
12:00 - 13:00	Work

2.
How do we do requirements

High level overview of how a project starts

Requirements content

Introduction	General Description	Specific Requirements	Analysis Models	Change Management Process
• Purpose • Scope • Definitions, Acronyms, and Abbreviations • References • Overview	• Product Perspective • Product Functions • User Characteristics • General Constraints • Assumptions and Dependencies	• External Interface Requirements • Functional Requirements • Use Cases • Classes / Objects • Non-Functional Requirements • Inverse Requirements • Design Constraints • Logical Database Requirements • Other Requirements	• Sequence Diagrams • Data Flow Diagrams (DFD) • State-Transition Diagrams (STD)	

Introduction

all of the information needed to adequately design and implement the software

Introduction

Purpose

and the (intended) audience for which it is written.

Scope

name, what will, and, if necessary, will not do, relevant benefits, objectives, and goals

Overview

what the rest of the doc contains, explain how the SRS is organized.

General description

general factors that affects the product and its requirements.

General description

Product Perspective
puts the product into perspective with other related products or projects

Product Functions
provide a summary of the functions that the software will perform.

User Characteristics
general characteristics of the eventual users of the product that will affect the specific requirements

General Constraints
description of any other items that will limit the developer's options for designing the system

Specific requirements

Specific requirements

Each requirement in this section should be:

- Correct
- Traceable (both forward and backward to prior/future artifacts)
- Unambiguous
- Verifiable (i.e., testable)
- Prioritized (with respect to importance and/or stability)
- Complete
- Consistent
- Uniquely identifiable (usually via numbering like 3.4.5.6)

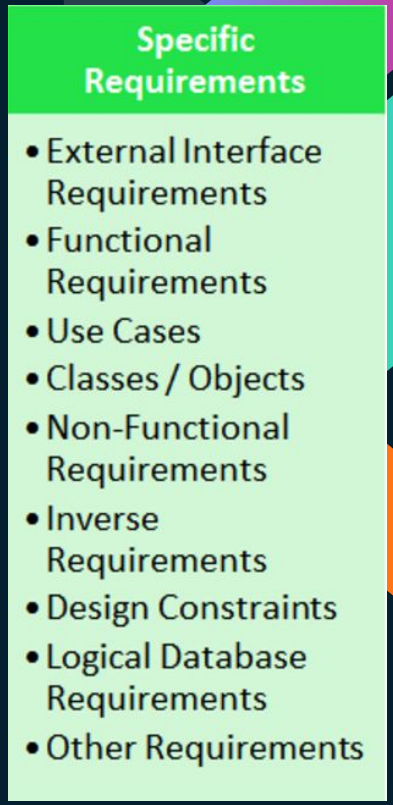
External Interface Requirements

User Interfaces

Hardware Interfaces

Software Interfaces

Communications Interfaces



Functional Requirements

This section describes specific features of the software project.

<Functional Requirement or Feature #1>

3.2.1.1 Introduction

3.2.1.2 Inputs

3.2.1.3 Processing

3.2.1.4 Outputs

3.2.1.5 Error Handling

Use Cases

Classes / Objects

Specific Requirements

- External Interface Requirements
- Functional Requirements
- Use Cases
- Classes / Objects
- Non-Functional Requirements
- Inverse Requirements
- Design Constraints
- Logical Database Requirements
- Other Requirements

Non-Functional Requirements

Performance

Reliability

Availability

Security

Maintainability

Portability

Specific Requirements

- External Interface Requirements
- Functional Requirements
- Use Cases
- Classes / Objects
- Non-Functional Requirements
- Inverse Requirements
- Design Constraints
- Logical Database Requirements
- Other Requirements

Inverse Requirements

Design Constraints

other standards, company policies, hardware limitation, etc. that will impact this software project.

Logical Database Requirements

Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc.

Other Requirements

Specific Requirements

- External Interface Requirements
- Functional Requirements
- Use Cases
- Classes / Objects
- Non-Functional Requirements
- Inverse Requirements
- Design Constraints
- Logical Database Requirements
- Other Requirements

Analysis Models

Analysis models

Sequence Diagrams

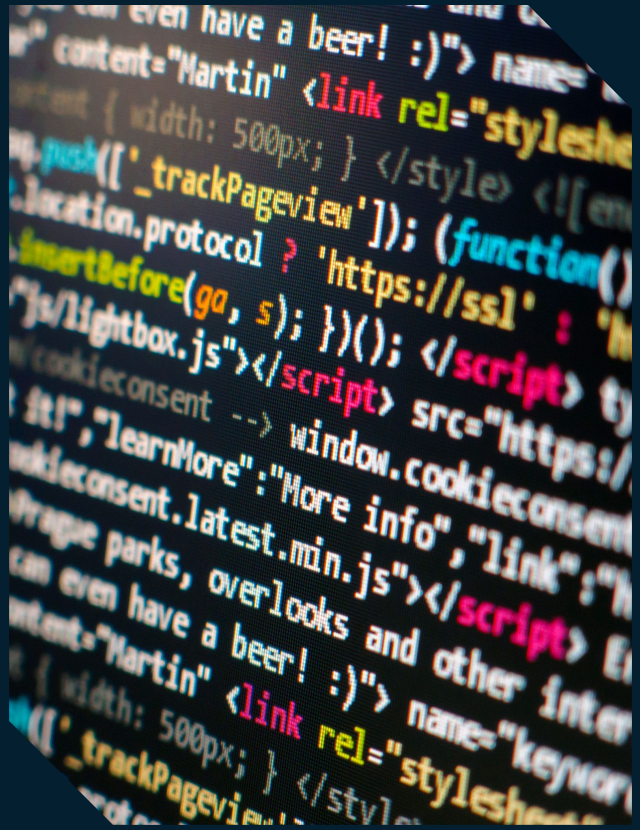
Data Flow Diagrams (DFD)

State-Transition Diagrams (STD)

Change Management Process

3.
Let’s start our own project

On your marks, get set, GO!



The PROBLEM

Don't Just Start Development,
Solve A Problem

The story of
Philip Pirrip

Let me tell you the story of Philip Pirrip, a brilliant young professional writer and his journey to work excellence

The story of Pip

The first memory Pip has is about his grandparent's attic - the old smell of used books, the creaking wooden boards and the dust particles that danced silently in the sun rays like a game of hiding and seek.

He spent a long time there with his small furry friend - Arnold. The cat was not impressed by the large amounts of book, maps, and chests available, he was there usually to take a nap in the daytime, and he loved the silence. **For Pip everything changed.**

The love of books and endless hours of reading his grandfather's collections led Pip to the University of Charles Dickens where legendary professors took his talent further.

Now he just rented a small loft in the downtown area of his town and recently launched his first official blog post on his new website.

"**I have great expectations from this**," he said to Arnold. The cat heard him but was not impressed and continued his nap.

The story of Pip

After six months he's already a successful blogger, a freelance writer working with 7 different publications at a time and writing the 8th chapter of the first draft of his first book "**Great Expectations.**"

Soon he quickly realized that all this work needs to be organised somehow. "My office is a mess" he said in slight desperation one Saturday morning. "**My work is so chaotic... I need a simple way to manage it all** or else I'll go insane!"

What should I do?

“I want something that is easy to use and quick. I would much rather spend my time creating valuable work...”

Too much manual work

A simple cost effective electronic solution is in demand.

costly

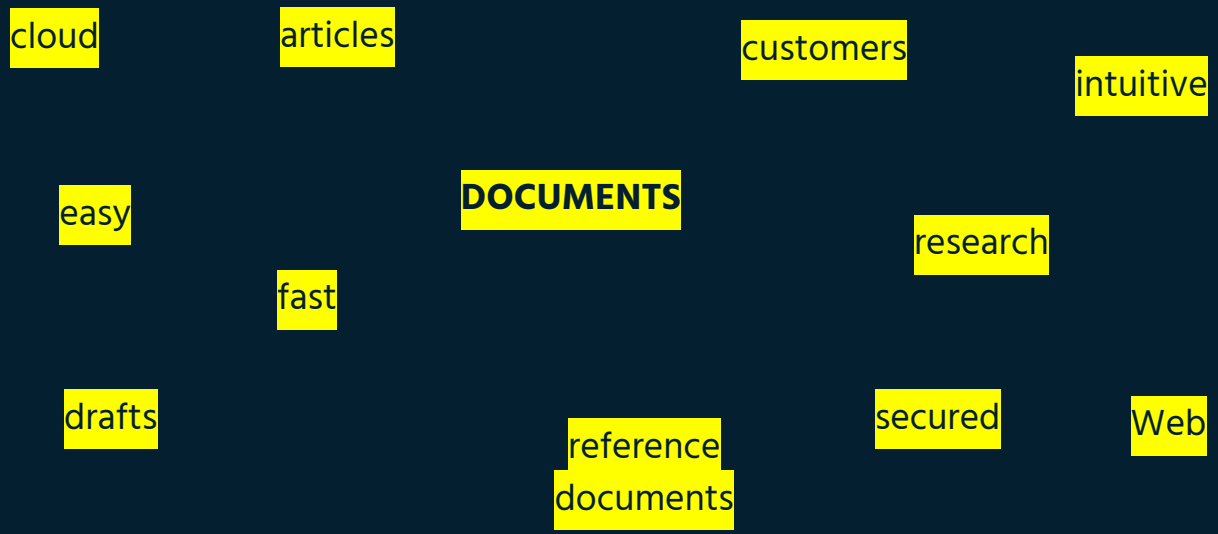
time

boring

hard

The solution

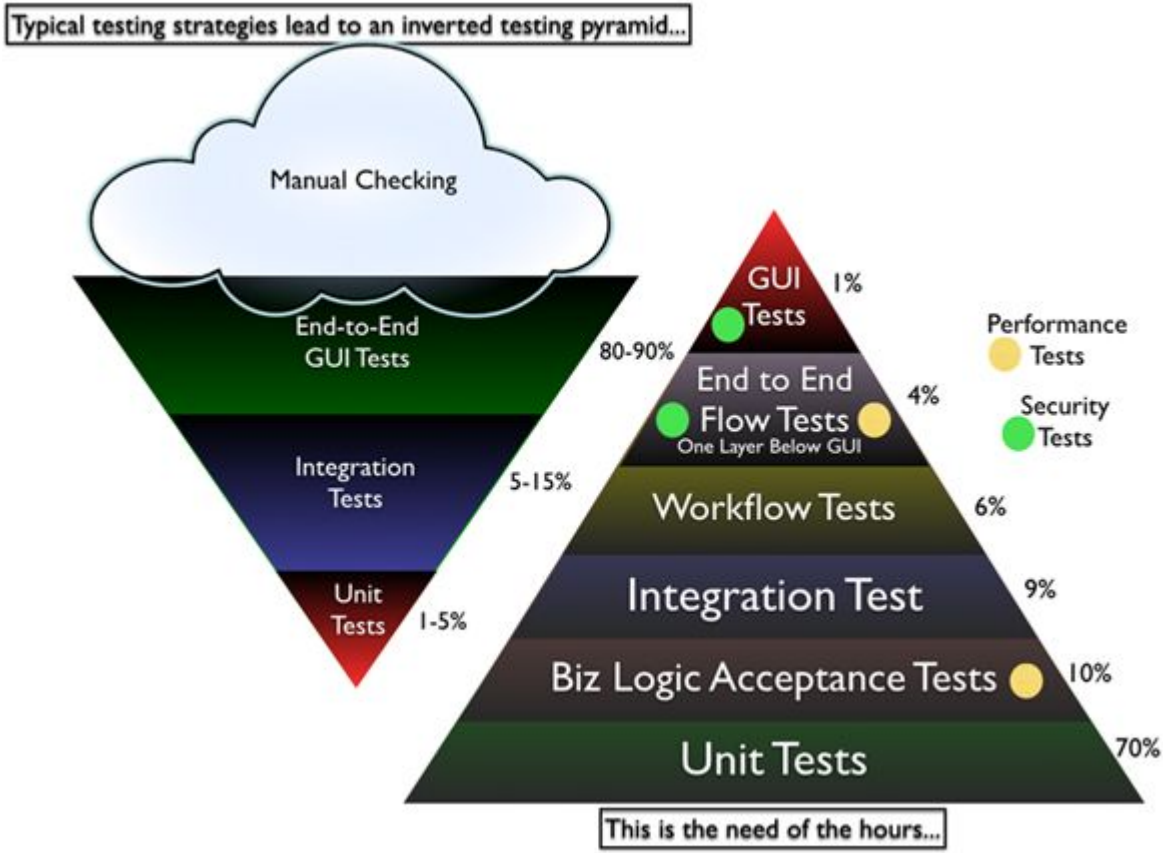
A simple cost effective electronic solution is in demand.



I LIKE HOW WE'VE HAD THE INTERNET FOR DECADES, YET "SENDING FILES" IS SOMETHING EARLY ADOPTERS ARE STILL FIGURING OUT HOW TO DO.

4.
Testing

What and when do we test?


Typical testing strategies lead to an inverted testing pyramid...
Manual Checking
End-to-End GUI Tests
80-90%
Integration Tests
5-15%
Unit Tests
1-5%
GUI Tests
1%
End to End Flow Tests
One Layer Below GUI
4%
Workflow Tests
6%
Integration Test
9%
Biz Logic Acceptance Tests
10%
Unit Tests
70%
Performance Tests
Security Tests
This is the need of the hours...

Cloud Storage

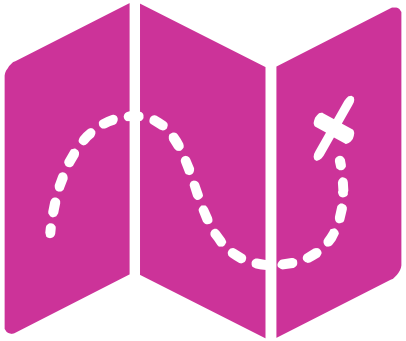
Your cloud, your way

Introducing **Cloud Storage**

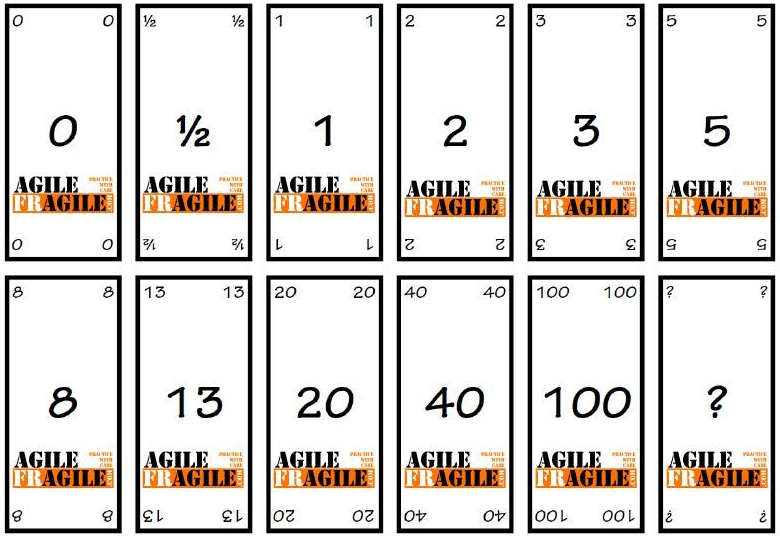
Structure

Scenario: Some determinable business situation
Given some precondition
    And some other precondition
When some action by the actor
    And some other action
    And yet another action
Then some testable outcome is achieved
    And something else we can check happens too
But something doesn't happen

User stories



⏳ 10 min.



A good plan implemented today is better than a perfect plan implemented tomorrow.

Reflections from today

What have we learned?

“Enjoying success requires the ability to adapt.
Only by being open to change will you have a true opportunity to get the most from your talent.”

- adapt to it, don't fight it

Tomorrow we'll become UX hackers

@lonut is going to present and he'll lead you in the realm of users and wireframes

Thank you!

We meet again tomorrow at 9:00

Day 2

Welcome back

April 2017 #VismaAClabs @ligaac_labs

What did we learn yesterday?

Quick overview

UX Hacks

Or the super-lazy way for increased UX nirvana

Goals with this brief training

Become UX Hackers

#1 Improve the **understanding** of UX

#2 Establish a **user-centred work process**

#3 Enable you to **perform own UX-related tasks**

0.
Agenda

Let’s begin

Breakdown of our time together

4 awesome hours

1	Introduction to UX
2	How does Visma UX
3	User research & user centered design
4	Paper prototyping
5	User interviews
6	Wireframing like a pro

What does UX stand for?

The many names and meanings of UX

- User experience (UX)
- Human Computer Interaction
- User friendliness
- Usability

  "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use." (ISO 9241-11)

- Interaction design (IxD)
- Service design

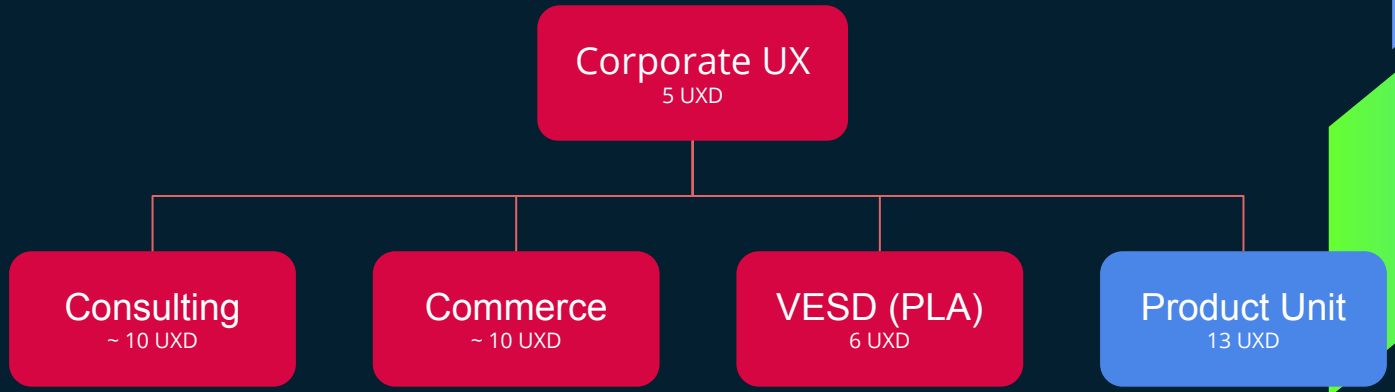
Science!

- Human Computer Interaction
- Human Factors
- Cognitive Science
- Psychology
- Anthropology
- Linguistics
- Neuroscience

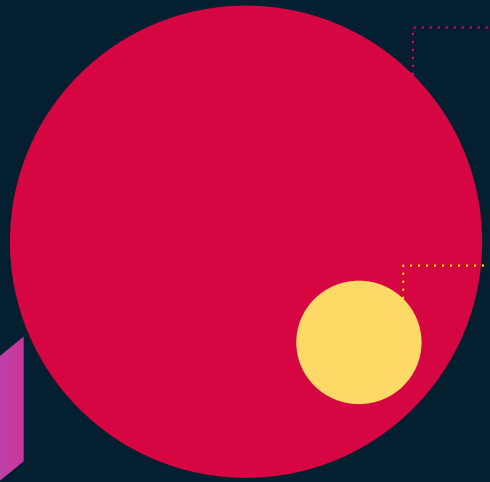
Different UX job roles

- UX Architect
- UX Designer
- Graphical Designer
- UX Developer

The UX in Visma environment



The tasks of the UX resources



Assist teams with UX expertise (**90%**)

- Perform user studies
- Create sketches / prototypes

Knowledge sharing (**10%**)

- Maintain the UX portal
- Write UX guidelines
- Build and quality assure reference libraries
- Hold UX trainings

Coordinates UX in different divisions

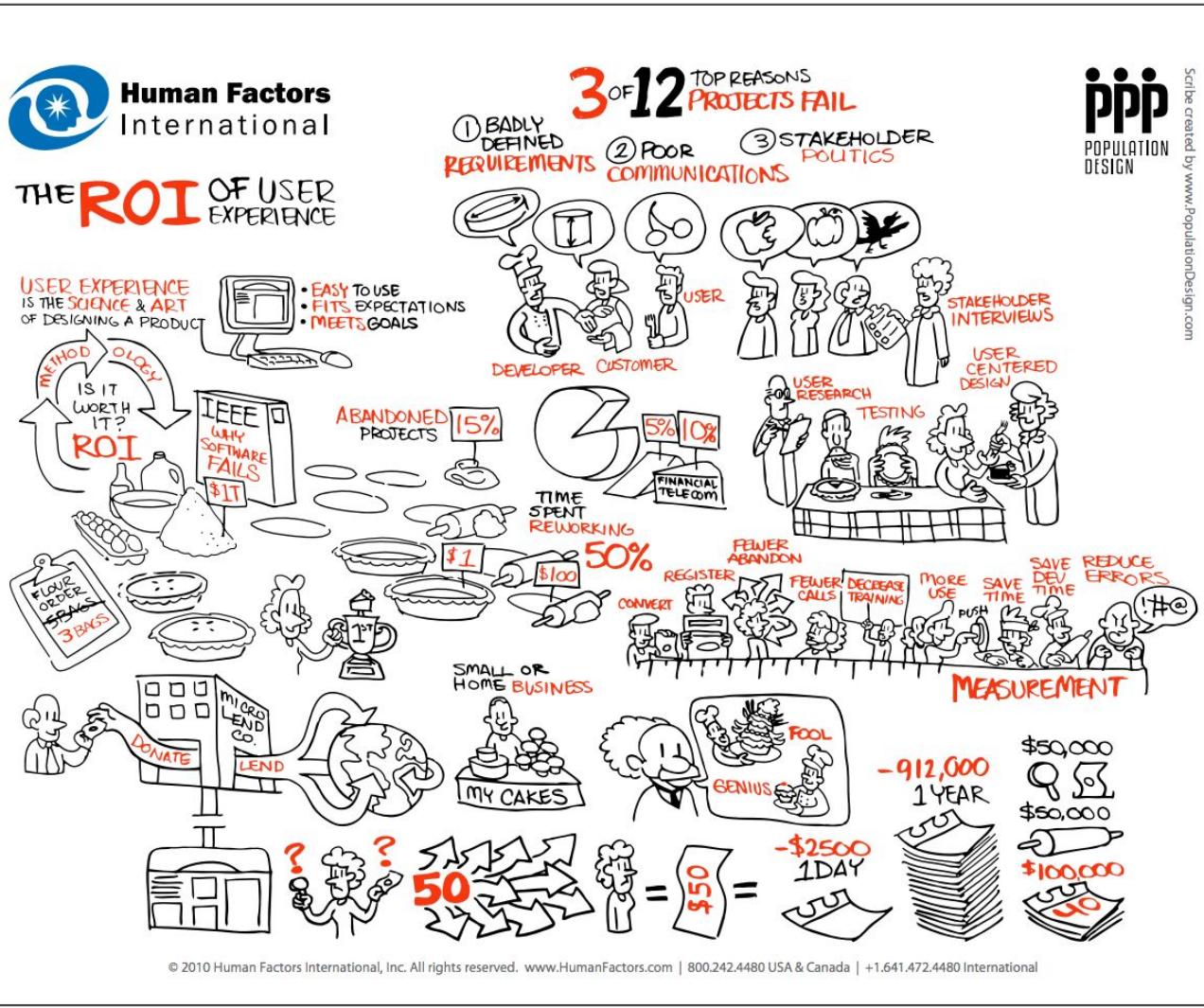
Produces all graphical design

NC3 **Visma Nordic Cool 3**

Visma UX Guidelines

Visma Web Library


-912,000
1 YEAR
$50,000
-$2500
1 DAY



*"Once a piece of software is released, the **cost of fixing an error** can be 100 times as high as it would have been during the design and development stage."*

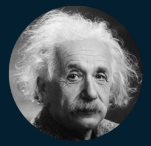
*Software specialists spend 40-50 % of their time on **avoidable rework**, basically work that's not done right the first time.*

3 of 12 top reasons to why software projects fail are directly related to ***poor user-centred design work and poor user experience.***

Good UX and a UCD process

- reduces costs for development, maintenance, redesign, support, training...
- increases revenue, traffic, market shares, easy of learning, ease of use
- improves customer retention, trust in the software, user satisfaction...

"Any intelligent fool can make things bigger and more complex. It takes a touch of genius - and a lot of courage – to move in the opposite direction."



“Simple” isn’t what you think

- minimize number of steps in a flow
- minimize time required to solve task
- minimize number of features
- minimize elements on each page

Cognitive Overhead

“how many logical connections or jumps your brain has to make in order to understand or contextualize the thing you’re looking at.”

How do we create a good user experience and a user friendly application/service?

What is UX?

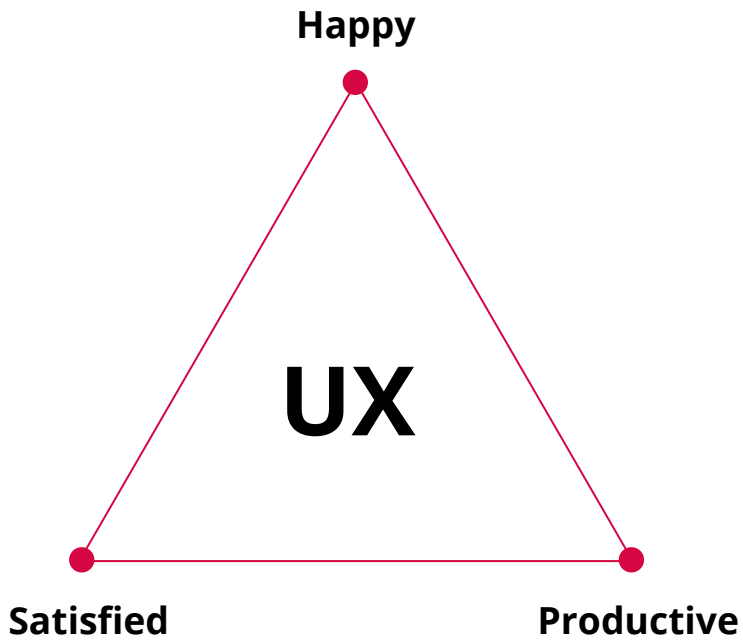
"User experience" (UX) encompasses ***all aspects*** *of the* ***end-user****'s interaction with the* ***company****, its* ***services****, and its* ***products****.*

Jakob Nielsen

Don Norman

“How a person feels about interacting with a product.”

Desired result



Happy

- Good news travels fast
- Increased sales

Satisfied

- Increased loyalty
- Fewer support calls/requests
- Ideas vs. complaints

Productive

- Most Valuable Proponents (MVP)

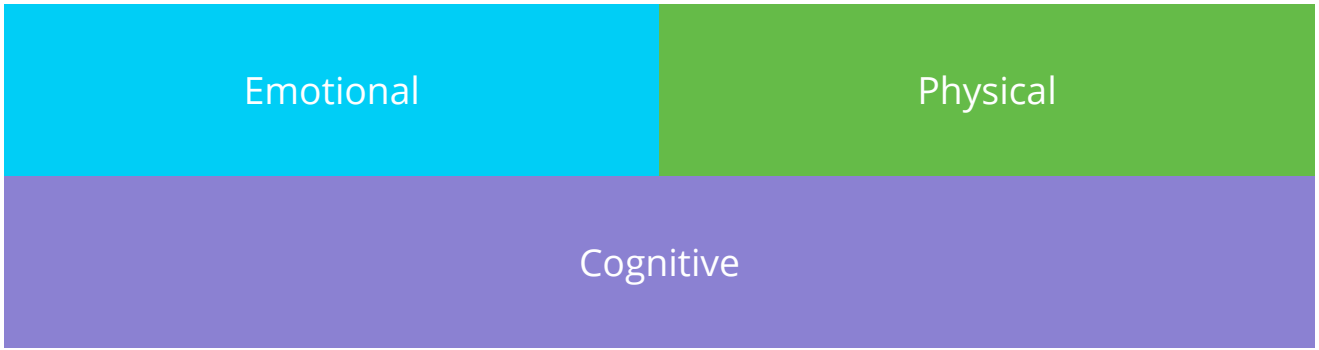
Hacking feelings?

- Person
- How they feel
- Human brain
- How do we hack that?
- Root API

The deeper meaning

“The measure of **friction** between a user and his **goal**”

Types of friction



Cognitive friction

- Needless brain activity
- The brain CPU cycles
- Conscious
- Subconscious
- Target of our UX hacks

  (mind and vision)

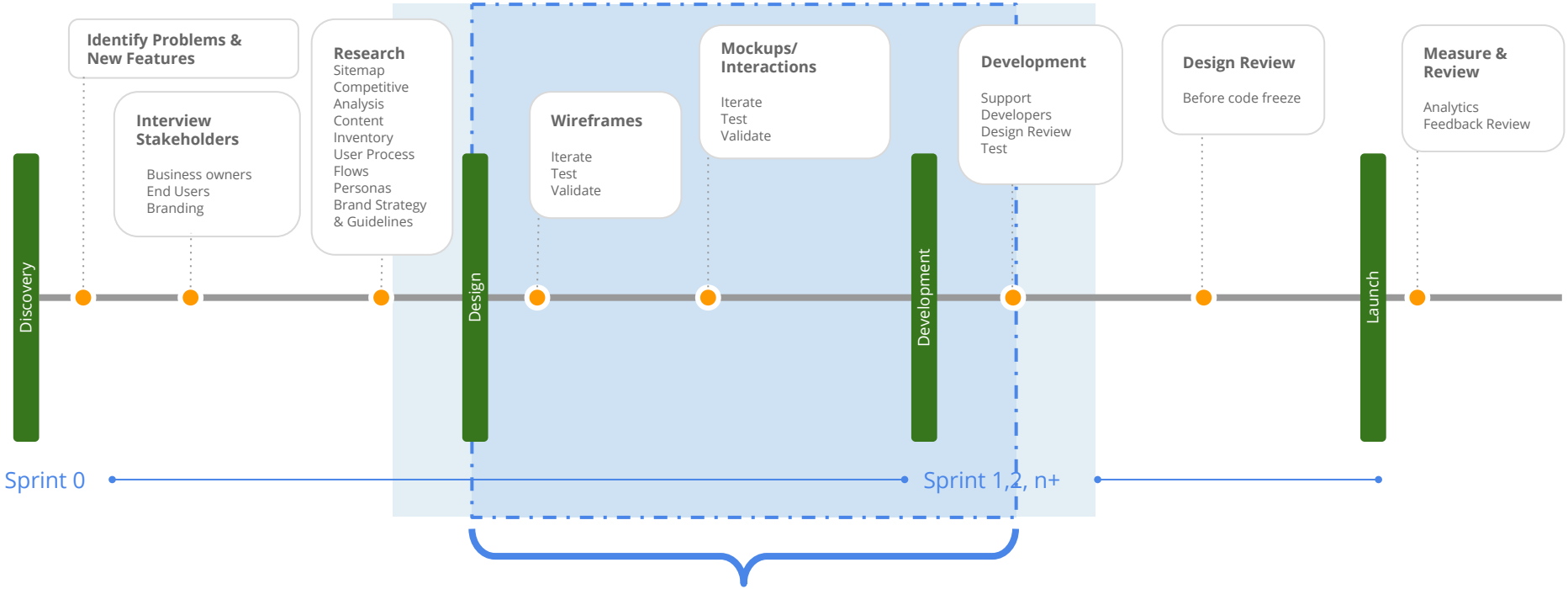
UX Layers

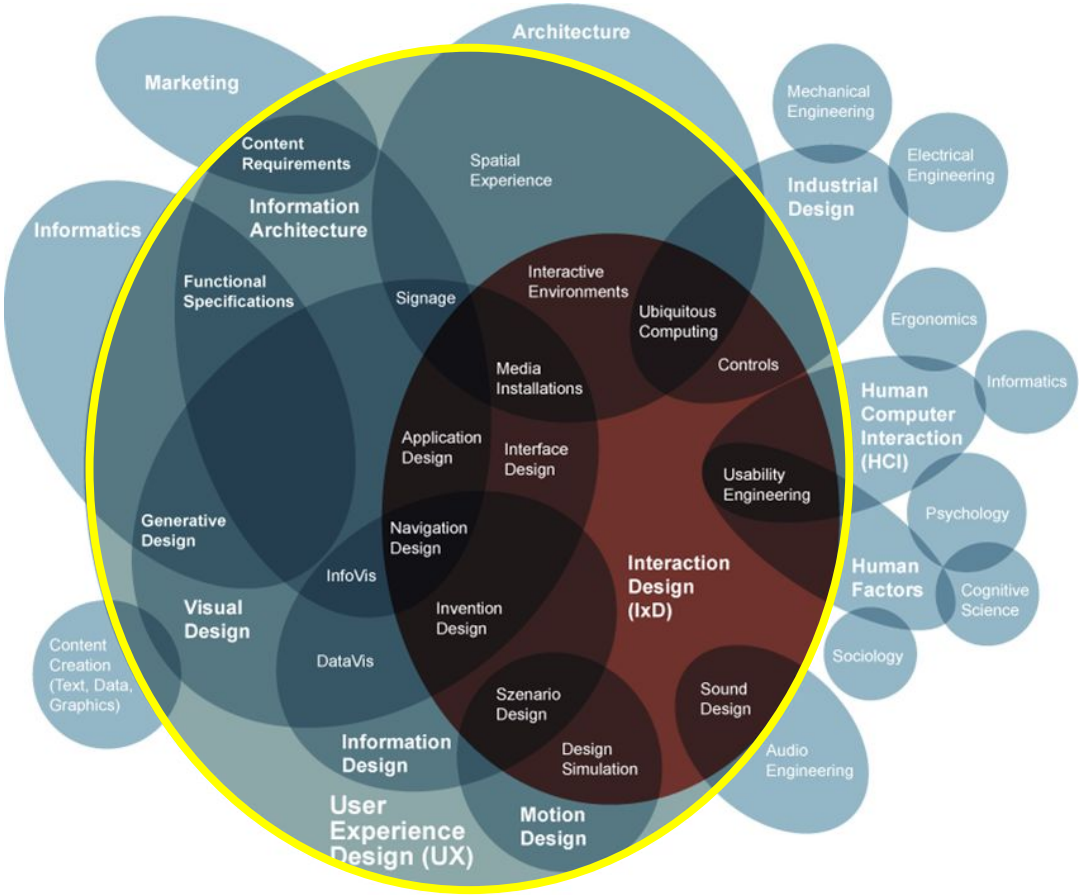
- Visual Language (Nordic Cool)
- Motion Graphics
- Information Design
- Information Architecture
- Interaction Design
- User Research
- Experience Strategy

UI

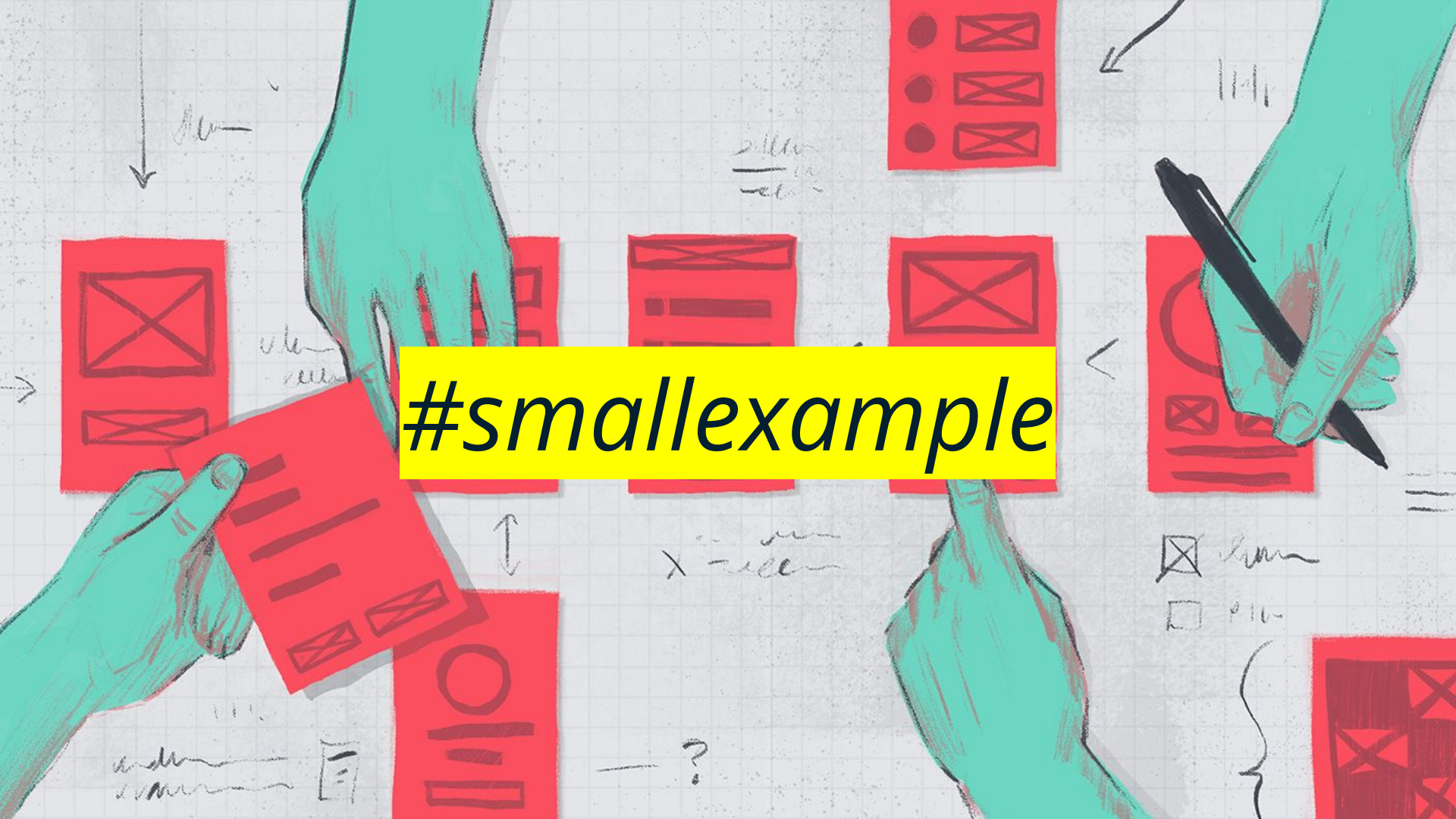
UX


Aesthetics
UX guidelines & libraries
Interface dynamics
UCD process
Usability
Interface standards
UX guidelines & libraries


Identify Problems & New Features
Interview Stakeholders
Business owners
End Users
Branding
Research
Sitemap
Competitive Analysis
Content Inventory
User Process Flows
Personas
Brand Strategy & Guidelines
Wireframes
Iterate
Test
Validate
Mockups/ Interactions
Iterate
Test
Validate
Development
Support Developers
Design Review
Test
Design Review
Before code freeze
Measure & Review
Analytics
Feedback Review
Discovery
Design
Development
Launch
Sprint 0
Sprint 1,2, n+


Architecture
Marketing
Mechanical Engineering
Electrical Engineering
Industrial Design
Content Requirements
Spatial Experience
Information Architecture
Informatics
Functional Specifications
Signage
Interactive Environments
Ubiquitous Computing
Ergonomics
Controls
Media Installations
Informatics
Human Computer Interaction (HCI)
Application Design
Interface Design
Usability Engineering
Psychology
Generative Design
Navigation Design
Interaction Design (IxD)
Human Factors
InfoVis
Cognitive Science
Visual Design
Invention Design
Content Creation (Text, Data, Graphics)
DataVis
Sociology
Szenario Design
Sound Design
Information Design
Design Simulation
Audio Engineering
User Experience Design (UX)
Motion Design


UX Marathon


#smallexample

DCE Mobile - search user component

Simona Secosan to me 13 Apr

Hi,

We have created a component (similar with a dialogue ☺) to be used for selecting one or more users as recipient if invoice will be sent on mobile.
Attached there are several screenshots of how it looks now (We - I - are not pleased of how it looks) and we will appreciate several suggestions/improvements.

Expected functionality:
- autocomplete component which fetch several users
- scroll in list
- search user support to filter the list
- multiple selection

It can be also another entire screen but we need to consider the view when is just one item in list; needs to be considered that we might be constraint by the used component supported functions.

If questions/uncertainties let me know and we could set a short session for more details.

Thanks!

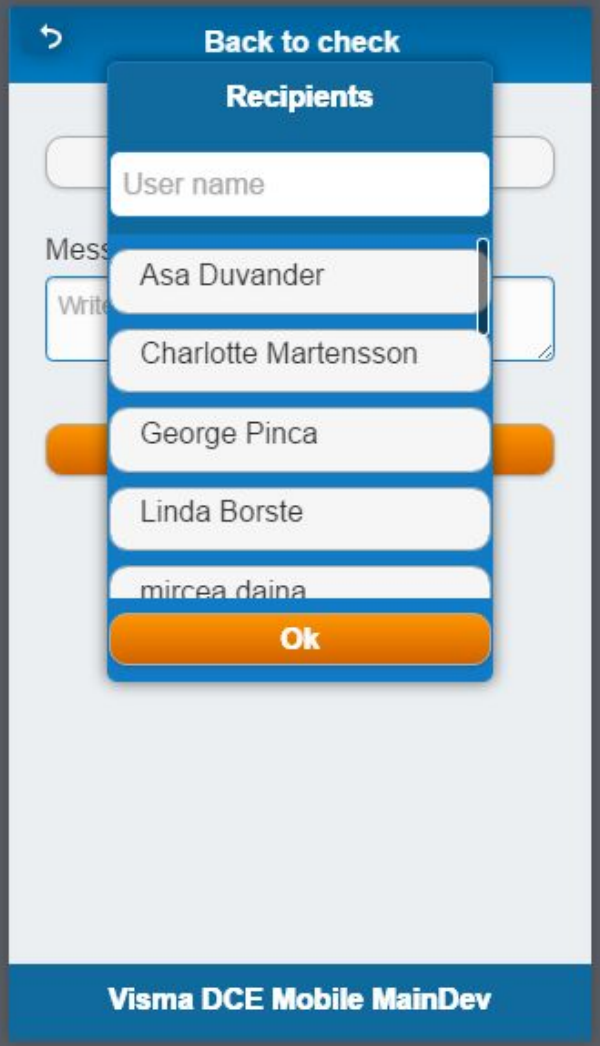
Br,
Simona

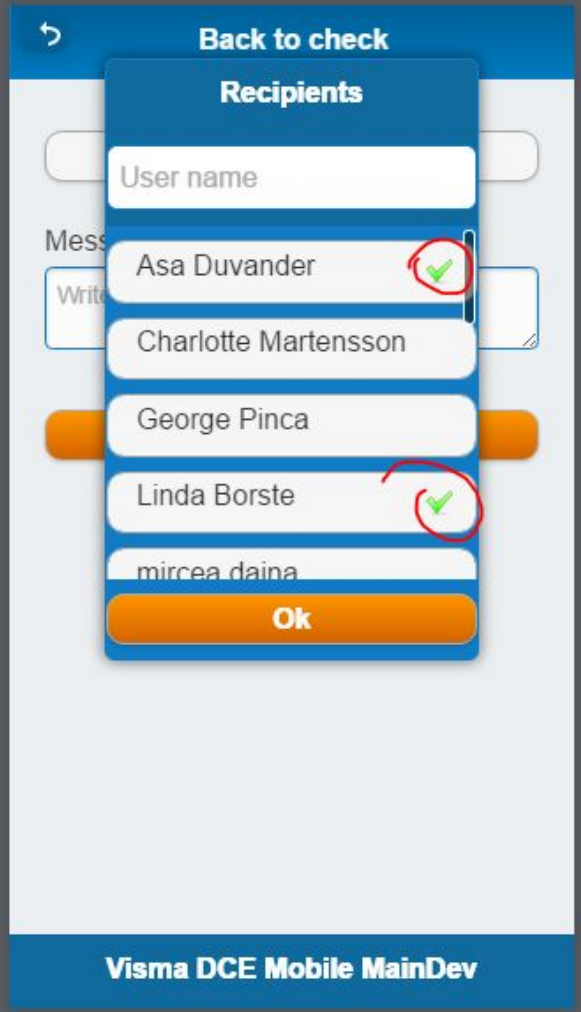
--

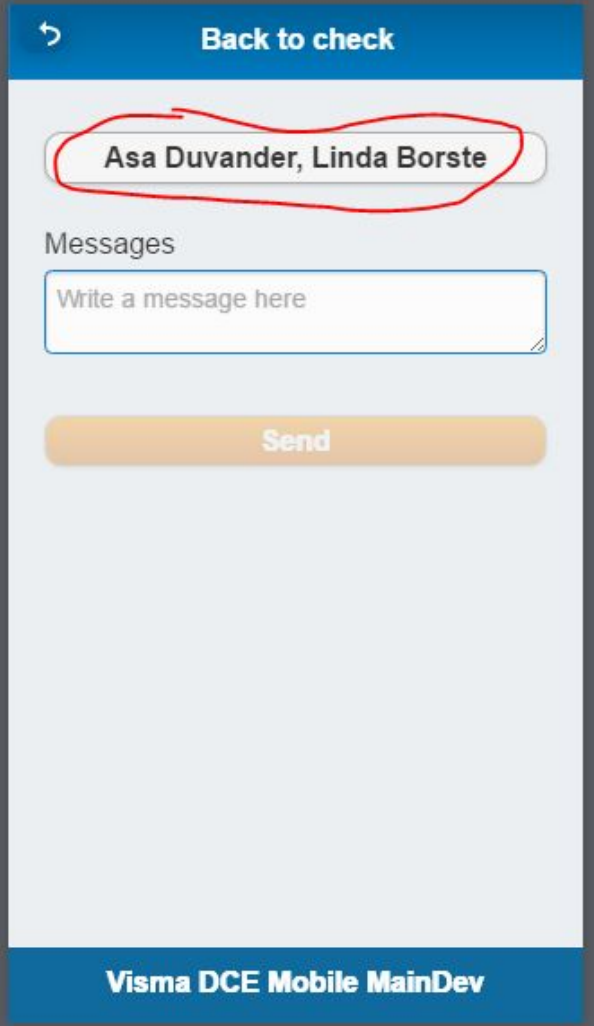
Simona Secosan

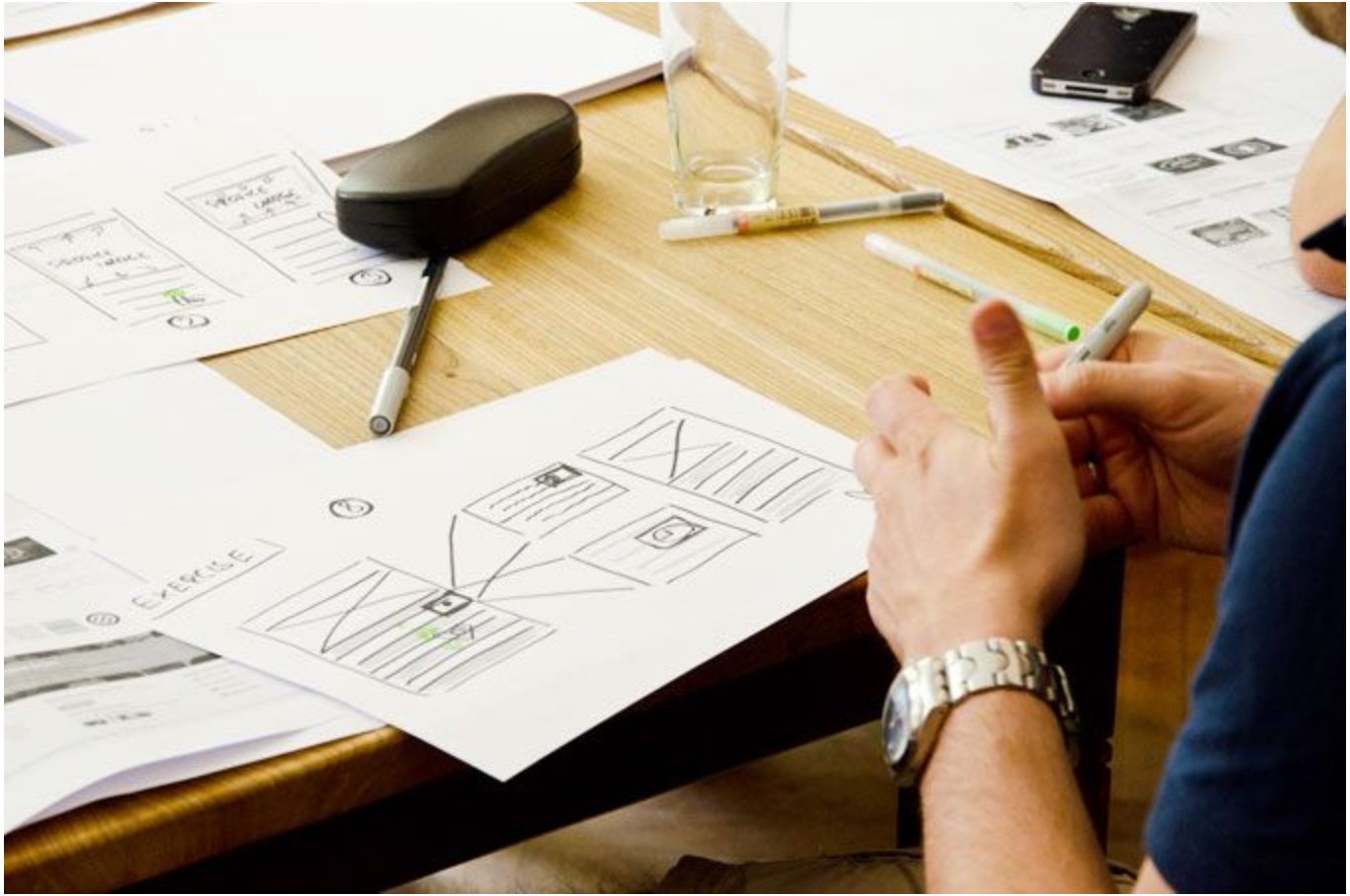
Business Analyst

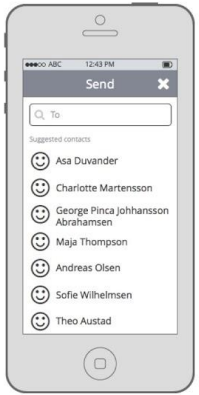
E-mail Simona.Secosan@visma.com | Switchboard ++40 269 244 768


Back to check
Recipients
User name
Asa Duvander
Charlotte Martensson
George Pinca
Linda Borste
mircea daina
Ok
Visma DCE Mobile MainDev


Back to check
Recipients
User name
Asa Duvander
Charlotte Martensson
George Pinca
Linda Borste
mircea daina
Ok
Visma DCE Mobile MainDev


Back to check
Asa Duvander, Linda Borste
Messages
Write a message here
Send
Visma DCE Mobile MainDev





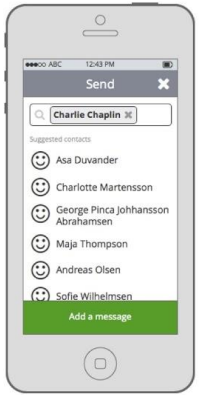
The share screen should start with selecting the recipients first. That way the cognitive friction will be kept to a minimum. The person wanting to share a particular information already has someone in mind at this time.

The only text available is 'Send', 'To' and in smaller characters 'Suggested contacts'.
Suggestions should go as following: Recent contacted persons, Highest contacted persons and a more advanced - 'this type of invoice tipically goes to the following persons' listings. Ideally would go with the last one but depends on your use cases, technical contraints and of course time allocated for this task.

Another way to go for the top menu is by removing the cancel icon top right and replacing it with a back arrow on top left. I suggested this variant based on the screens shown and trated the screen to have that pop-up feeling to it ;)

If the list is too big or the contact is not easily recognised (the reason I've added the profile icon) a search with autocomplete is also mandatory. Search fields could include e-mail adresses as well (although if you know the e-mail you would probably know the persons name) or other criterions that might be usefull at this stage (department, company id?, etc.)

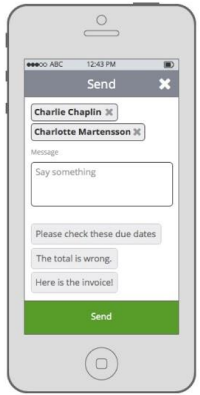
Also I thinks that while searching the contact persons should be delimited with a thing gray line in order to make it easier to see the hit/touch area.



Adding a contact should place the selected person in a combogrid with multi-select.

At this time at the very bottom the 'Add message' appears and will be always on top.

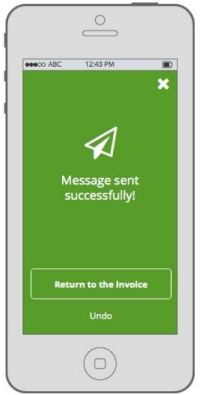
It's easy to add and remove contacts and that this way selected people are always visible.



If the message is optional the 'Message' should be renamed to 'Optional message'.

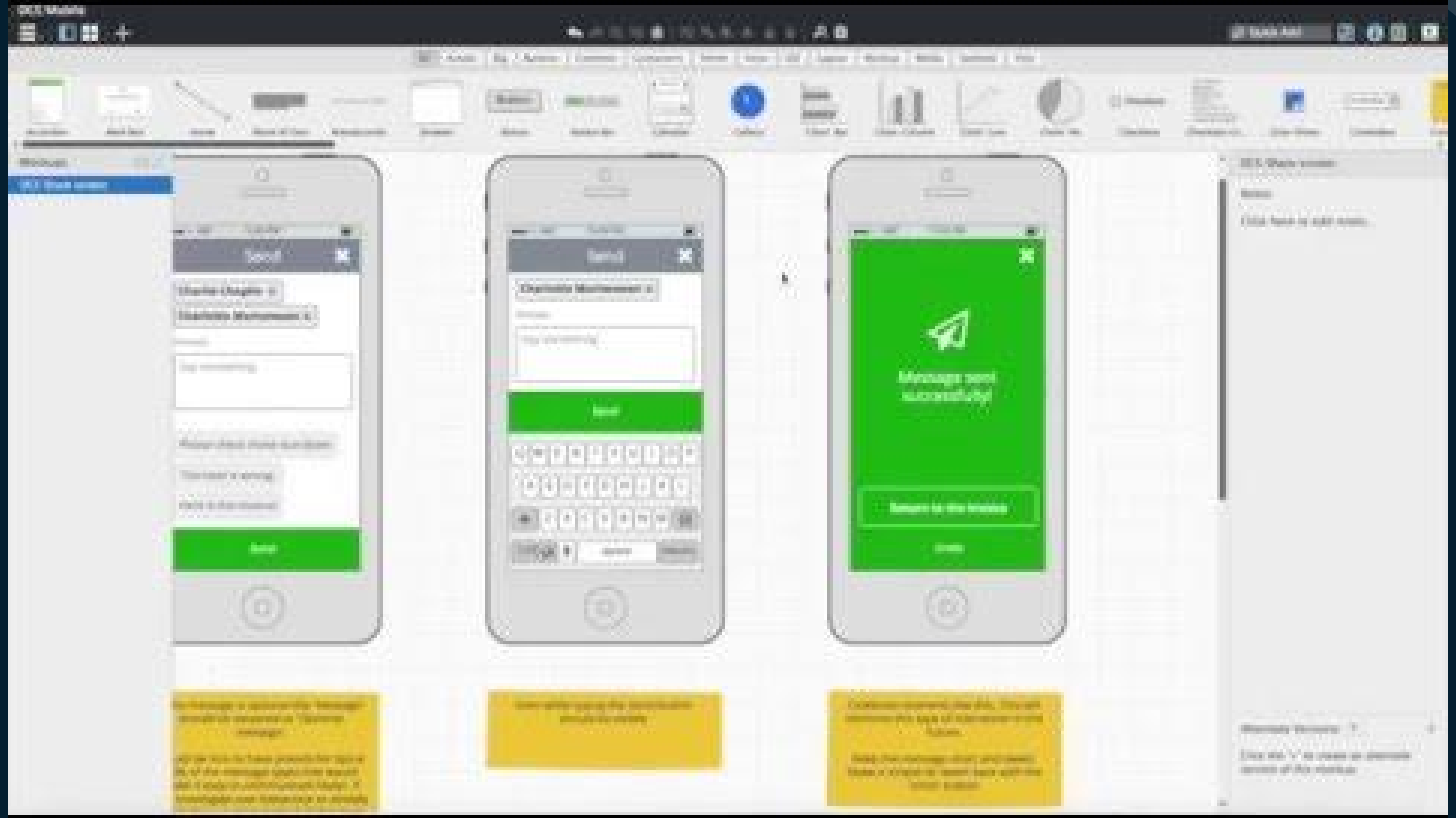
Would be nice to have presets for tipical 80% of the message types that would make it easy to communicate faster. If you investigate user behaviour or already know this use at least top 3 of the short messages implied above (translations would make it awesome).

Even while typing the Send button should be visible.



Celebrate moments like this. This will reinforce this type of interaction in the future.

Keep the message short and sweet. Make it simple to revert back with the 'Undo' button.


Send!
Send!
Message sent successfully!
Return to the timeline


Message sent successfully!
Return to the invoice
Undo

User research

User research

- Understand how the users **think** and **behave**
- Gather **facts** and **data** instead of rely on opinion or speculation

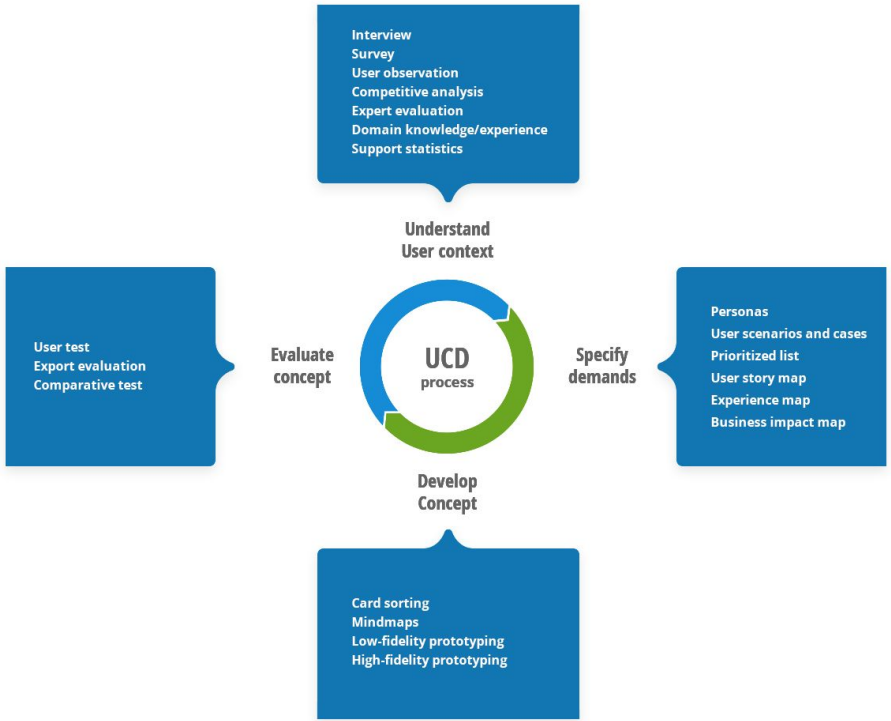
- Many different methods available

- Perform **studies**, design and **test on users** first before implementing anything
- Iterate!

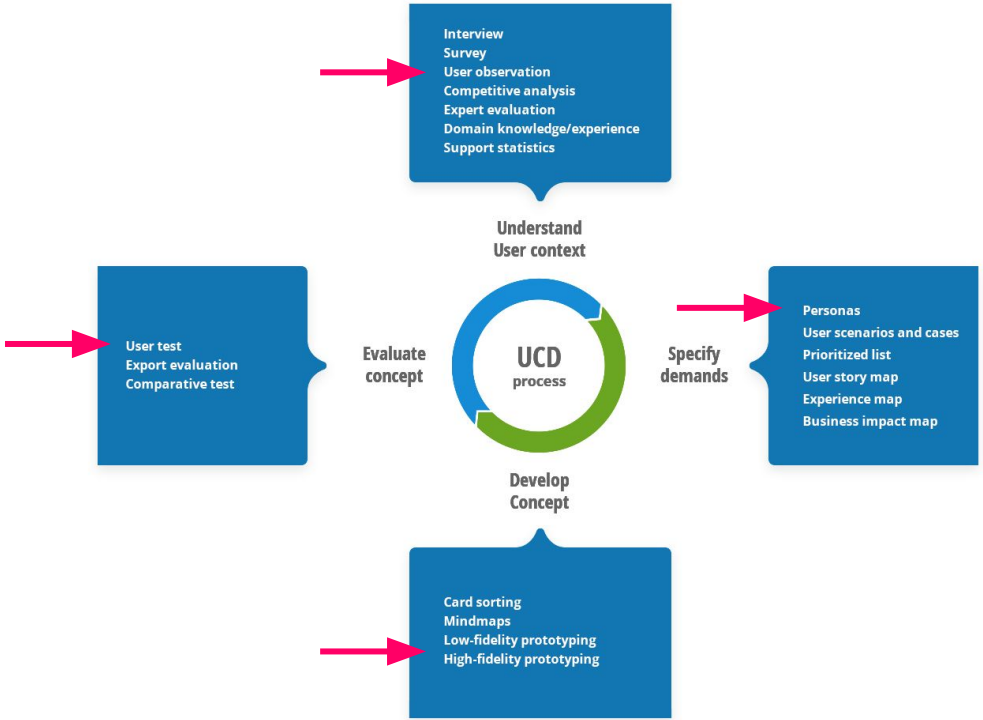
How to do user research and work with our User-Centred Design process?


Aesthetics
UX guidelines & libraries
Interface dynamics
UCD process
Usability
Interface standards
UX guidelines & libraries

Visma's User-Centred Design process



Useful methods for you



Understand user context

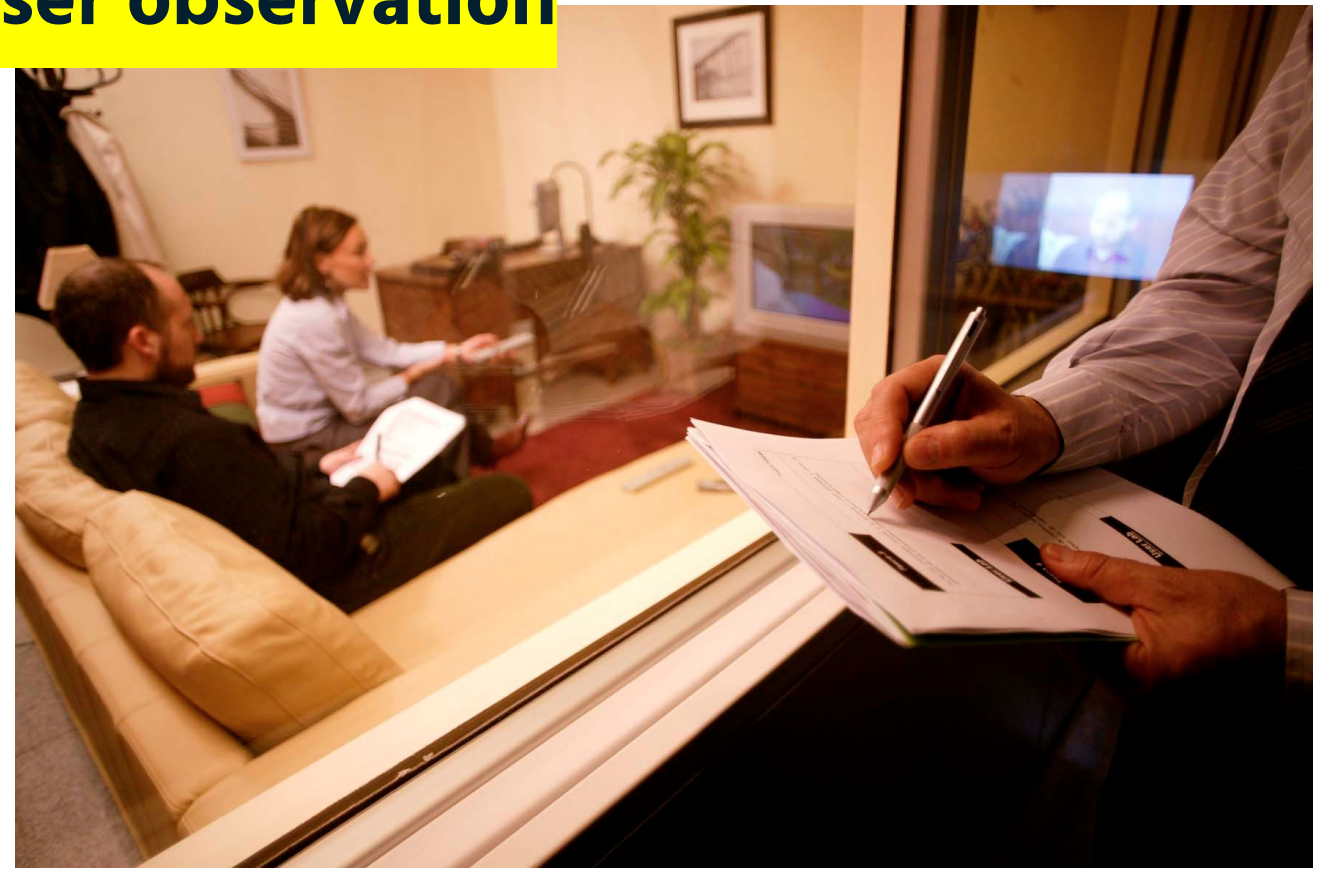
- The most important step in the process
- Who are the users?
- In what environment/situation do they use the product?

- What goals shall the product help the user to achieve?

- What demands must the application fulfil to satisfy the user?

Understand user context – ideal world vs. hairy reality

User observation

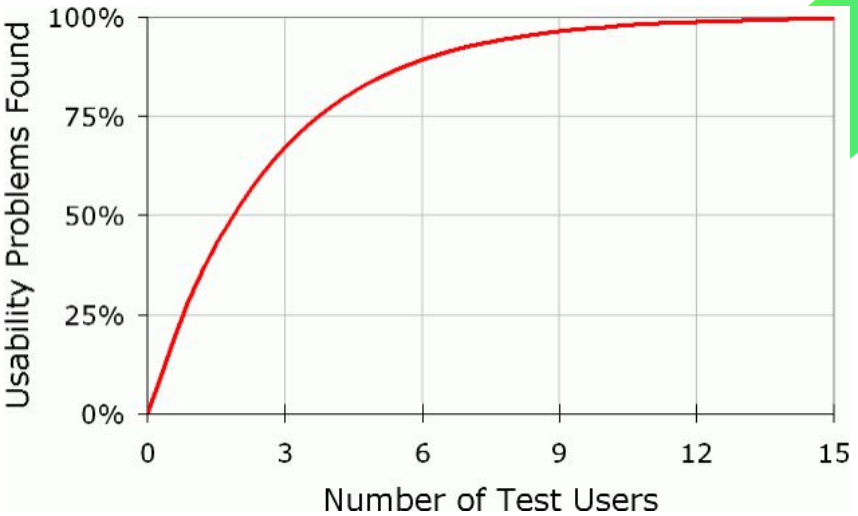


User observation

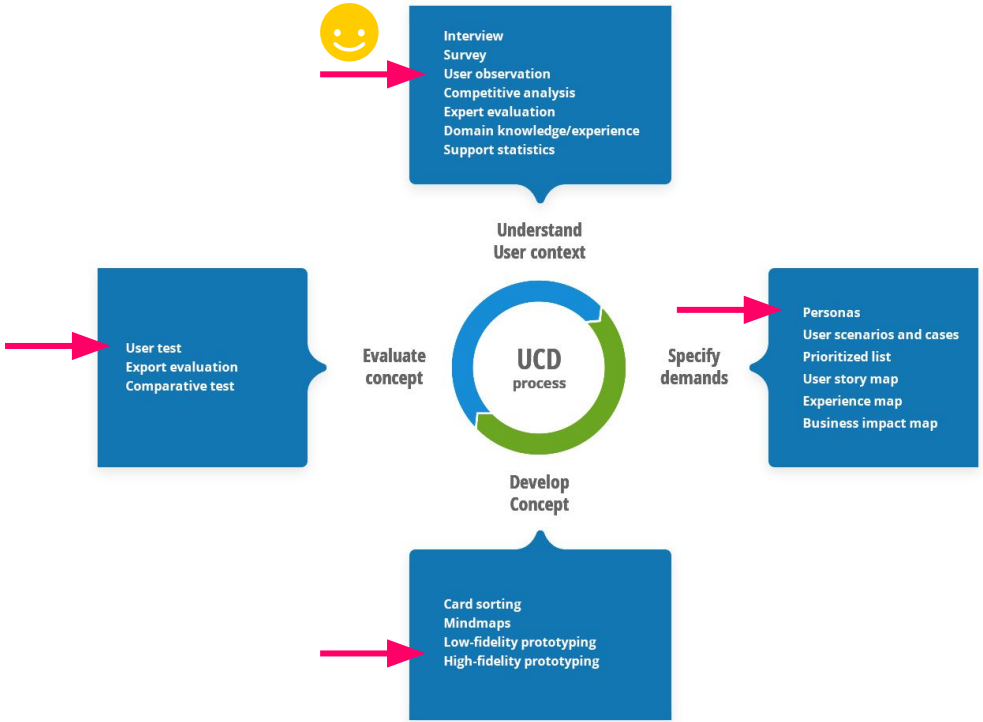
- Observe the user performing natural work tasks
- Study the user behaviour, environment, needs
- Observe what is actually done – not what is said!

User observation

- Classic Jacob Nielsen Research
- Identifies 60-80% of usability problems with 3-5 users



Useful methods for you



Persona development

The creation of Pip

The story of
Philip Pirrip

Let me tell you the story of Philip Pirrip, a brilliant young professional writer and his journey to work excellence

The story of Pip

The first memory Pip has is about his grandparent's attic - the old smell of used books, the creaking wooden boards and the dust particles that danced silently in the sun rays like a game of hiding and seek.

He spent a long time there with his small furry friend - Arnold. The cat was not impressed by the large amounts of book, maps, and chests available, he was there usually to take a nap in the daytime, and he loved the silence. **For Pip everything changed.**

The love of books and endless hours of reading his grandfather's collections led Pip to the University of Charles Dickens where legendary professors took his talent further.

Now he just rented a small loft in the downtown area of his town and recently launched his first official blog post on his new website.

"**I have great expectations from this**," he said to Arnold. The cat heard him but was not impressed and continued his nap.

The story of Pip

After six months he's already a successful blogger, a freelance writer working with 7 different publications at a time and writing the 8th chapter of the first draft of his first book "**Great Expectations.**"

Soon he quickly realized that all this work needs to be organised somehow. "My office is a mess" he said in slight desperation one Saturday morning. "**My work is so chaotic... I need a simple way to manage it all** or else I'll go insane!"

What should I do?

“I want something that is easy to use and quick. I would much rather spend my time creating valuable work...”

Philip Pirrip | **Persona for Cloud Storage**

Creative & social archetype

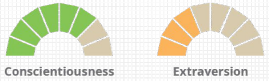
Passionate and high skilled writer

"I will always strive to do a good job. Part of the reason why I'm working for myself is that my practical skills have always been valued. Being creative and hard working has earned the respect of my classmates and teachers over the years."

About Pip

- *July 22, 1994 (22 years old)*
- *Kronprinsens gate 146, Oslo*
- *12 years writing experience, brief summer job as a waiter at local coffee shop*
- *7 161 EUR yearly salary*
- *Does not have a girlfriend*
- *Loves mountain hiking and skiing in the weekends, meeting up with friends as often as possible.*

Personality

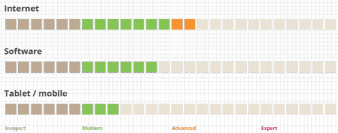


Personality type **ESFP**

ESFPs are cooperative, "here and now" people-persons that enjoy excitement and love new adventures. Because of their highly social nature, they are especially lively when they are the center of attention and hate being alone. ESFPs have a practical side that allows them to finish work efficiently and are often good problem solvers.

Technology

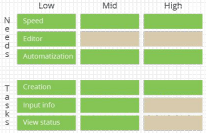
Expertise level **Medium**



Experience Goals

1. Dont waste time
2. Speed
3. High degree of control
4. Confident and secure
5. Helpful

Goals



Must Do

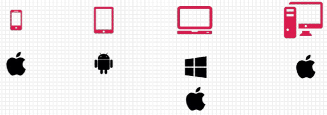
Hide unnecessary processes - Not interested in having the 'control'.
Shorten the time spent in the application - Wants to get things moving fast.
Clear and transparent - Wants to quickly understand what's going on.
Customizable - Wants to make preparations and decisions right away.

Must Never

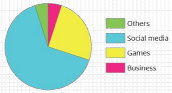
Slow application - Waiting makes her unefficient
Error messages / limitations - Creates a barrier for the end goal
Show all actions at the same time - Feels overwhelmed with decisions, would rather follow the given flow.

Referent & Influences

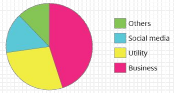
Devices and platforms



App



Software



Relationship with Invoice Maker Pro

L2 **Level 2**
Know but does not actively use our products / services.

Seek and Value

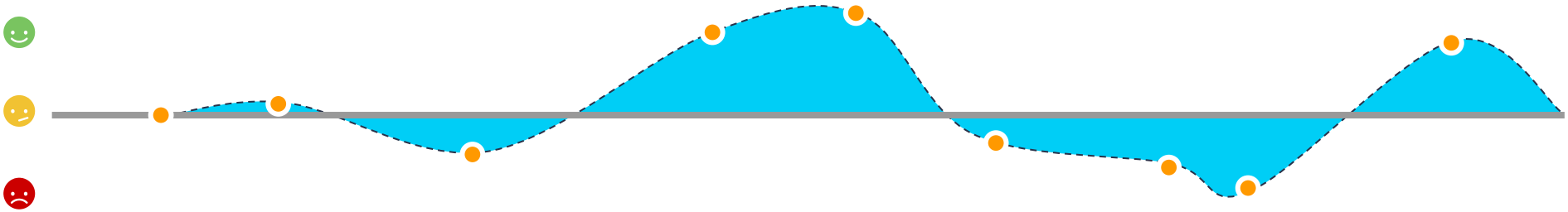
1. Fast
2. High degree of control
3. Guide or help as much as possible
4. Customizable

Brand is

"One of the biggest software solutions in my country."

Product is

"I think I heard someone talking about it..."



Internal decision

Found via Google

Submits online form

First use / on-boarding

Upload document

Manual entries

Document retrieval

Marketing

Possible lead

Pre engagement

Commercial

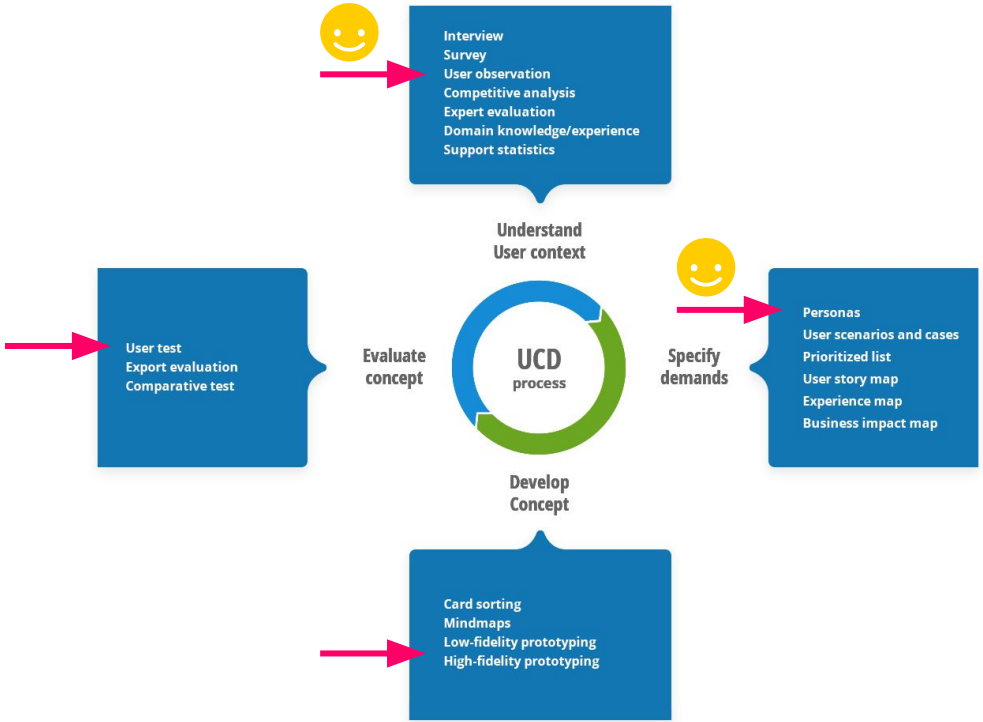
Consultation

Initiation

Coordination

Completion

Useful methods for you



Prototyping

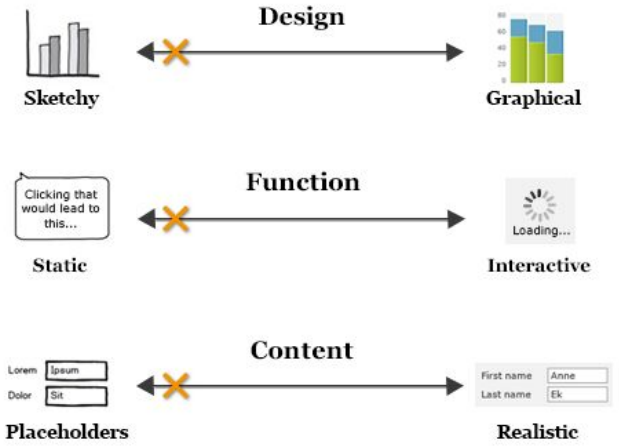
Let’s develop a concept

Develop concept

- Creative step
- Explore new ideas
- Fun and exciting
- Two types of prototyping
    - Low-fidelity **(always start here)**
    - High-fidelity

Low and High fidelity prototypes

- Fidelity : how close the prototype resembles a final solution
- Three dimensions:



What is low-fidelity prototyping?

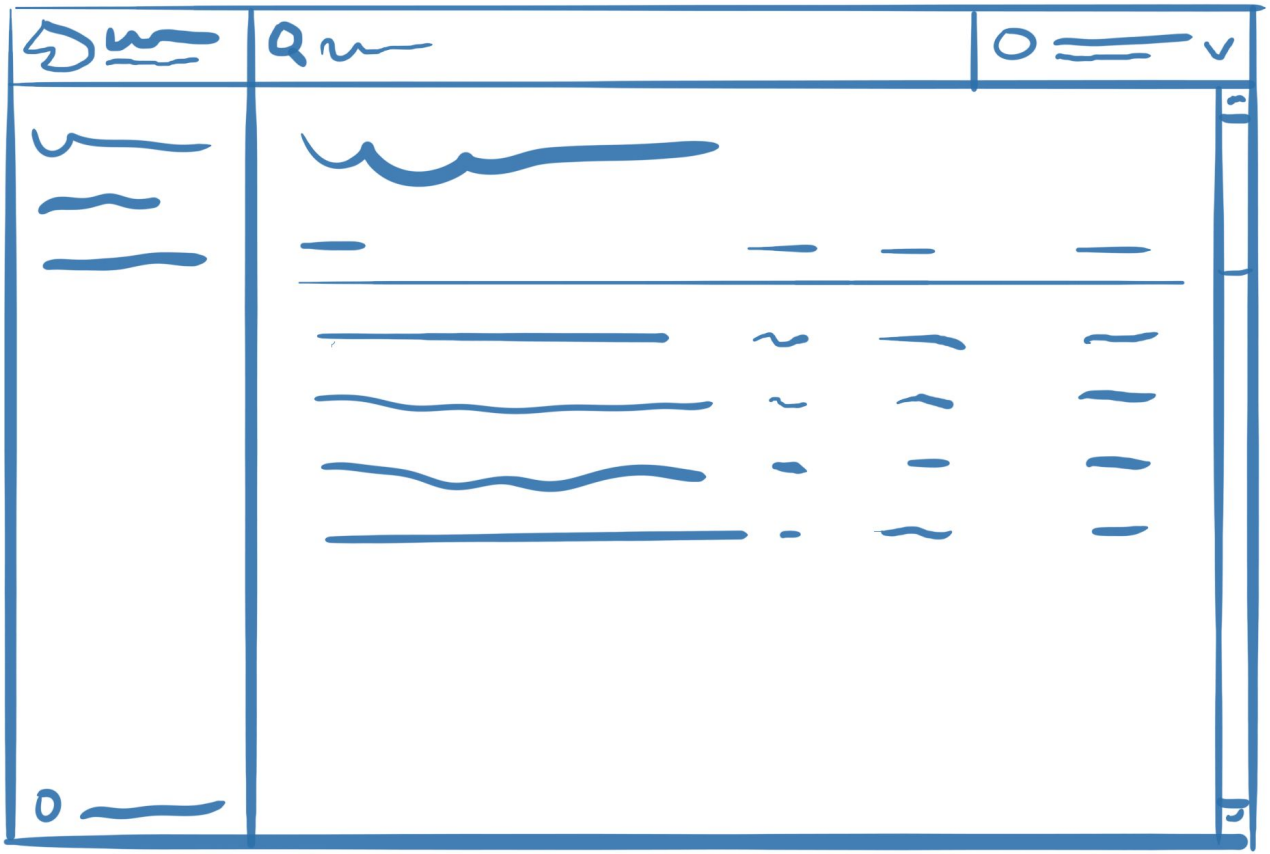
- Fast
- Easy
- Cheap

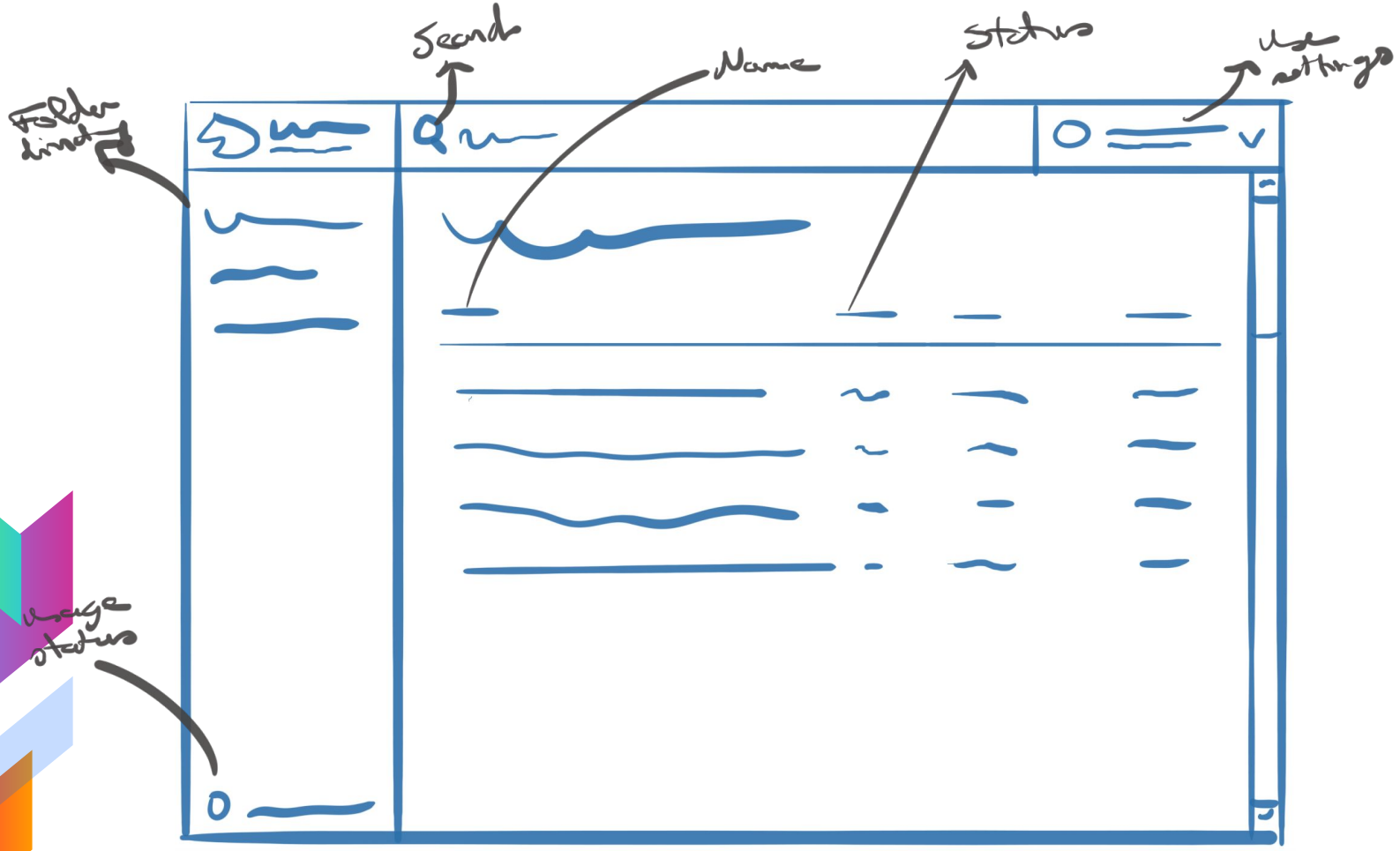
- Sketchy
- Also called wireframes or mockups

How many of you usually draw on paper when a new concept is specified/implemented?

How should we low-fidelity prototype?

- Always start on paper – paper prototyping, it will save you time in the long run
- Focus on structure and function, no details
- Do not apply any design
- Create multiple concepts in the beginning to explore ideas
- Not supposed to be pretty!

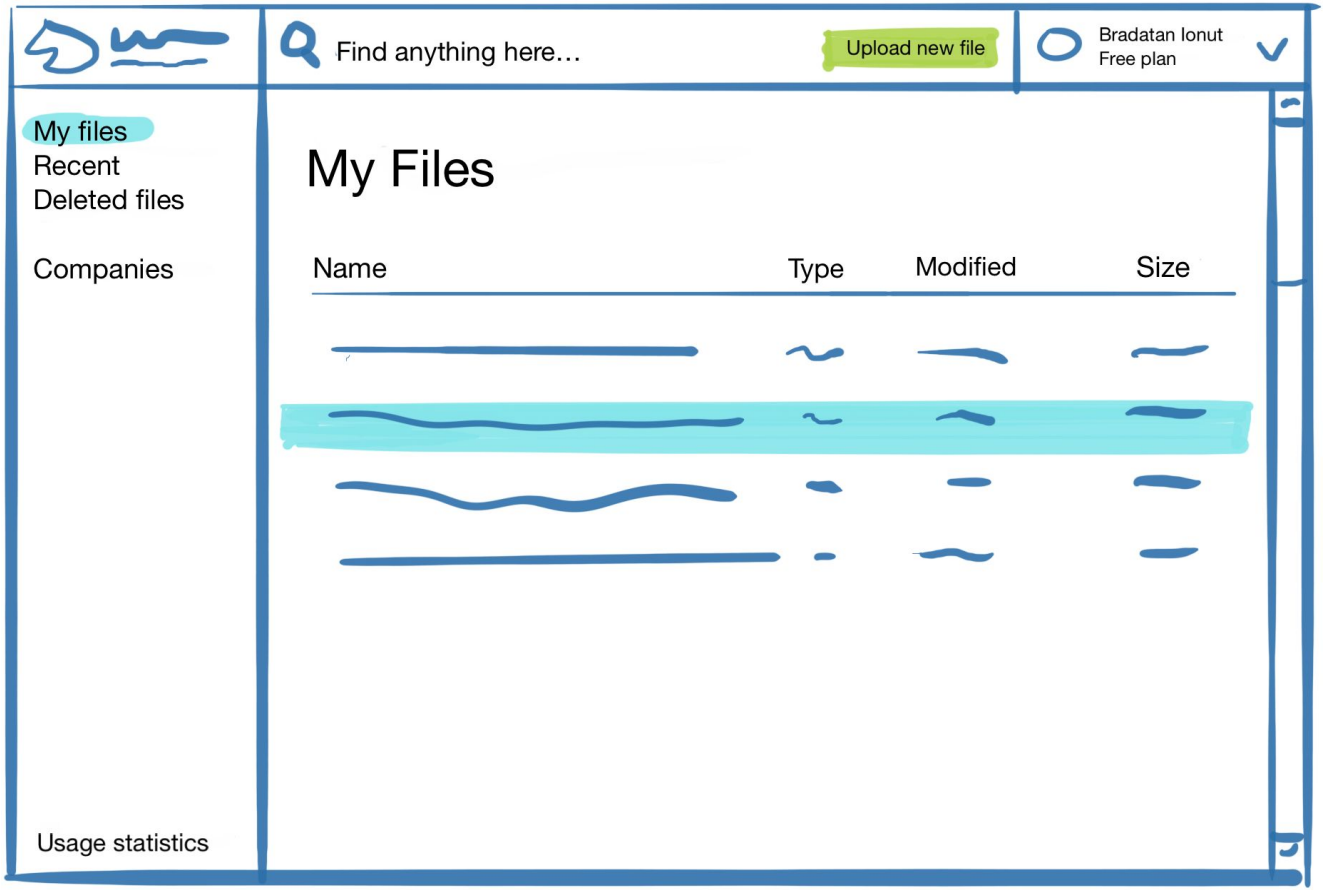


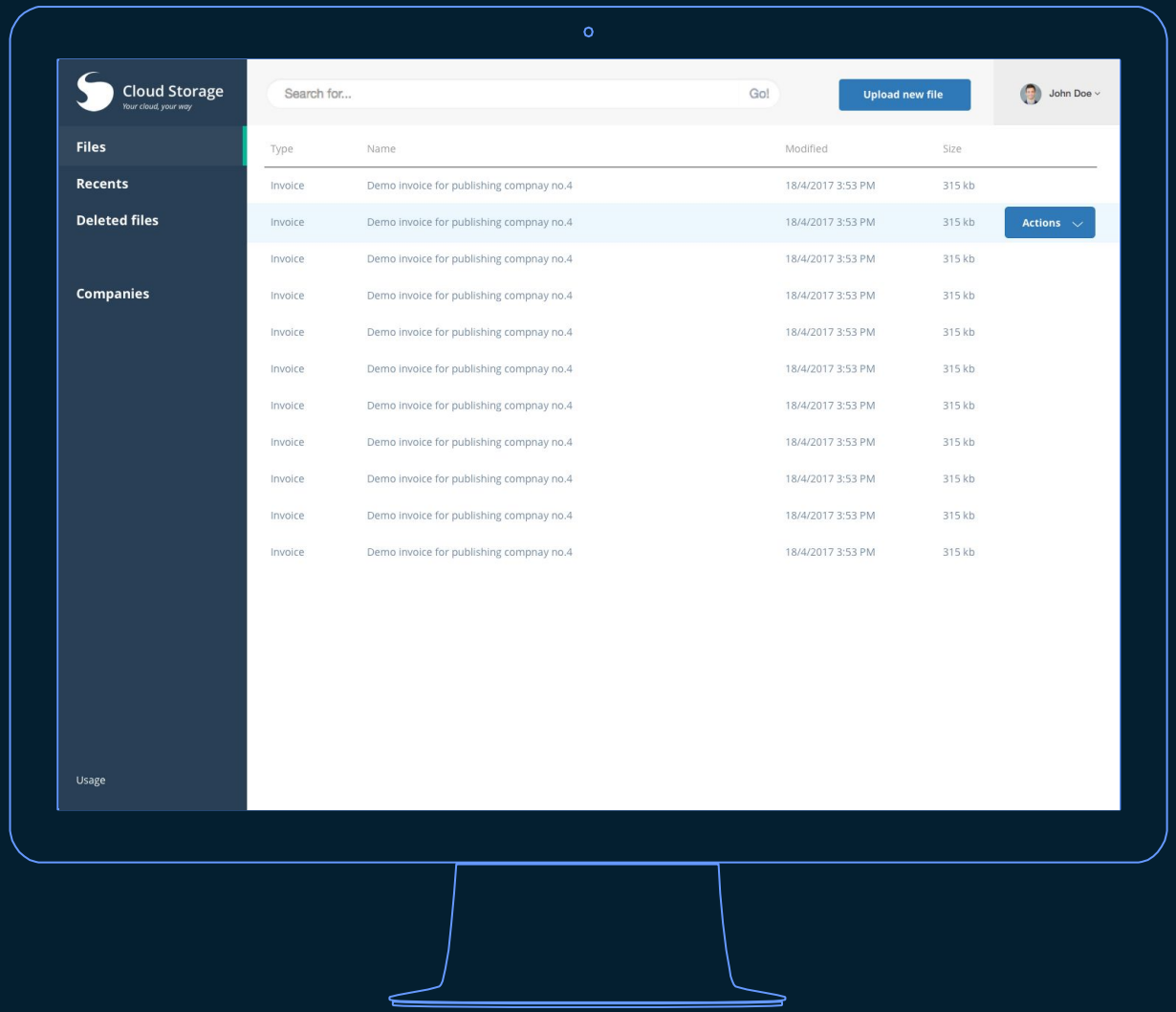

Search
Name
Status
Folder tree
Usage status

First evaluations of your sketches

- Yourself and others in your project while sketching
- Discuss and look at the sketches with a critical eye
- See it from your users perspective, will they understand?

- Pick out the good ideas from the different concepts and combine
- **Iterate!** Refine, elaborate and narrow down number of concepts


Find anything here...
Upload new file
Bradatan Ionut
Free plan
My files
Recent
Deleted files
Companies
Usage statistics
My Files
Name
Type
Modified
Size


Cloud Storage
Your cloud, your way
Search for...
Go!
Upload new file
John Doe
Files
Recents
Deleted files
Companies
Usage
Type
Name
Modified
Size
Invoice
Demo invoice for publishing compnay no.4
18/4/2017 3:53 PM
315 kb
Actions

Later evaluations of your sketches

- When having a bit more elaborated sketches we can and shall test on users
- Making “interactive” paper prototypes by having them in sequence to show a flow makes it possible to test a lot

Microspective Multimedia
Raymond Kingston
81 Benton Street
Manchester, CT 06040

860 643 4423
ray@microspective.net
microspective.net

hello. this is your invoice.

Company Name
57 Winter Street
Brookfield MA 02382

INVOICE DATE	PLEASE PAY	DUE DATE
Nov 05, 2009	$ 772.50	Nov 20, 2009

DESCRIPTION	HRS / QTY	RATE / PRICE	SUBTOTAL
Nov 05, 2009 · Website Updates	03:00	$ 40	$ 120.00
Nov 05, 2009 · Website Updates (Home page)	05:00	$ 40	$ 200.00
Nov 05, 2009 · Flyer (2 versions)	06:30	$ 40	$ 260.00
Nov 05, 2009 · Business Card Template	04:20	$ 25	$ 112.50
Nov 05, 2009 · Custom Error 404 Page	02:00	$ 40	$ 80.00

Subtotal	$ 772.50
Total	**$ 772.50**
Previous Balance	$ 0.00
Total Due	**$ 772.50**

PAYMENT We accept cash, money orders, checks *(payable to **Microspective**)*, PayPal, and credit cards *(accepted online at **microspective.net/payment**)*.

PACKAGES If you purchased a package, this invoice may not reflect your total balance due. Please review the terms indicated in your project design proposal.

TRAVEL The first 30 minutes of round-trip travel is free; $5 every additional 15 minutes.

QUESTIONS If you have any questions about your bill, please feel free to contact us at your convenience. We will reply as soon as we get your message.

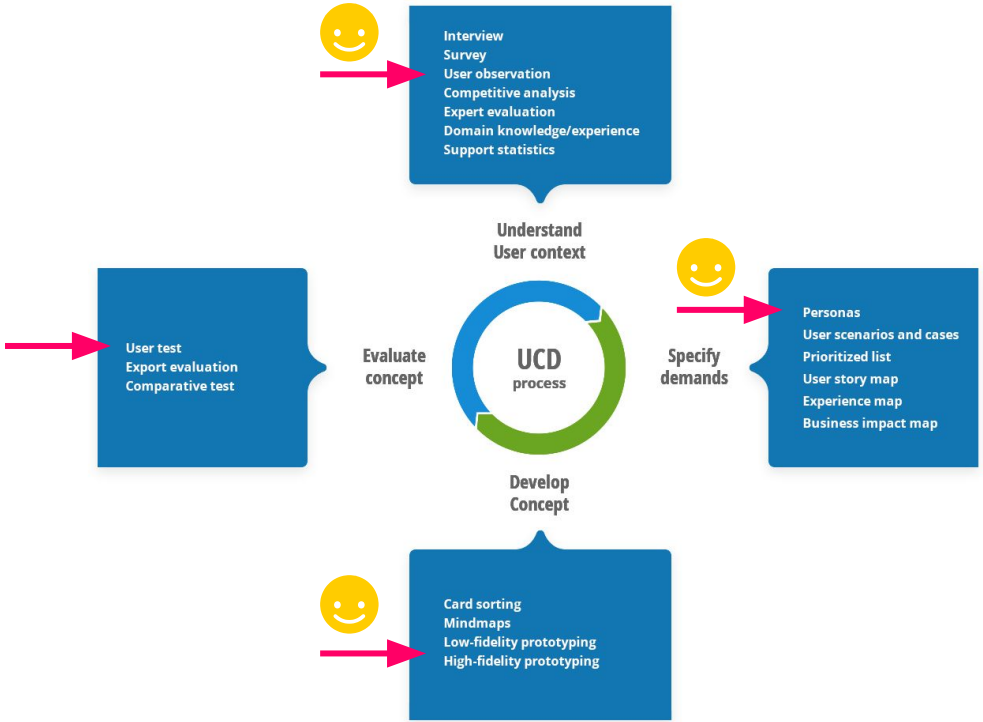
Thank you for choosing Microspective Multimedia.

Let’s do this!

- ⏳ 20 min.
- Use paper and pencil (groups of two or three)
- Present your findings

Present your findings

Useful methods for you



User testing

Let’s test a concept

Why should you user test?

- 3-5 users can find 60-80% of the usability problems
- Important method for evaluating concepts
- You cannot sit and wonder, ask users and see what they do
- Easy, effective and cheap

That is user testing

- Evaluation by the users of course
- The goal is to discover errors and areas of improvement
- What can you test?
    - Concepts on paper
    - More refined prototypes e.g. in Balsamiq
    - Implementations, beta versions
    - Live product, previous version

- Most common and easy is to perform face to face testing

That is user testing

- Decide what to test, e.g. paper prototypes
- Plan tasks based on most important problems/workflows, you cannot test everything, focus on the right things

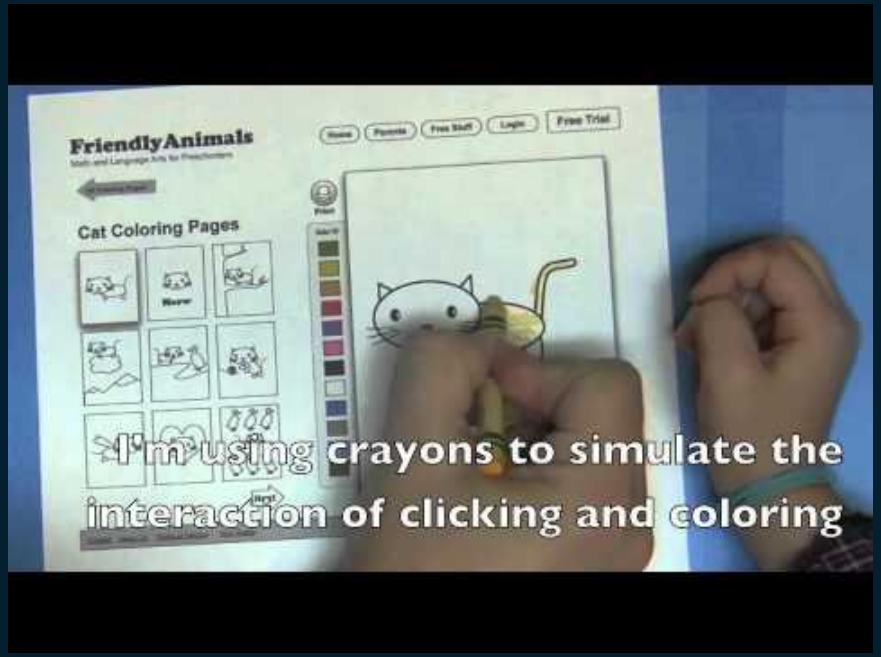
- Create a test guide
- Select a user group to test: Find different type of users, to get a good representation of the different needs.

- Contact the users
- Send out background questions in advance
- Visit them at their office to see context

Conduct a user test

- Welcome and introduce the user to the test
- Provide one task at the time
- Use think-aloud technique as with observations
- Do not help the user during the test, treat it as you are not in the room
- Document with notes and record the screen

Example of a face to face user testing on paper prototypes



Tips for user testing

- Find different type of users.
- Be realistic about the amount of tests you can do
- Put the user at ease by explaining the test's purpose, that you are not testing them but the product

- Collect general impressions at the end
- Bring something with you as a nice gesture
- If you cannot get hold of real users, test on colleagues outside the team, friends, family – better than no test at all

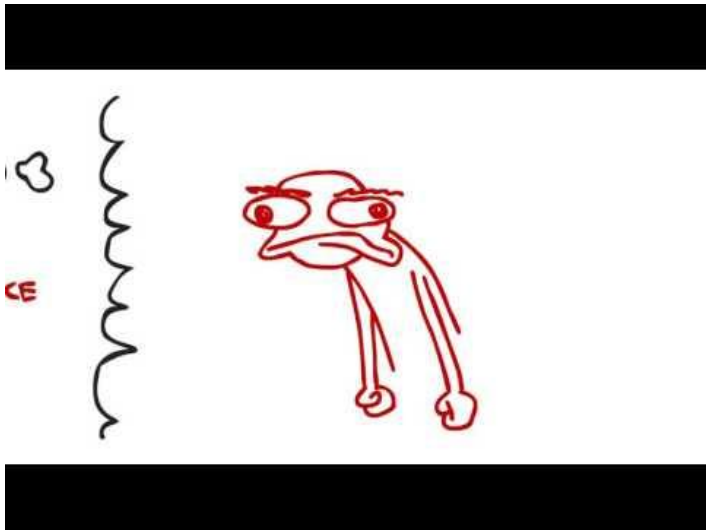
Common mistakes while user testing

- Not giving the user enough time to complete the task
- Telling the user what to do
- Asking leading questions
  - I.e. “Choose Documents in the menu” vs. “Go to the page where you can create a new file.”
- Not asking questions to gather more info during the test
- Ask questions like “Was that what you were expecting to happen?” and “What is it you are looking for?”

User testing time!

At least 3 users, refine your paper prototype if you feel the need need

30 min.



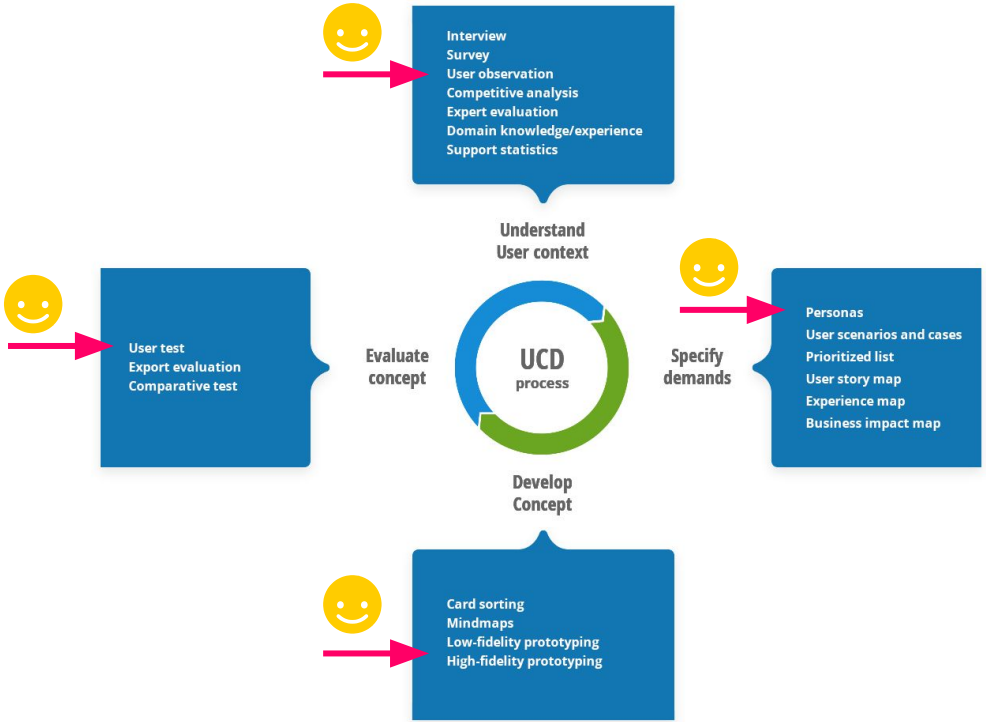
Get back inside in 30 minutes

Refine your sketches if you prefer

Reflections after the user test

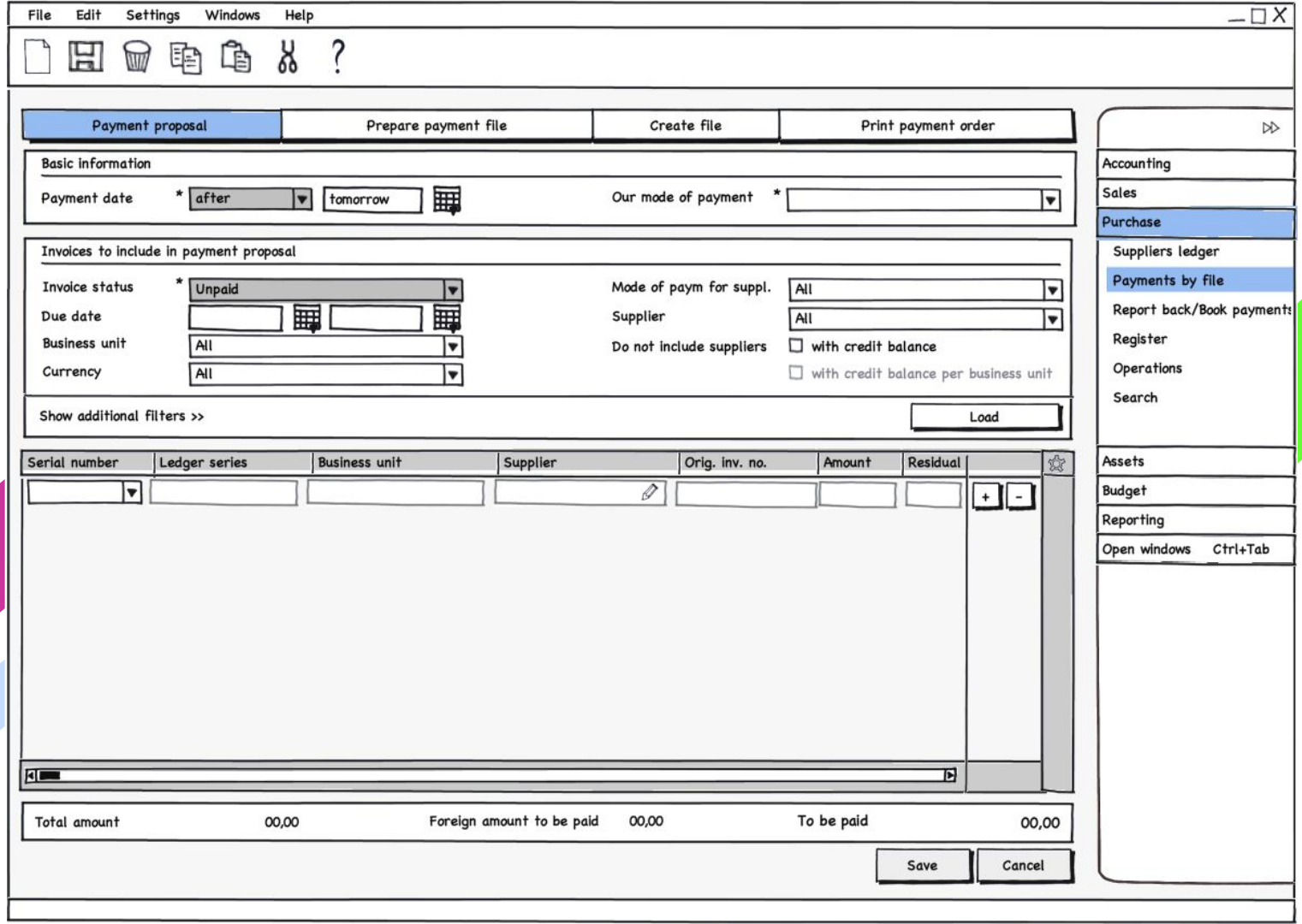
How did it go?

Useful methods for you



What tools can be used for low-fidelity prototyping?

- Paper
- Powerpoint
- Photoshop
- Visio
- **Balsamiq**
- Use the tool you are comfortable with


File
Edit
Settings
Windows
Help
Payment proposal
Prepare payment file
Create file
Print payment order
Basic information
Payment date
after
tomorrow
Our mode of payment
Invoices to include in payment proposal
Invoice status
Unpaid
Due date
Business unit
All
Currency
All
Mode of paym for suppl.
All
Supplier
All
Do not include suppliers
with credit balance
with credit balance per business unit
Show additional filters >>
Load
Serial number
Ledger series
Business unit
Supplier
Orig. inv. no.
Amount
Residual
Total amount
00,00
Foreign amount to be paid
00,00
To be paid
00,00
Save
Cancel
Accounting
Sales
Purchase
Suppliers ledger
Payments by file
Report back/Book payments
Register
Operations
Search
Assets
Budget
Reporting
Open windows
Ctrl+Tab

Balsamiq Mockups (www.balsamiq.com)

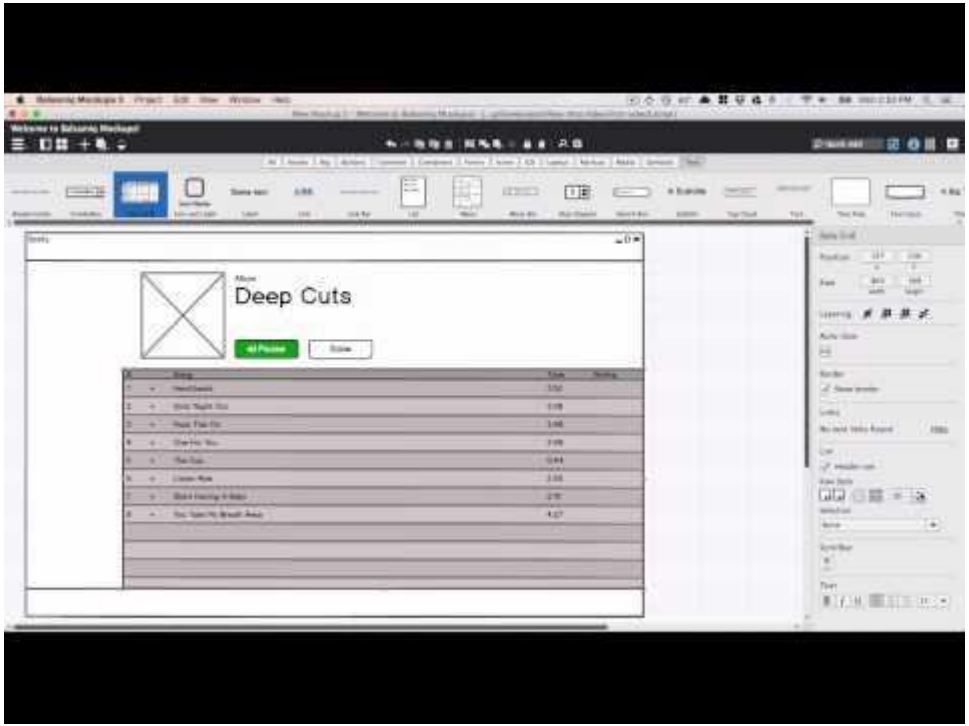
➔ Platforms: Windows, Mac, Linux, Web (cloud)

➔ Plugins
- Google Drive
- JIRA
- Confluence
- etc.

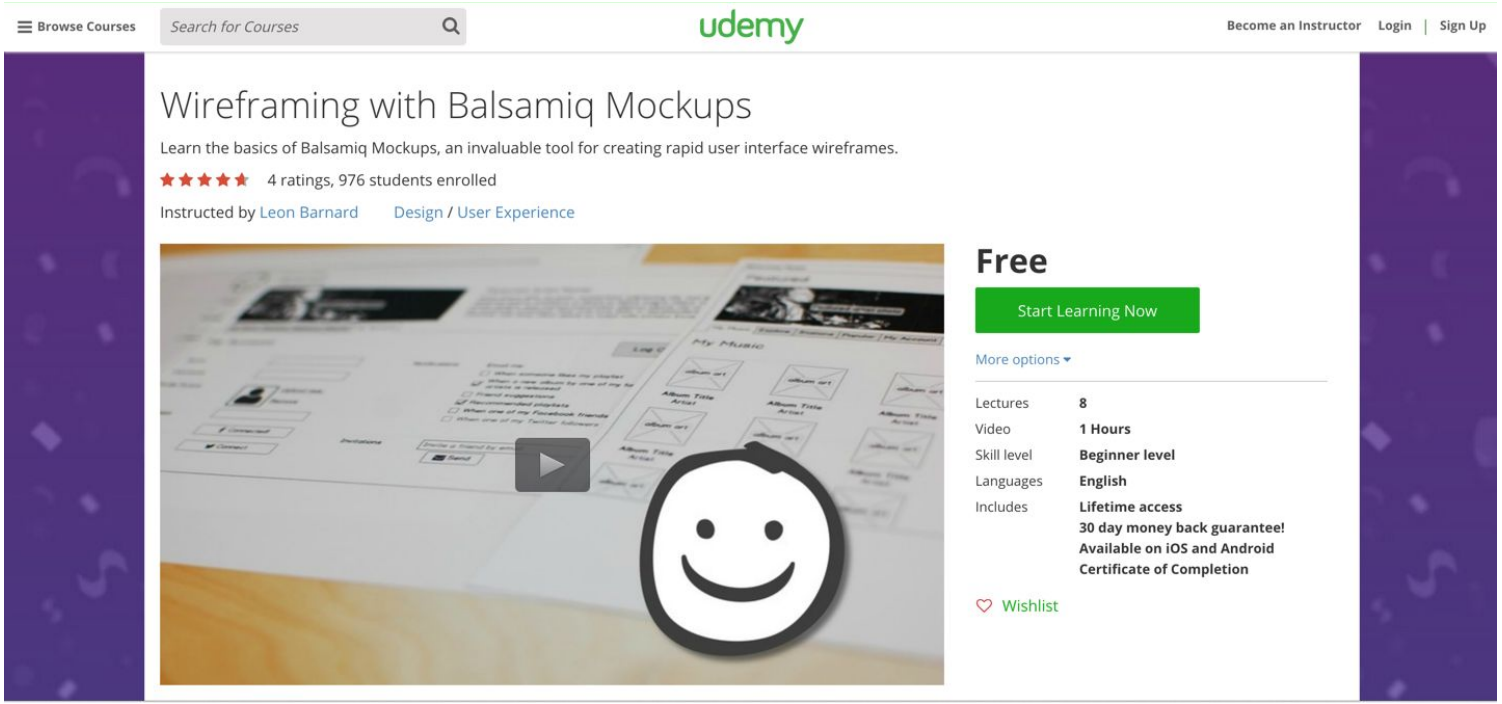
Balsamiq Mockups (www.balsamiq.com)

- ➔ Easy and fast to learn for everybody in your team
- ➔ Built-in user components and icons
- ➔ Exports to
  - PNG (images)
  - PDF with links (shareable, clickable mockups!)
  - Balsamiq code
- ➔ Design patterns and controls for reuse by others
  - https://mockupstogo.mybalsamiq.com/projects

Quick Balsamiq demo



Free 1 hour Balsamiq course!



Balsamiq Prototyping

Group exercise

Exercise: Balsamiq prototyping

- ➔ Same groups as before
- ➔ Create Balsamiq prototypes of your paper sketches
- ➔ You may make smaller improvements on the concept based on relevant feedback in your user tests

- ➔ Discuss which feedback was most important
- ➔ Do not change everything just because one user has made a comment, discuss pros/cons and what feels most logical
- ➔ Make everyone work in Balsamiq, split up the screens

- ➔ Make the prototype clickable if you have time

45 min.

How was it to use Balsamiq?

See each groups sketches and explain your process

Evaluate concept

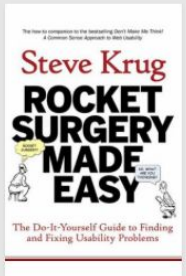
- Evaluate against demands
- Find areas of improvement
- **Iterate!**

That’s a wrap!

Revisited goals

4 awesome hours

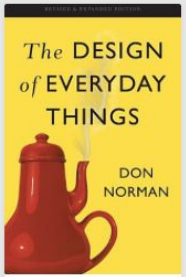
1	Introduction to UX
2	How does Visma UX
3	User research & user centered design
4	Paper prototyping
5	User interviews



Rocket Surgery Mi

[Luke_Wroblewski]

Don't Make Me Th
Steve Krug

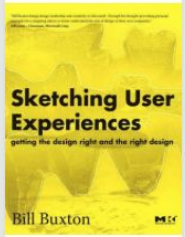


The Design of Eve
Norman, Don

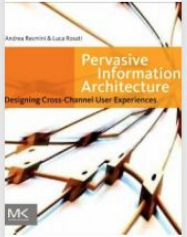
Mobile First
Luke Wroblewski

Why We Fail: Learı
Victor Lombardi

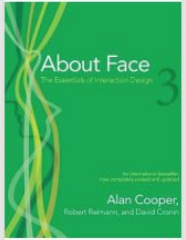
Search Patterns: D
Peter Morville



By Bill Buxton Ske
Bill Buxton

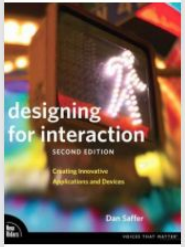


Pervasive Informa
Andrea Resmini

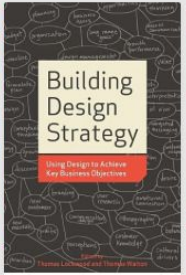


About Face 3: The
Alan Cooper

Remote Research
Tony Tulathimutte

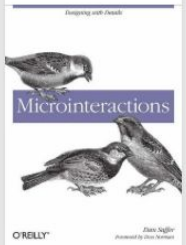


Designing for Inter
Dan Saffer



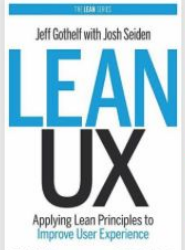
Building Design St
Thomas Lockwood, Th

Interviewing Users
Steve Portigal



Microinteractions
Dan Saffer

Smashing UX Des
Jesmond Allen and Ja

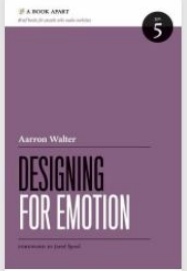


Lean UX: Applying
Jeff Gothelf

The User Experien
Leah Buley

Communicating th

The UX Book Proc



Designing for Emc
Aarron Walter

Tomorrow we'll begin development

@Emil is going to present and he'll lead you in the realm of code

Thank you!

We meet again tomorrow at 9:00

Day 7

Welcome back

April 2017 #VismaAClabs @ligaac_labs

What did we learn yesterday?

Quick overview

Testing

“Testing is an infinite process of comparing the invisible to the ambiguous in order to avoid the unthinkable happening to the anonymous.” - James Bach

Goals with this brief training

Be a good tester!

1	Learn testing terminology
2	Establish a testing plan
3	Automate as much as possible

Quality

Standards

ISO/IEC 9126

- Quality model
- External metrics
- Internal metrics
- Quality in use metrics

ISO/IEC 9241-11

- proposed a framework that describes the usability components

ISO/IEC 25000:2005

- guidelines for Software Quality Requirements and Evaluation (SQuaRE); replaces the two old ISO standards, i.e. ISO-9126 and ISO-14598.

Standard	Description
IEEE 829	A standard for the format of documents used in different stages of software testing.
IEEE 1061	A methodology for establishing quality requirements, identifying, implementing, analyzing, and validating the process, and product of software quality metrics.
IEEE 1059	Guide for Software Verification and Validation Plans.
IEEE 1008	A standard for unit testing.
IEEE 1012	A standard for Software Verification and Validation.
IEEE 1028	A standard for software inspections.
IEEE 1044	A standard for the classification of software anomalies.
IEEE 1044-1	A guide for the classification of software anomalies.
IEEE 830	A guide for developing system requirements specifications.
IEEE 730	A standard for software quality assurance plans.
IEEE 1061	A standard for software quality metrics and methodology.
IEEE 12207	A standard for software life cycle processes and life cycle data.
BS 7925-1	A vocabulary of terms used in software testing.
BS 7925-2	A standard for software component testing.

Myths

- Testing is Too Expensive
- Testing is Time-Consuming
- Only Fully Developed Products are Tested
- Complete Testing is Possible
- A Tested Software is Bug-Free

Myths

- Missed Defects are due to Testers
- Testers are Responsible for Quality of Product
- Test Automation should be used wherever possible to Reduce Time
- Anyone can Test a Software Application
- A Tester's only Task is to Find Bugs

What means testing?

Testing

The process used to identify the correctness, completeness and quality of developed product

Checks - Specifications, Functionality, Performance

Testing ethics

- PUBLIC - shall act consistently with the public interest.
- CLIENT AND EMPLOYER - shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.
- PRODUCT - shall ensure that the deliverables they provide meet the highest professional standards possible.
- JUDGMENT - shall maintain integrity and independence in their professional judgment.

Testing ethics

- MANAGEMENT - shall subscribe to and promote an ethical approach to the management of software testing.
- PROFESSION - shall advance the integrity and reputation of the profession consistent with the public interest.
- COLLEAGUES - shall be fair to and supportive of their colleagues, and promote cooperation with software developers.
- SELF - shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Many terms and their meanings

- Failure - a non fulfillment of a given requirement
- Fault/defect/bug - user experience a failure/problem is an error/mistake made by a person
- Error - human action that produces the incorrect result that produces a fault
- Debugging - localization and correction of faults

Many terms and their meanings

- Testing - detection of failures
- Test case - test conditions, inputs and expectations
- Test scenarios - a combination of dependent test cases
- Test run/suite - execution of one or more test cases

Benefits of testing

“Quality is free, but only to those who are willing to pay heavily for it.”
Tom DeMarco

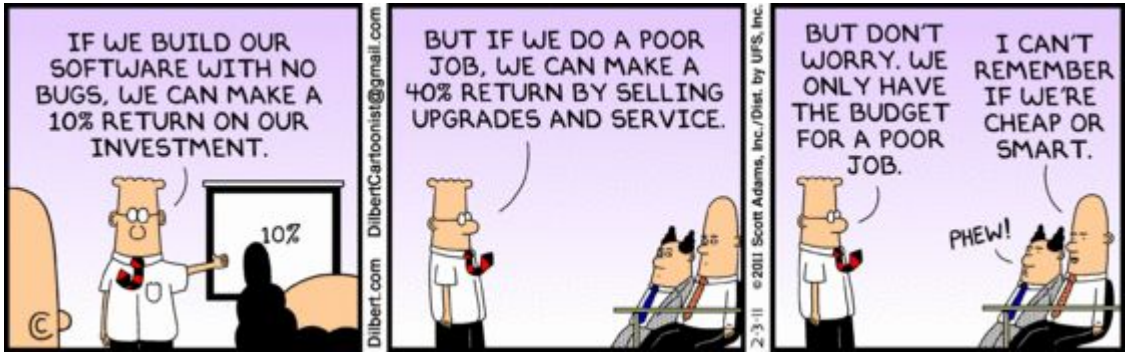
§30-85 errors are made per 1000 lines of source code

§extensively tested software contains 0.5-3 errors per 1000 lines of source code

§testing is postponed, as a consequence: the later an error is discovered, the more it costs to fix it.

§error distribution: 60% design, 40% implementation. 66% of the design errors are not discovered until the software has become operational.

Time for a Joke



Why do we test

ROCKET LAUNCH ERRORS - In 1996, a European Ariane 5 rocket was set to deliver a payload of satellites into Earth orbit, but problems with the software caused the launch rocket to veer off its path a mere 37 seconds after launch. As it started disintegrating, it self-destructed (a security measure). The problem was the result of code reuse from the launch system's predecessor, Ariane 4, which had very different flight conditions from Ariane 5. More than $370 million were lost due to this error.

FLIGHT CRASHES - In 1994 in Scotland, a Chinook helicopter crashed and killed all 29 passengers. While initially the pilot was blamed for the crash, that decision was later overturned since there was evidence that a systems error had been the actual cause.

MEDICAL ERROR - A bug in the code controlling the Therac-25 radiation therapy machine was directly responsible for at least five patient deaths in the 1980s when it administered excessive quantities of X-rays

Why do we test

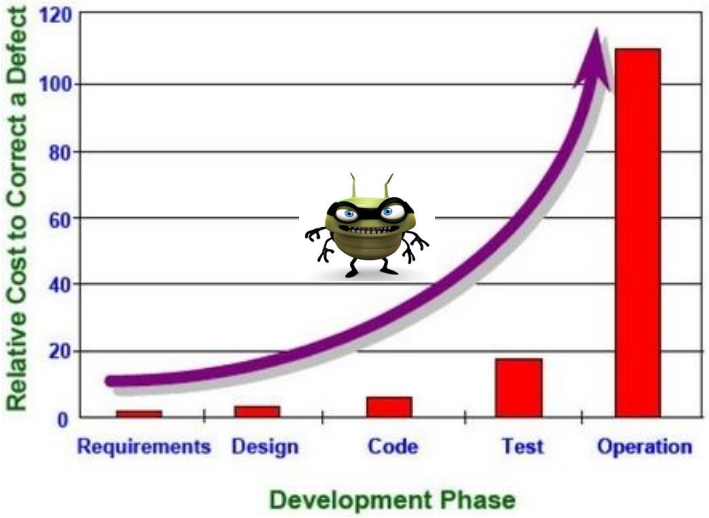
BANK ISSUE - An error in the payment terminal code for Bank of Queensland rendered many devices inoperable for up to a week. The problem was determined to be an incorrect hexadecimal number conversion routine. When the device was to tick over to 2010, it skipped six years to 2016, causing terminals to decline customers' cards as expired

Pentium FDIV bug – When a math professor discovered and publicized a flaw in Intel's popular Pentium processor in 1994, the company's response was to replace chips upon request to users who could prove they were affected. Intel calculated that the error caused by the flaw would happen so rarely that the vast majority of users wouldn't notice. Angry customers demanded a replacement for anyone who asked, and Intel agreed. The episode cost Intel $475 million.

Knight's $440 Million Error – One of the biggest American market makers for stocks struggled to stay afloat after a software bug triggered a $440 million loss in just 30 minutes. The firm's shares lost 75 percent in two days after the faulty software flooded the market with unintended trades. One of Knight's trading algorithms reportedly started pushing erratic trades through on nearly 150 different stocks, sending them into spasms.

GOOGLE ENGINE - In January 2009, Google's search engine erroneously notified users that every web site worldwide was potentially malicious, including its own

Finding a BUG



Test quadrants

Agile Testing Quadrants

Examples

Simulations

Prototypes

Exploratory testing

Security Testing

Component tests

Performance testing

Usability testing

Scenarios

UAT(User Acceptance Testing)

Unit tests

Beta testing

Workflow testing

Alpha testing

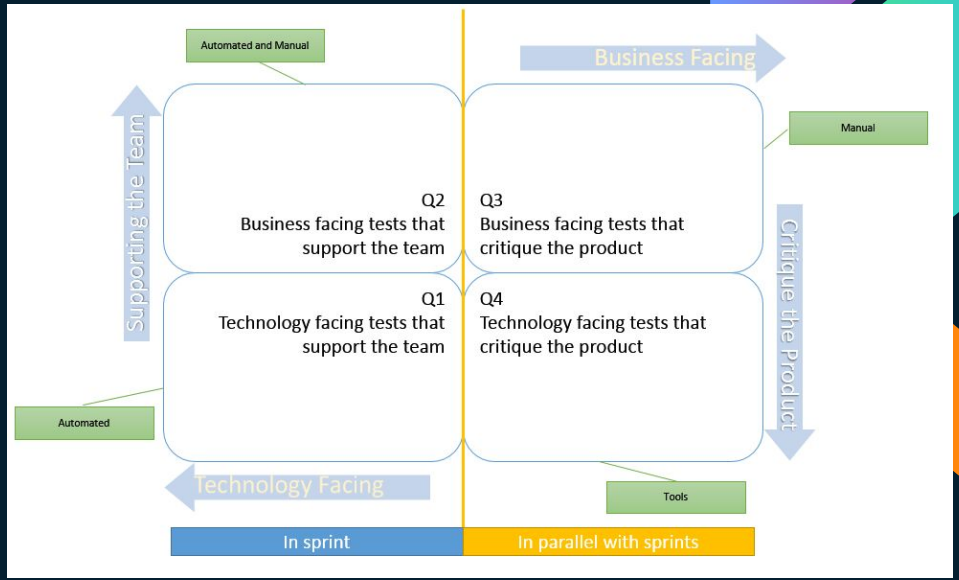
Functional tests

“ility” testing

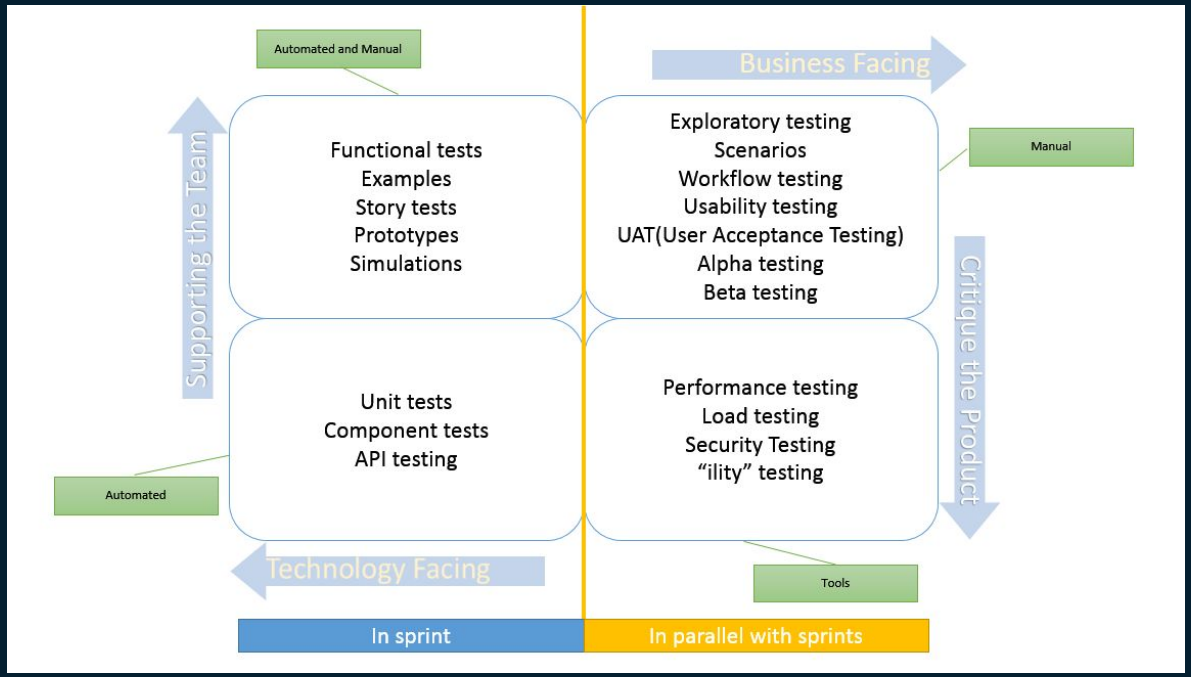
Load testing

API testing

Story tests



Agile Testing Quadrants



Testing methodologies

Black box Testing

Black-box testing is a method of software **testing** that examines the functionality of an application without peering into its internal structures or workings. - Wikipedia

- Equivalence Class - divides the input into classes
- Boundary Value Analysis - validated against both the valid boundaries and invalid boundaries.
- Domain Tests - select domain specific test cases
- Orthogonal Arrays - statistical approach to pairwise interactions
- Decision Tables - every decision is taken each way, true and false
- State Models - execute valid and invalid state transitions
- Exploratory Testing - experience of testers
- All-pairs testing - combination of all variables

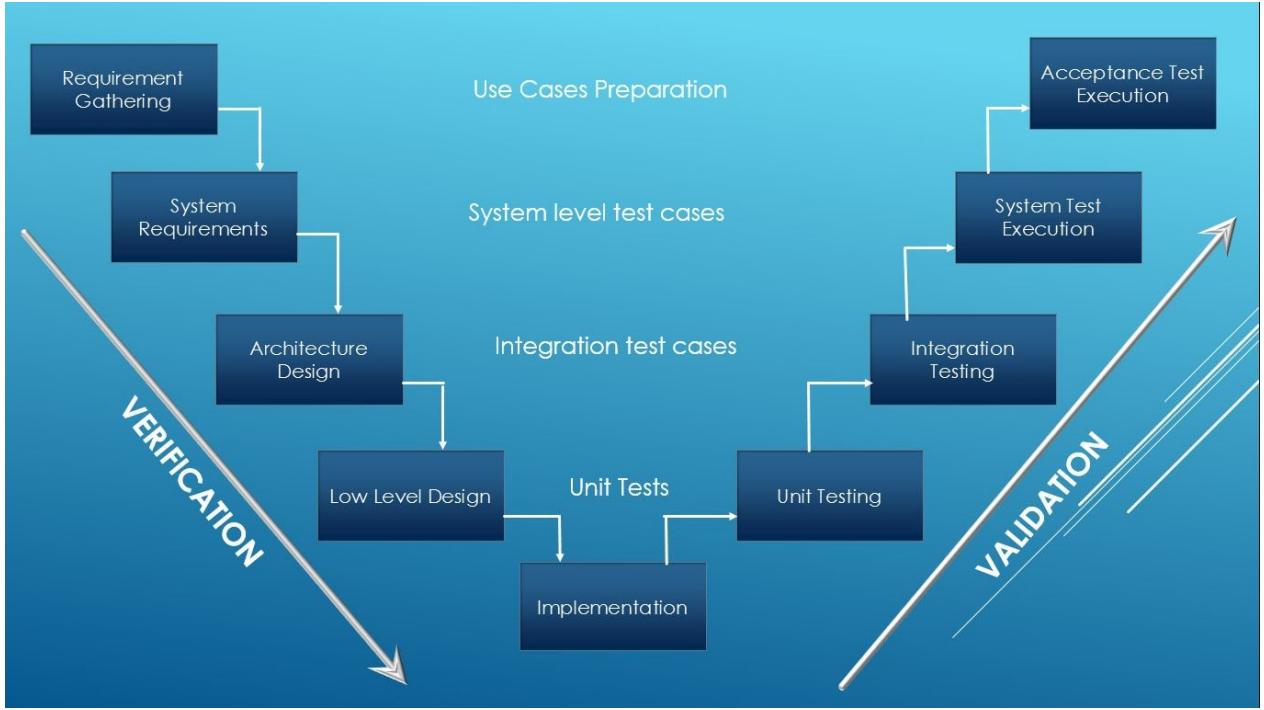
White box Testing

White-box testing is a method of **testing** software that **tests** internal structures or workings of an application, as opposed to its functionality. - Wikipedia

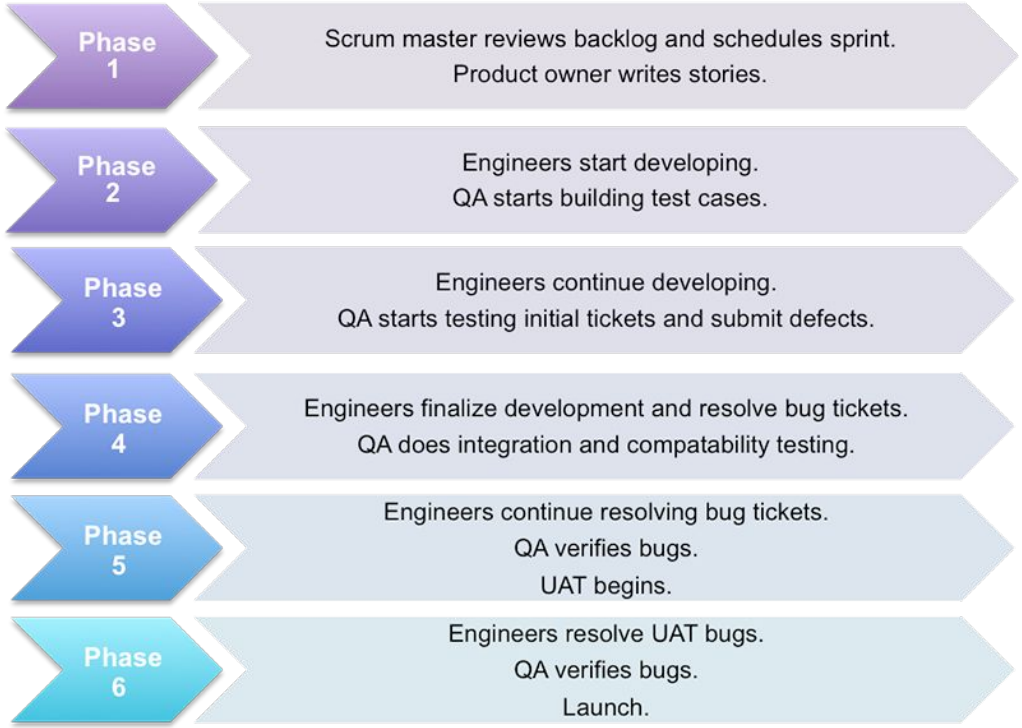
- Statement Coverage - This technique is aimed at exercising all programming statements with minimal tests. Statement Testing = (Number of Statements Exercised / Total Number of Statements) x 100 %
- Branch Coverage - This technique is running a series of tests to ensure that all branches are tested at least once. Branch Testing = (Number of decisions outcomes tested / Total Number of decision Outcomes) x 100 %
- Path Coverage - This technique corresponds to testing all possible paths which means that each statement and branch is covered. Path Coverage = (Number paths exercised / Total Number of paths in the program) x 100 %

Testing Levels

V-Cycle Testing



Testing phases (practical view)



Unit Testing

- Functional - does it do what is supposed to do
- Boundaries - Min, max, special chars, other types,...
- Termination - normal/abnormal termination
- Outputs - what is expected, where are they sent, what if null
- Inputs - what is expected, what if wrong type, what if null
- Interaction - is there interaction with other components, can be affected by their changes

Integration Testing

- Big Bang Integration - all units are linked at once
- Top Down Integration - simulate the behaviour of the lower-level modules
- Bottom Up - lower hierarchy is tested individually and then the upper components
- Mixed Integration - standalone modules are combined and tested as a single entity

System Testing

Alpha - takes place at the developer's site by the internal teams, before release to external customers. This testing is performed without the involvement of the development teams

Beta - takes place at the end users site by the end users to validate the usability, functionality, compatibility, and reliability testing.

Acceptance - technique performed to determine whether or not the software system has met the requirement specifications

Performance

- **Load testing -** behaviour of the system under a specific load.
- **Stress testing -** find the upper limit capacity of the system
- **Soak testing -** the system parameters under continuous expected load to detect memory leaks
- **Spike testing -** increasing the number of users suddenly and measuring the performance

List of testing types

- **Ad-hoc testing** - informal and unstructured and can be performed by any stakeholder
- **Accessibility Testing** - if the contents of the website can be easily accessed by disabled people.
- **Agile Testing** - accommodates agile software development approach and practices
- **API Testing** - similar to unit testing. Each APIs are tested as per API specification
- **Automated testing** - use of testing tools and/or programming to run the test cases
- **Backward Compatibility Testing** - check newer version can work installed over previous version
- **Browser compatibility Testing** - combination of different browsers and operating systems.
- **Compatibility testing** - can be run on different hardware, operating system, bandwidth, etc..
- **Component Testing** - a group of units as code together as a whole

List of testing types

- **Dynamic Testing** - can be done only by executing code or software.
- **End-to-end Testing** - end to end flows (mimicking real life scenarios and usage)
- **Fuzz Testing** - with unexpected or random inputs.
- **GUI (Graphical User Interface) testing** - checking the UI (the length and capacity of the input fields)
- **Glass box Testing** - White box testing.
- **Gorilla Testing** - exercise one functionality thoroughly by multiple people test the same functionality.
- **Happy path testing** - focuses on selective execution that do not exercise for negative conditions.
- **Interface Testing** - interface being testing like GUI or API or CLI.
- **Internationalization Testing** - software that support Internationalization
- **Keyword-driven Testing** - automated testing approach than a type of testing itself.

List of testing types

- **Localization Testing** - software is expected to adapt to a particular locale (font, date time, currency)
- **Negative Testing** - “attitude to break”
- **Non functional testing** - non-functional requirements like performance, usability, localization etc.
- **Pair Testing** - can be done by software testers, developers or Business analysts (BA).
- **Penetration Testing** - how secure software and its environments (Hardware, OS and network) are.
- **Regression Testing** - find defects that got introduced to defect fix(es)
- **Retesting** - carried out by software testers as a part of defect fix verification.
- **Risk based Testing** - functionality to be tested are prioritized as Critical, High, Medium and low.
- **Smoke testing** - check if the new build provided by development team is stable enough
- **Security Testing** - how good are: authorization mechanism, authentication, confidentiality, integrity of the data, availability in an event of an attack.

List of testing types

- **Sanity Testing** - quick evaluation of the software, environment, ... are up & running
- **Scalability Testing** - the ability of the software to scale up with increased users, transactions,...
- **Stability Testing** - how stable is when it is subject to loads at acceptable levels, peak loads,...
- **Static Testing** - code is not executed instead it is reviewed for syntax, commenting, naming,...
- **Stress Testing** - is subjected to peak loads and even to a break point to observe how he behave
- **Soak Testing** - is subjected to load over a significant duration of time (days, weeks).
- **Usability testing** - understand how user friendly the software is.
- **Volume testing** - to find the response of the software with different sizes of the data being received
- **Vulnerability Testing** - involves identifying, exposing the product's vulnerabilities that can be exploited by hackers and other malicious programs likes viruses or worms. Vulnerability Testing is critical for success of a Business.

Tomorrow we'll start testing

Thank you!

We meet again tomorrow at 9:00

Day 8

Welcome back

April 2017 #VismaAClabs @ligaac_labs

What did we learn yesterday?

Quick overview

Testing

“Testing is an infinite process of comparing the invisible to the ambiguous in order to avoid the unthinkable happening to the anonymous.” - James Bach

Goals with this brief training

What did we learn yesterday?

TestPlan

Template

Agile Test Plan

Project Name		Browser:	
Test Plan ID		Version:	
Written By:		Description:	
Tested By:		Tested On:	

Test Type	Feature	Test #	Date	Action	Expected Results	Actual Results	Pass?
							☐
							☐
							☐
							☐
							☐
							☐
							☐
							☐
							☐
							☐
							☐
							☐
							☐
							☐
							☐

Test Automation

Automated Testing

End-to-end testing

GUI testing

Happy path testing

Interface testing

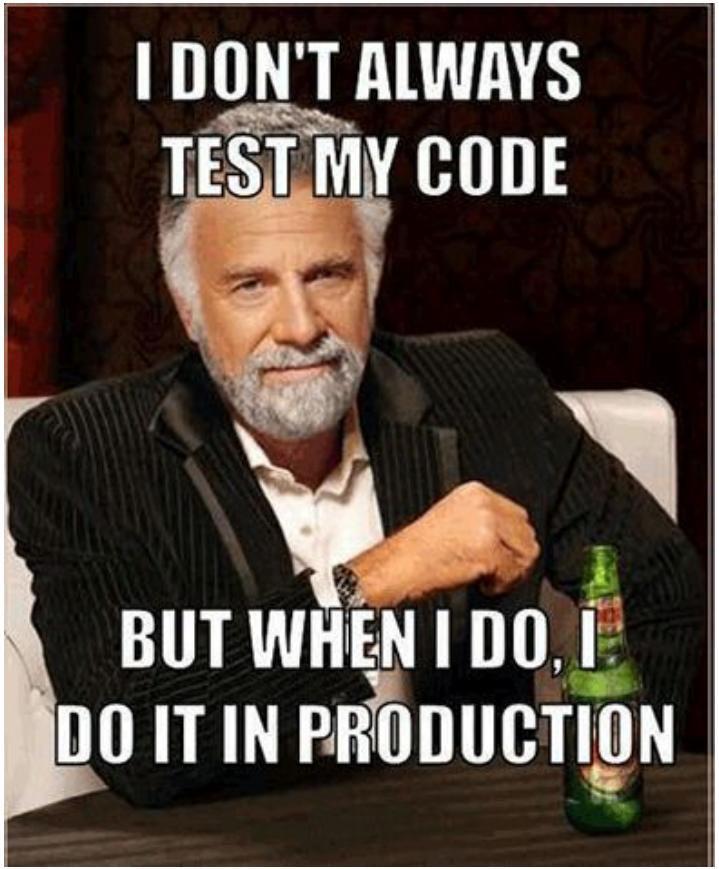
Smoke testing

Retesting

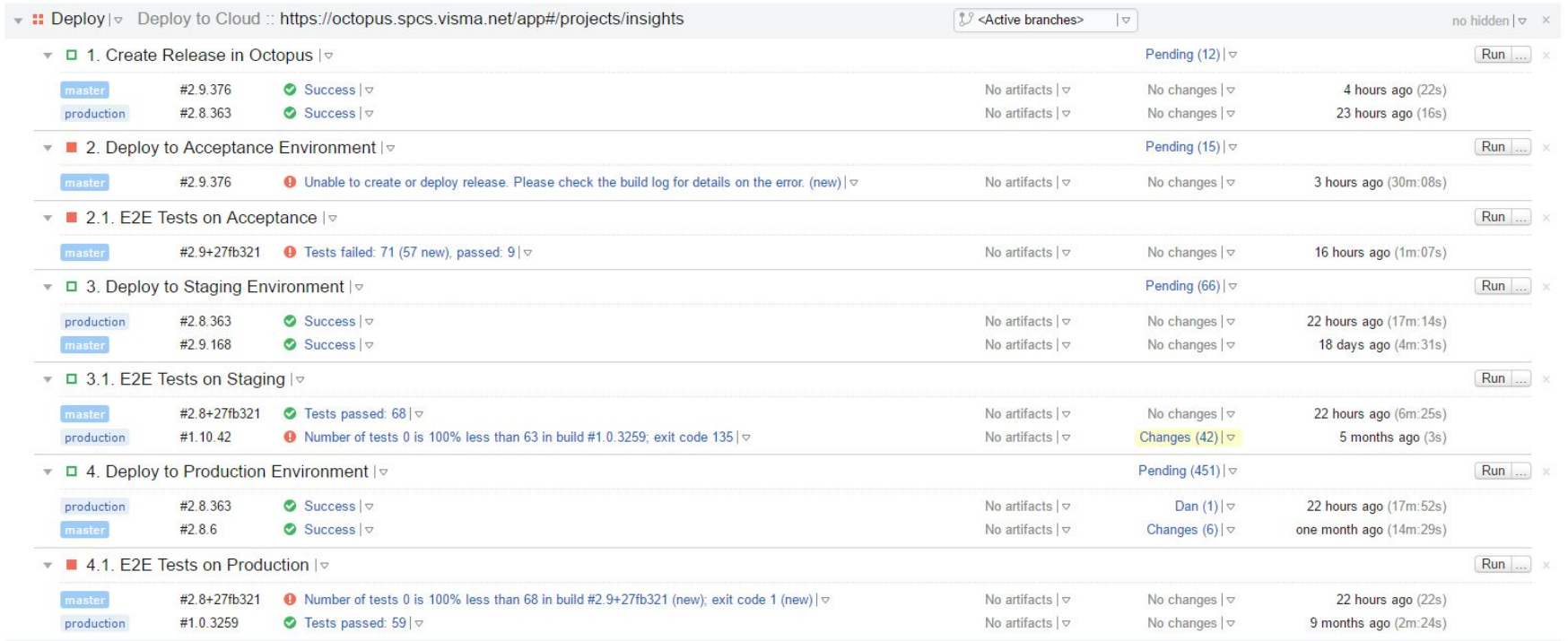
Negative testing

Localization testing

REGRESSION Testing



Prevent issues from occurring in Production

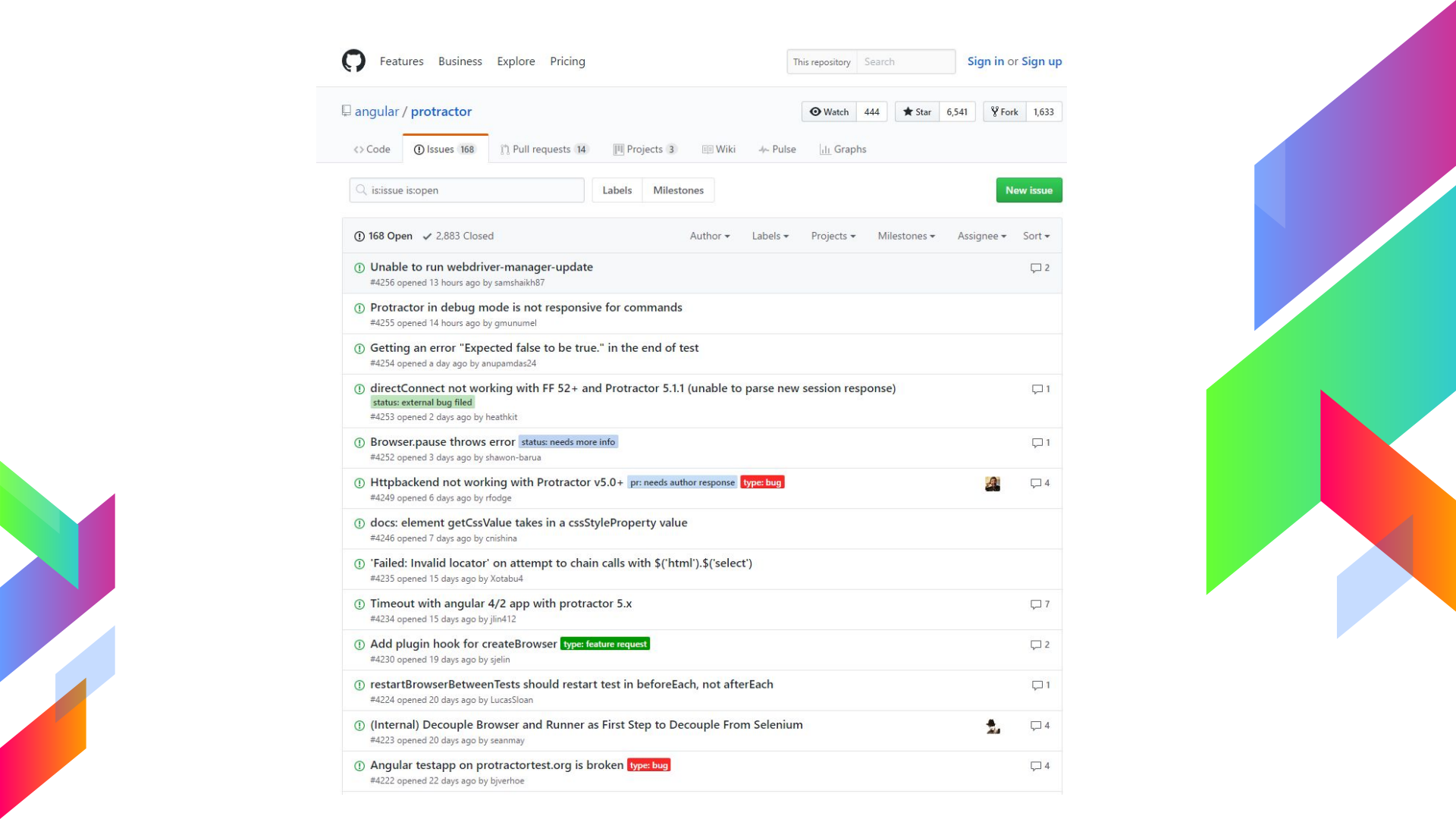


Test Automation

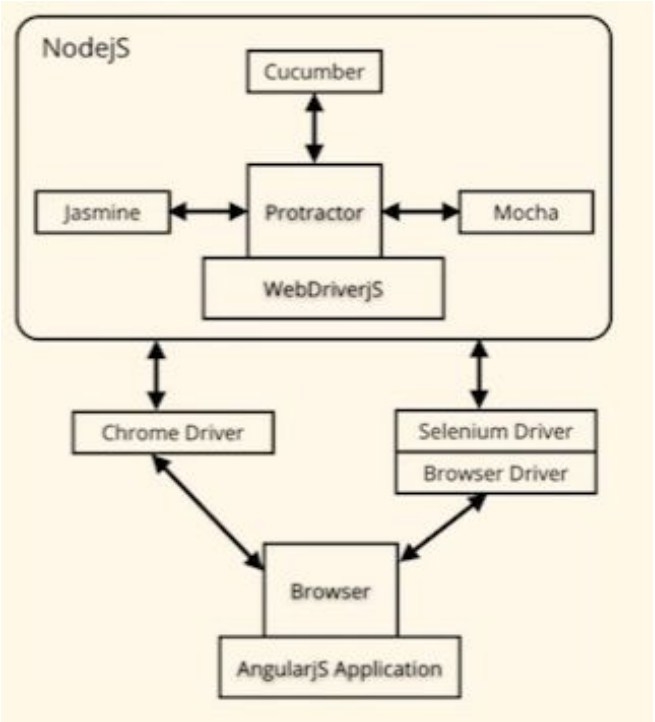
- Run **fast** and **frequently**
- Optimization of **speed, efficiency, quality**
- **Decrease costs**
- **Quickly react** (Agile/Cont. Deployment)
- Advance a tester **motivation** and **efficiency**

Protractor

- End to end test framework, first released in 2013
- Wrapper on top of WebdriverJS to control browsers
- Node.js program
- Written by Julie Ralph - SW Engineer in Google
- Public on GitHub
- @Juliemr is very active, answer feature requests, fixing bugs
- We can simulate clicks, drag and drop, any action the user can do with the browser


Features Business Explore Pricing
This repository Search
Sign in or Sign up
angular / protractor
Watch 444 Star 6,541 Fork 1,633
Code Issues 168 Pull requests 14 Projects 3 Wiki Pulse Graphs
is:issue is:open
Labels Milestones
New issue
168 Open 2,883 Closed
Author Labels Projects Milestones Assignee Sort
Unable to run webdriver-manager-update
#4256 opened 13 hours ago by samshaikh87
2
Protractor in debug mode is not responsive for commands
#4255 opened 14 hours ago by gmunumel
Getting an error "Expected false to be true." in the end of test
#4254 opened a day ago by anupamdas24
directConnect not working with FF 52+ and Protractor 5.1.1 (unable to parse new session response)
status: external bug filed
#4253 opened 2 days ago by heathkit
1
Browser.pause throws error status: needs more info
#4252 opened 3 days ago by shawon-barua
1
Httpbackend not working with Protractor v5.0+ pr: needs author response type: bug
#4249 opened 6 days ago by rfodge
4
docs: element getCssValue takes in a cssStyleProperty value
#4246 opened 7 days ago by cnishina
'Failed: Invalid locator' on attempt to chain calls with $('html').$('select')
#4235 opened 15 days ago by Xotabu4
Timeout with angular 4/2 app with protractor 5.x
#4234 opened 15 days ago by jlin412
7
Add plugin hook for createBrowser type: feature request
#4230 opened 19 days ago by sjelin
2
restartBrowserBetweenTests should restart test in beforeEach, not afterEach
#4224 opened 20 days ago by LucasSloan
1
(Internal) Decouple Browser and Runner as First Step to Decouple From Selenium
#4223 opened 20 days ago by seanmay
4
Angular testapp on protractortest.org is broken type: bug
#4222 opened 22 days ago by bjverhoe
4

The frameworks inside Protractor



WebDriver JS

- WebDriver refers to the language bindings and the implementation of the **browser control code**
- **WebDriverJS** has an important difference from other languages - it is **asynchronous: last action we perform is resolved so as to continue**
- Control Flow
  - no **Sleep**
  - no **Wait**
  - no **Promises**
- Managing entirely asynchronuous API (WebdriverJS)
- **Queue of pending promises**

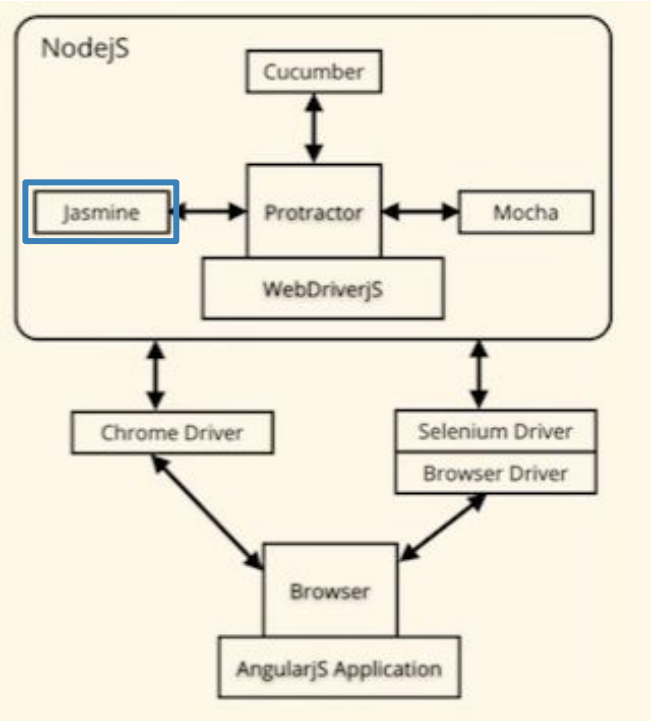
Easy to Catch Elements

Protractor Global Variables

- element
- by
- browser

```
element(by.id('someId'));
element(by.css('#search'));
element(by.model('name'));
browser.get(url);
browser.switchTo().iframe(indexOrName);
browser.wait(1000);
```

The frameworks inside Protractor



Jasmine-reporters

- **Jasmine** is a testing framework for javascript
- **Protractor** uses **Jasmine** for its test syntax
- **describe, it, expect, toBe**

Test Structure Scenario

beforeEach
    command
afterEach
    command
it
    command
    expectation

Let See How It Really Works!

Setup

- Use npm to install Protractor globally
  >npm install -g protractor@5.1.1

This will install two command line tools, Protractor and webdriver-manager
Try running **protractor --version** to make sure it's working

- Update and start webdriver
  >webdriver-manager update
  >webdriver-manager start

This will start up a Selenium Server and will output a bunch of info logs
Information about the status server at http://localhost:4444/wd/hub

- Use npm to install jasmine-reporters globally
  >npm install -g jasmine-reporters@^2.0.7

Write a test

todo-spec.js

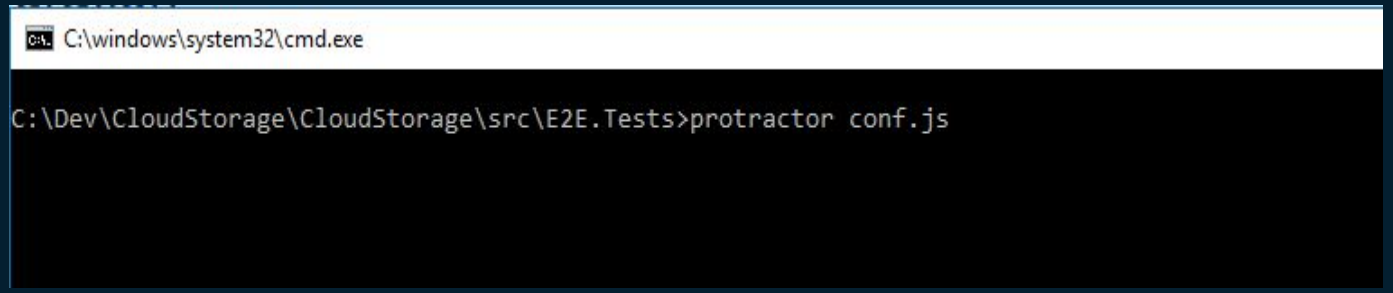
```
describe("hello-protractor", function () {
    it("Should display the correct title", function () {
            browser.driver.get('http://localhost:57483');

            expect(browser.driver.getTitle()).toEqual('Login');
        });
    });
```

Configuration

conf.js

```
1  exports.config = {
2    seleniumAddress: 'http://localhost:4444/wd/hub',
3    specs: ['todo-spec.js']
4  };
```

Run the test




```
C:\Dev\CloudStorage\CloudStorage\src\E2E.Tests>protractor conf.js
[16:58:27] I/launcher - Running 1 instances of WebDriver
[16:58:27] I/hosted - Using the selenium server at http://localhost:4444/wd/hub
Started
..


2 specs, 0 failures
Finished in 6.445 seconds
```

Interacting with elements

- **Locators**
  - Tell Protractor how to find a certain DOM element
  - by.css(‘.myClass’); by.id(‘myid’); by.model(‘name’);

- **Actions**
  - Asynchronous, all actions return a promise

```
var el = element(locator);
    el.click();
    el.clear();
    el.sendKeys(‘myText’);
    el.getAttribute(‘value’);
```

  -

Interacting with elements

- **Find multiple elements**
  - use element.all to deal with multiple DOM elements
  - element.all() has several helper functions

```
element.all(locator).count();
element.all(locator).get(index);
element.all(locator).first();
element.all(locator).last();
```

- **Find Sub-elements**
  - use chained locators to find a list of sub-elements

```
element(by.css('some-css')).element(by.tagName('tag-within-css'));
```

  - use chained locators to find a list of sub-elements

```
element(by.css('some-css')).all(by.tagName('tag-within-css'));
element(by.css('some-css')).first().element(by.tagName('tag'));
```

Writing multiple scenarios

todo-spec.js

```
describe("hello-protractor", function () {

    it("Should display the correct title", function () {

            browser.driver.get('http://localhost:57483');
            expect(browser.driver.getTitle()).toEqual('Login');
        });

    });

    it('Should add usename and password then login', function(){

            browser.driver.findElement(by.id('Username')).sendKeys('test');
            browser.driver.findElement(by.id('Password')).sendKeys('Password');
            browser.driver.findElement(by.id('RememberMe')).click();

            browser.driver.findElement(by.id('login')).click();
       });

```

Changing the configuration

conf.js

```
1  exports.config = {
2    seleniumAddress: 'http://localhost:4444/wd/hub',
3    specs: ['todo-spec.js'],
4
5    capabilities: {
6      browserName: 'firefox'
7    }
8
9  };
```

```
C:\Dev\CloudStorage\CloudStorage\src\E2E.Tests>protractor conf.js
[17:29:34] I/launcher - Running 1 instances of WebDriver
[17:29:34] I/hosted - Using the selenium server at http://localhost:4444/wd/hub
Started
..


2 specs, 0 failures
Finished in 0.615 seconds

[17:29:39] I/launcher - 0 instance(s) of WebDriver still running
[17:29:39] I/launcher - firefox #01 passed
```

Changing the configuration

conf.js

```
exports.config = {
  seleniumAddress: 'http://localhost:4444/wd/hub',
  specs: ['todo-spec.js'],

  multiCapabilities: [{
    browserName: 'firefox'
  }, {
    browserName: 'chrome'
  }]
};
```

```
[17:34:37] I/testLogger - [firefox #01] PID: 11156
[firefox #01] Specs: C:\Dev\CloudStorage\CloudStorage\src\E2E.Tests\todo-spec.js
[firefox #01]
[firefox #01] [17:34:31] I/hosted - Using the selenium server at http://localhost:4444/wd/hub
[firefox #01] Started
[firefox #01] ..
[firefox #01]
[firefox #01]
[firefox #01] 2 specs, 0 failures
[firefox #01] Finished in 0.933 seconds
[firefox #01]

[17:34:37] I/testLogger -

[17:34:37] I/launcher - 0 instance(s) of WebDriver still running
[17:34:37] I/launcher - chrome #11 passed
[17:34:37] I/launcher - firefox #01 passed
```

What did we learn?

```
describe('angularjs homepage', function() {
  it('should greet the named user', function() {
  browser.get('http://www.angularjs.org');
  element(by.model('yourName')).sendKeys('Julie');
  var greeting = element(by.binding('yourName'));
  expect(greeting.getText()).toEqual('Hello Julie!');
});

describe('todo list', function() {
  var todoList;
  beforeEach(function() {
  browser.get('http://www.angularjs.org');
  todoList = element.all(by.repeater('todo in todos'));
});

it('should list todos', function() {
  expect(todoList.count()).toEqual(2);
  expect(todoList.get(1).getText()).toEqual('build an angular app');
});

it('should add a todo', function() {
  var addTodo = element(by.model('todoText'));
  var addButton = element(by.css('[value="add"]'));

  addTodo.sendKeys('write a protractor test');
  addButton.click();

  expect(todoList.count()).toEqual(3);
  expect(todoList.get(2).getText()).toEqual('write a protractor test');
});
```

Example Spec from GitHub

Challenges

- Lack of Domain Specific Language Support
- Code Duplication
- High Coupling
- Though Maintenance

Ideally

- Domain specific tests
- Flexible and ready for change
- Easy to maintain
- Avoid code duplication

Page Objects

- **Design Pattern** popular in test automation
- Enhancing **test maintenance**
- **Reducing** code duplication
- Object-oriented class that serves as an interface to a page
- The tests use the methods of this page object class

- **Tests will not change with the UI changes only the code within the page object**

The page object

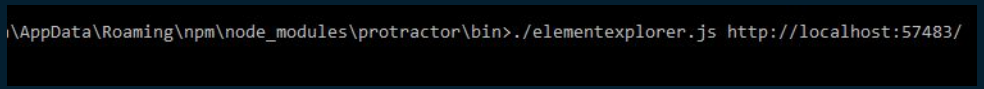
```
var AngularHomepage = function() {
  var nameInput = element(by.model('yourName'));
  var greeting = element(by.binding('yourName'));

  this.get = function() {
    browser.get('http://www.angularjs.org');
  };

  this.setName = function(name) {
    nameInput.sendKeys(name);
  };

  this.getGreeting = function() {
    return greeting.getText();
  };
};
```

Refactored Spec

```
describe('angularjs homepage', function() {
  it('should greet the named user', function() {
    var angularHomepage = new AngularHomepage();
    angularHomepage.get();

    angularHomepage.setName('Julie');

    expect(angularHomepage.getGreeting()).toEqual('Hello Julie!');
  });
});
```

Debugging Tools

- broswer.**pause**();
  - in your test code: browser.pause();
- broswer.**debugger**();
  - in the terminal: protractor debug conf.js
- **elementExplorer.js**
  - \AppData\Roaming\npm\node_modules\protractor\bin



- **document.querrySelectorAll();**

Test like a Pro

URL: https://signin.stag.visma.net/
Username: demopudd@gmail.com
Password: Password11!

Questions?

monica.iovan@visma.com
bradatan.ionut@visma.com
emil.craciun@visma.com
adriana.hazulea@visma.com
zoltan.sandor@visma.com