Giving Your Technical Vision a Voice

Here for the workshop? Move to the front and introduce yourself to your neighbors.

Giving Your Technical Vision a Voice

Joanna Power

September 2016

@atsalix

Introductions

- Me to you
- You to your neighbors
- You to me

Supplies

- 8 sheets of paper
- With laptop, only 2

Technical design documents

Why do devs hate writing them?

- Nobody reads them.
- They get out of date.
- I'm not a good writer.
- I know what I'm doing.
- I don't know what I'm doing.
- I just want to write code.

Workshop guidelines

- Everyone is included
- Feedback is actionable, specific and kind

Workshop goal

By the end of today…

- You will have written most of a technical design document.
- You will know how to do it again.

What is a technical design document?

- Blueprint for software
- Describes problem
- Presents solution

What isn't it?

- Functional spec
- Visual mockups
- Implementation tasks
- Operational playbook

Who writes it?

Developer or architect

Who reads it?

Developers and architects

What good is it?

- Better product
- Fewer surprises during development
- More team involvement

Why me?

- Grow as an engineer
- Earn teammates' trust
- Earn your own trust
- Demonstrate technical leadership

Questions?

Document structure

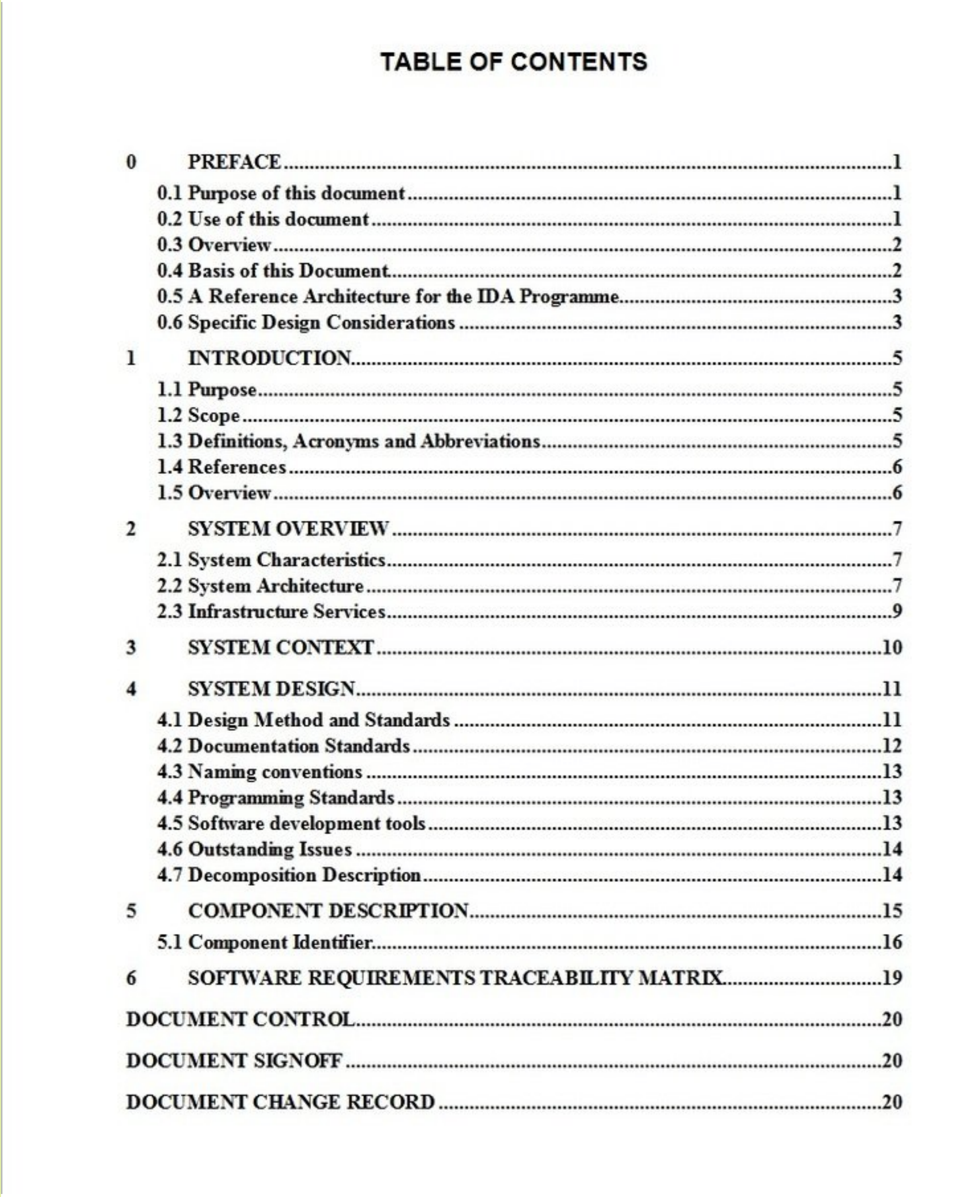


TABLE OF CONTENTS

4GIFs

Document structure

- Overview
- Requirements
- Data model
- System components

Ideas

What software will you document?

- Something you understand
- Something with multiple components

You have 30 seconds.

You have 30 more seconds to write down 2 more ideas.

Share ideas

Choose one idea, explain it to a neighbor, and ask for feedback. In particular…

- If too simple, can other components be included?
- If too complicated, can components be combined or removed?

Actionable, specific, kind.

You have 60 seconds.

Title

On a new piece of paper, write down your title:

<Project> Technical Design

My example:

WeatherWise Technical Design

Overview

Purpose

WeatherWise example:

Getting dressed every day is hard. Sometimes people pick the wrong outfit and are too hot or cold, or they get wet because it rains. Weather Wise helps people choose weather-appropriate outfits every day.

Purpose

What is the problem?

- Why does your software exist?
- What real-world problem does it solve?

You have 60 seconds.

Share purpose

Trade with your neighbor and ask for feedback. In particular…

- Does the purpose make sense?
- Is it compelling?

Actionable, specific, kind.

You have 60 seconds.

Users

WeatherWise example:

WeatherWise helps professionals who hate getting up in the morning and can't think before they've had coffee.

Users

Who are the users?

- What are they like?
- Why do they need your software?

You have 60 seconds.

Share user description

Trade with your neighbor and ask for feedback. In particular…

- Does the user exist?
- Does the user need the software?

Actionable, specific, kind.

You have 60 seconds.

Requirements

Requirements

WeatherWise example:

- Show weather forecast in relation to yesterday's weather; e.g. "warmer & wetter"
- Suggest outfit from user's wardrobe
- Remember favorite outfits

Requirements

What does the software do?

- Hit the high points
- Be concise

You have 60 seconds.

Share requirements

Trade with your neighbor and ask for feedback. In particular…

- Do the items make sense?
- Is anything big missing?

Actionable, specific, kind.

You have 60 seconds.

Non-requirements

WeatherWise example:

- No offline support
- No web version
- No locale-based ad integration

Non-requirements

What doesn't the software do?

- Address obvious gaps
- Be concise

You have 60 seconds.

Share non-requirements

Trade with your neighbor and ask for feedback. In particular…

- Do the items make sense?
- Is anything big missing?

Actionable, specific, kind.

You have 60 seconds.

Overview, again

Design summary

WeatherWise example:

WeatherWise is a **native mobile app** for **Android and iPhone**. The app uses a **REST API** to communicate with a **backend service** to get the relative forecast and general outfit suggestions, e.g. blazer and pants. Using the **device cache**, the app maps the outfit suggestions to the user's wardrobe, e.g. black wool blazer and sparkly leggings. The backend service...

Design summary

What is the elevator pitch?

- General architecture
- Major technologies

You have 60 seconds.

Share design summary

Trade with your neighbor and ask for feedback. In particular…

- Is the general architecture clear?
- Are the main technologies named?

Actionable, specific, kind.

You have 60 seconds.

Data model

Data entities

Data entities

What are the entities?

- Name everything

You have 30 seconds.

Entity details

WeatherWise example:

Forecast

- date (datetime)
- postalCode (string)
- precipRange (int, int)
- tempRange (int, int)
- windRange (int, int)
- warmthIndex (float)

Entity details

What are the internal data attributes of one entity?

- Name everything
- Type everything

You have 30 seconds.

Entity relationships

WeatherWise example:

- An outfit contains one or more garments
- A garment belongs to zero or more outfits

Entity relationships

How do the entities relate?

- Show relationships with cardinality

You have 30 seconds.

Share entities and relationships

Trade with your neighbor and ask for feedback. In particular…

- Do the entities make sense?
- Is naming consistent?
- Are the relationships clear?

Actionable, specific, kind.

You have 60 seconds.

System components

Components

WeatherWise example:

- Native mobile app
- Backend service
- Third-party weather service

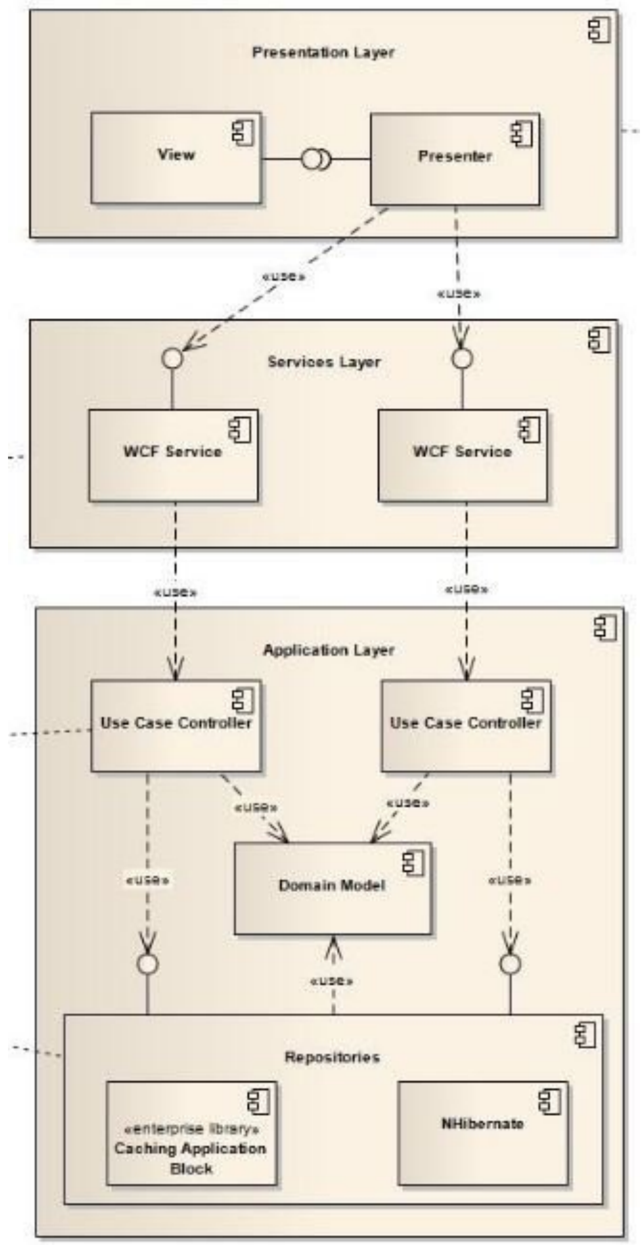
Components

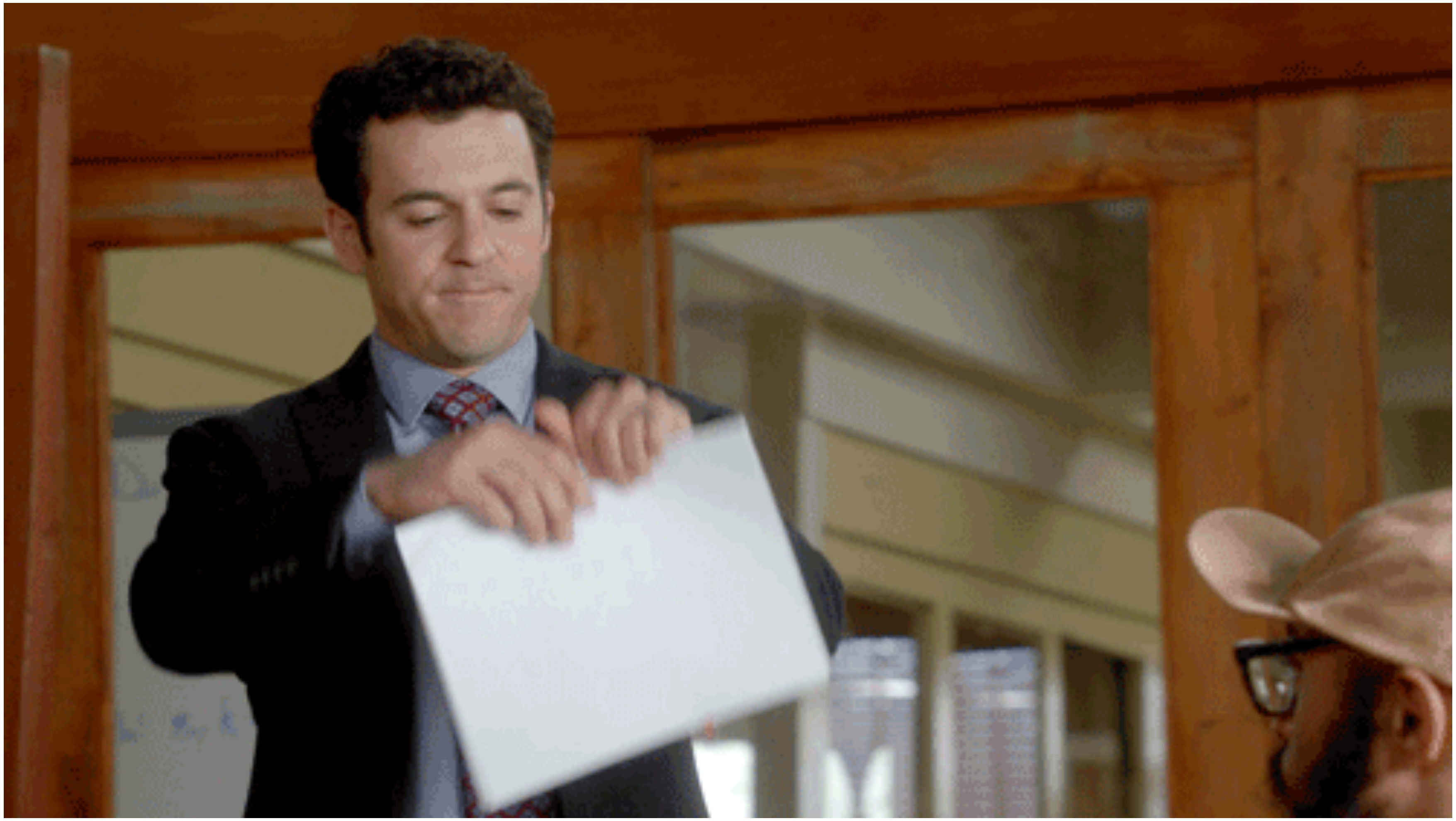
What are the separate parts of the system?

- Name everything

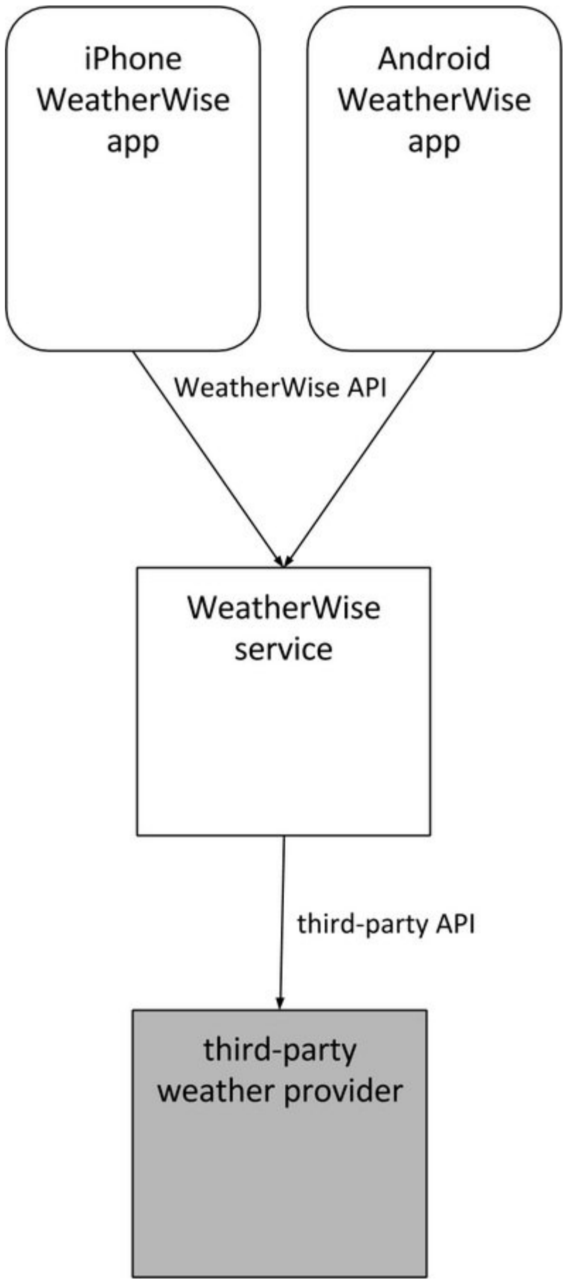
You have 30 seconds.

Component diagram





Component diagram



Component diagram

How are the components connected?

- Name everything
- Label connections
- Indicate directionality

You have 90 seconds.

Share component diagram

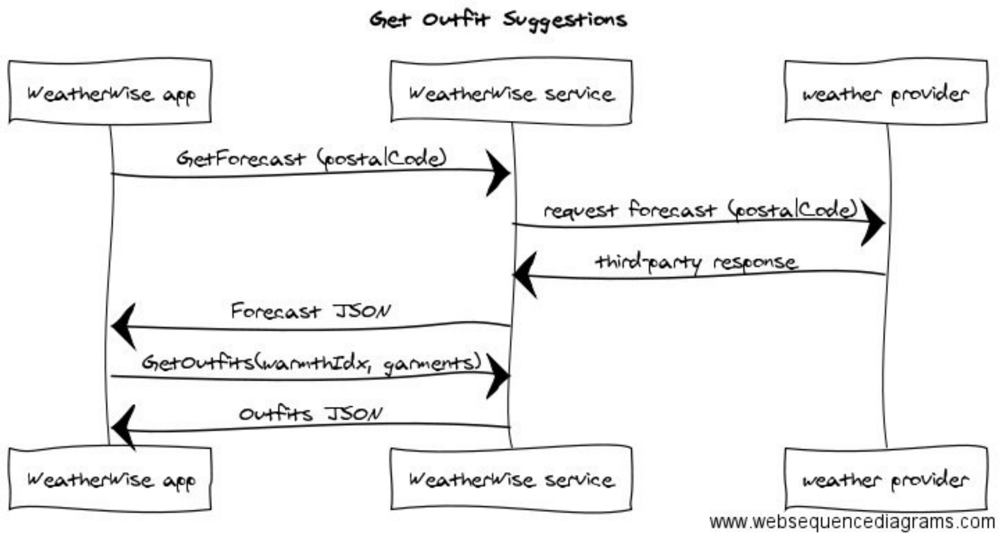
Trade with your neighbor and ask for feedback. In particular…

- Is anything missing?
- Is naming consistent?
- Are the connections directed and labeled?

Actionable, specific, kind.

You have 60 seconds.

Interaction diagram



Interaction diagram

How do the components communicate?

- Components are actors
- Time moves down
- Requests labeled with params
- Responses labeled with return data

You have 90 seconds.

Share interaction diagram

Trade with your neighbor and ask for feedback. In particular…

- Does communication sequence make sense?
- Are the requests and responses directional and labeled?

Actionable, specific, kind.

You have 60 seconds.

Component details

WeatherWise example:

The WeatherWise mobile app

The app communicates with the backend service using a REST API over HTTPS. All user data is kept on device, so no authentication is required. The app maintains user wardrobe database in the device cache. To get outfit suggestions, the app calls the GetOutfits API, providing warmth index and list of available garments....

Component details

What is inside one component?

- Responsibilities
- Data dependencies, other dependencies
- Logic
- Technologies and infrastructure

You have 90 seconds.

Share component details

Trade with your neighbor and ask for feedback. In particular...

- Are responsibilities clear?
- Are dependencies clear?
- Do technologies make sense?
- Is anything missing?

Actionable, specific, kind.

You have 60 seconds.

That's everything!

Overview

- Purpose
- Users
- Design summary

Requirements

- Requirements
- Non-requirements

Data model

- Data entities
- Entity details
- Entity relationships

System components

- Component diagram
- Component details
- Interaction diagrams

Put it together!

Overview

- Purpose
- Users
- Design summary

Requirements

- Requirements
- Non-requirements

Data model

- Data entities
- Entity details
- Entity relationships

System components

- Component diagram
- Component details
- Interaction diagrams

Workshop goal

By now...

- You have written most of a technical design document.
- You know how to do it again.
- You have practiced giving and receiving feedback that is actionable, specific and kind.

Team feedback

Strategy

- While writing, discuss with trusted teammate.
- After writing, share with team and request feedback.
- Organize and lead design review.
- Incorporate feedback.
- Start coding!

Production system

Overview

Requirements

Data model

System components

Metrics, monitoring, logging

Security

Deployment plan

Appendices

But what about…?

- API specs
- Class diagrams
- Future work
- Links to resources
- etc.

Put it in an appendix.

Resources

http://stackoverflow.com/questions/677901/how-do-i-write-a-technical-specification-document-for-my-software-project

http://www.joelonsoftware.com/articles/AardvarkSpec.html

http://blog.slickedit.com/2007/05/how-to-write-an-effective-design-document/

http://www.yegor256.com/2015/11/10/ten-mistakes-in-specs.html

https://github.com/jpowerwa/tech-talks/act-w-2016/