Codemill

- Young company
  - Founded 2007
  - Today 30 employees
- Wide range of projects
  - iPhone/Android apps
  - Web applications
  - Life science
  - Media Asset Management (MAM)

Martin

- Studied here in Umeå
- First TFE
- Master in Computer Science afterwards
- Thesis at Codemill
- Consultant

Today

- Technologies
- Frameworks
- Case-study
- Focus on
  - Tests
    - Make them less magic and get encouraged to start writing them
    - REST API
    - As an example
- Please ask questions
  - Yes, even if it is something I mentioned a while ago and you got stuck thinking about.
  - You learn more if you do.

First off

- Consultant
  - Never infinite resources.
  - Bang-for-buck
- Page with 500 database queries, loads in 1 sec.
  - Really worth 2,000-5,000 kr to optimize for >0.5s?
- Impacts technology choices

We use a lot of

- Django
  - Python framework
  - Model View Controller
  - Good security practices "for free"
  - Easy to test
    - Very nice array of tools to ease testing.
    - Sane project structure
We use a lot of

- Java (EE)
  - Typed language
  - Very mature platform
  - Lots of good tools and utilities
- Google Web Toolkit
  - Java code compiled to HTML and JavaScript for frontend
- Angular
- Other frontend/template

Not limited to Java & Python

- C/C++
- PHP
- Perl
- C#
- Shellscript
- Ruby

REST

- If we get the choice REST wins
  - Easy to work with.
  - Supported by all (major) languages.
  - Rich frameworks.
  - Easy to integrate with (near) future.
  - For applications and client-side.
  - Machine + Webpage use same API
  - No wrapper to convert data into JSON.
  - However not all REST API:s support JSON.
  - Most customer API:s are REST.
  - At times SOAP or other XML-RPC variation.

Testing

- Manual testing is boring
- Manual testing is slow
- Manual testing is never done for everything
  - You cut corners
- 100% code coverage a waste of time
  - Waste of resources (time)
Testing – New code

- You find edge cases faster.
  - Only focusing on what your code will produce, not how, you’re more likely to spot edge cases.
- When someone wants to know what your code does it is the fastest way to show them.
  - Helps in code review.
- When the tests pass you are done.
  - Start out with a failing test so you know that is working.
    - Working means failing as expected.

Test example

- Tests should never depend on each other.
  - Once forced to
    - Alternative: Stop third party application, reset database, reset the application, reset the database, run test, stop third party app.
    - 5-15 mins running a few test cases. Whole suite took ages.
- Early development
  - Small code changes
  - Very little code is written.
  - After 70-95% done
    - A few large changes
  - Go through the list and delete what we created.
  - When all tests have run, restore the database from snapshot.

Testing – Extend with new features

- Don’t break existing features
  - This can be subtle
  - Web applications are messy by nature
    - Information aggregated in multiple places
    - Same as for new features
      - Done when tests pass
      - Find edge cases
      - Help in code review
- Example of mentioned mess
  - Touches upon optimization
  - Organisation has X departments
    - Each department has Y projects
    - Each project has Z resources
  - Render an overview page with information about everything
    - Many HTTP requests
      - Common solution
      - How departments?/project?/resource?/count?=/validdepartment_head=true
      - Now department code is reused project and resource code.
    - When you change Resource or Project you can break Department-list.

Testing – Change requests

- Rework a feature or workflow
- Early development
  - Small code changes
    - Very little code is written.
- After 70-95% done
  - A few large changes
Tests

- Types of tests
  - Unit
    - Test a single, or at least very limited, amount of methods or functions. You test a unit of source code.
  - Integration
    - Test how interaction between modules work. Ensure that the integration is working as planned.
  - End-to-end
    - Ensure that the whole thing works. Test from a user point of view. Insert data and ensure the correct things happen and returning expected data
  - Smoke
    - Is there something missing out?
    - Useful at deployment before allowing traffic
  - Good when optimizing and/or refactoring.

Testing – Mocks/fakes

- Good when
  - When working with (slow) remote API
  - External API lacking test flag/environment
  - Mimics external resource
    - /staff?
      - role=commissioner

Tests – How to design your code

- How to make your code easy to test
  - Structured
  - Separation of concerns
  - Spaghetti is not fun to work with

- Constructed example
  - Two external resources you can’t change and doesn’t do exactly what you need.

- Should these lines be moved to Staff.get_commissioner_by_name_or_404(name)?

In a Controller:

```python
commissioner_name = commission.commissioner_name
resp = get(remote_api.url/staff?role=commissioner
commissioners = data_utils.get_commissioners(resp.content)
for commissioner in commissioners:
    # commissioner.full_name = ... commissioner_name:
    commissioner_obj = commissioner
    break
```

Tests – Disable features and test modes

- Sometimes features need to be turned off for efficient testing
  - Database search engine
  - Hard to say when to add this option
    - Clean code makes it easier to patch in.
  - Test mode is useful for developers.

```python
settings testing_mode = True
request signature | = sign_registration_request(password, request)
if testing_mode: request[testing_mode] = 1
url = registration_url(request)
```

Tests – How to keep writing them

- We’ve experimented with quite a few options
  - Unit test close to everything
    - Results in maintenance as 5 min code change, 30-60 min test updates.
  - Primarily integration and end-to-end tests
  - Hybrids of the two above

- Conclusion
  - Goal is to deliver correct set of features
    - Without bugs
    - Unit test complex methods/functions
    * Complex code
    * Calculations
  - Cover the rest with larger integration tests or end-to-end tests
  - Time lost when fixing a rare bug is gained by not writing and maintaining loads of trivial tests.

Tests – Mixed examples

- We’ve experimented with quite a few options
- Times of tests close to everything
- Results in maintenance as 5 min code change, 30-60 min test updates.
- Primarily integration and end-to-end tests
- Hybrids of the two above

- Conclusion
- Goal is to deliver correct set of features
  - Without bugs
  - Unit test complex methods/functions
    * Complex code
    * Calculations
  - Cover the rest with larger integration tests or end-to-end tests
  - Time lost when fixing a rare bug is gained by not writing and maintaining loads of trivial tests.
Tests – Mixed examples 2

class OrderTests(TestCase):
    def test_resend_keys_from_removed_products(self):
        ...       self.assertFalse(order.is_settled())
        order.mark_as_settled()
        self.assertTrue(order.is_settled())

Media Asset Management (MAM)

- Domains
  - What you’re actually working with, what problems are you solving and why
  - Hardware controls
  - Workflow

Optimization – A few hints

- First
  - Never, ever, without benchmarks
- Never coders focus on level
- Is if () else if () better than switch () [case A, case B, default] in this language?
- Focus on the macros perspective instead
  - Reduce amount of database queries
  - Reduce amount of HTTP requests
  - Cache state is very easy to write
    - The benefit is that it's a way of thinking about what to cache and how long
    - If you mess up you can remove it
    - This is faster than ever
- Example
  - Give a few days per day or per server (google how much time you save if you save 150 CPUs instructions on an intel xenon or equivalent)
  - Reduce CPU idle time. 0.2 seconds per day by reducing periods during peak load.
  - 30 http requests = 30*0.25, about 7.5 seconds
  - 5 requests = 5*0.25, about 1.25 seconds
  - You just saved 6.25 seconds of each page load
- LIMIT
  - Never, ever, without benchmarks

MAM – Sample workflow

- Keys with AvidMedia to produce a promo, take a look here how we sourced from Five Davis Cup
- First VIDEODetails searches for all clips of 2013 and 2014 Davis Cup. Browse older footage might be useful too, to create a Fitzroy
- This is a very easy task
- The hard part people talk about is figuring out what to cache and for how long
- More or less fuzzy guidelines
- Find what works for your assignments and they will be timesavers.

Tests – Final words

- How will it look at my future workplace?
  - Some companies have good coverage
  - They have a clear structure
  - Some want to work on it but have not yet a clear direction of how they want to do it
  - Some things are not hard to test and automatically that it just isn’t worth it
  - Hardware controls
    - Hard to make 100% reliable mocks mimicking all hardware parts
    - Yes, tests can be helpful
  - I hope you gave writing tests a serious try
    - Find what works for your assignments and they will be timesavers.

Architecture – Overview

- Image removed

Tests – Final words

- How will it look at my future workplace?
  - Some companies have good coverage
  - They have a clear structure
  - Some want to work on it but have not yet a clear direction of how they want to do it
  - Some things are not hard to test and automatically that it just isn’t worth it
  - Hardware controls
    - Hard to make 100% reliable mocks mimicking all hardware parts
    - Yes, tests can be helpful
  - I hope you gave writing tests a serious try
    - Find what works for your assignments and they will be timesavers.

Architecture – Overview

- Image removed
Transcoder - ComplexJob

- View in browser

Middleware - Item

- View in browser