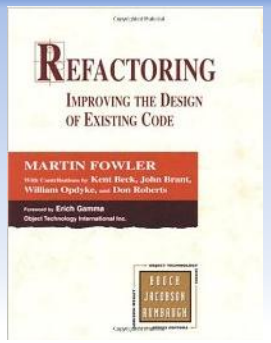
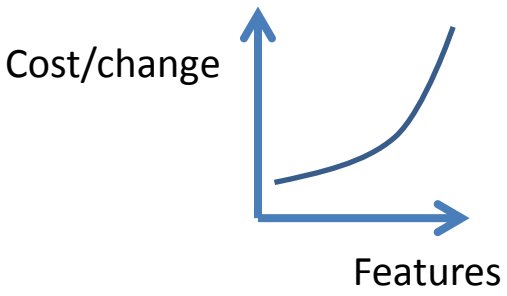
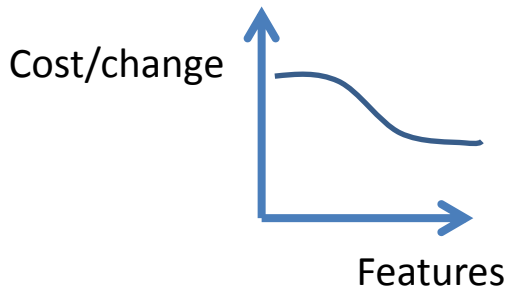
Refactoring

What is refactoring?



- "… the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure."
  Refactoring, Martin Fowler, 1999
- A good design first will later fall apart during changes
- Refactoring takes any design and improve it





How do you do refactoring?

- Write tests to verify external behavior
  - Hint: we just did that!
- Apply some refactoring
- Run unit tests to check we did mess up

Example refactoring

- Today, we have support in our IDE for it
  - E.g. extract method

```
// Move Tool bars
settings.setToolbarAlign(TinyMCESettings.Align.left);
settings.setToolbarLocation(TinyMCESettings.Location.top);
settings.setStatusbarLocation(TinyMCESettings.Location.bottom);
```

    - A method has a number of lines together with a comment
      - Understand purpose of the lines
      - Invoke extract method, select a name that conveys purpose
      - Delete comment
      - Run unit tests

```
private void moveToolbars(TinyMCESettings settings) {
    settings.setToolbarAlign(TinyMCESettings.Align.left);
    settings.setToolbarLocation(TinyMCESettings.Location.top);
    settings.setStatusbarLocation(TinyMCESettings.Location.bottom);
}
```

Refactoring exercise

- Put the AddressBook under test (& all related logic)
  - Refactor just enough to enable testing
  - Use the tests as protection for further refactoring
  - Don't change public-facing behaviour (other clients may depend on it)
  - Follow red/green/refactor
- Stub out the actual DB, we don't have access to it and don't want to test it.
- Measure & maximize test coverage! (ex: eclemma plugin for eclipse: http://update.eclemma.org)