Real-World Software Development

It's not just about the code

Sandy Kemsley ● www.column2.com ● @skemsley

Agenda

- Why developing software is so much more than just writing code
- Different types of software development lifecycles
- The role of model-driven development

What's Involved in Developing Software?

- Requirements
- Design
- Development
- Testing
- Deployment
- Maintenance

Getting Started: Requirements and Design

Requirements = What

- The overall business goals
- Key performance indicators and metrics
- What business tasks the user needs to accomplish

Design = How

- How the software will support those goals
- How the software will measure performance
- What functions the software will provide to meet those needs
- How the software will fit the existing architecture

Requirements and Design: Separation and Collaboration

- Business people understand business goals
- Business people are not designers

- Designers bring innovative ideas from other fields
- Designers are not subject-matter experts

Making It Work: Development and Testing

Development

- The actual coding
- Technical documentation
- User documentation

Testing

- Does the software run without errors?
- Does the code match the design?
- Does the software satisfy the business requirements?

Development and Testing: Tips from the Real World

- Just because you can code something doesn’t mean that you should
- Put yourself in the intended users’ shoes
- Testing/QA is a gatekeeper to production
- The best testers are devious

Making It Real: Deployment and Maintenance

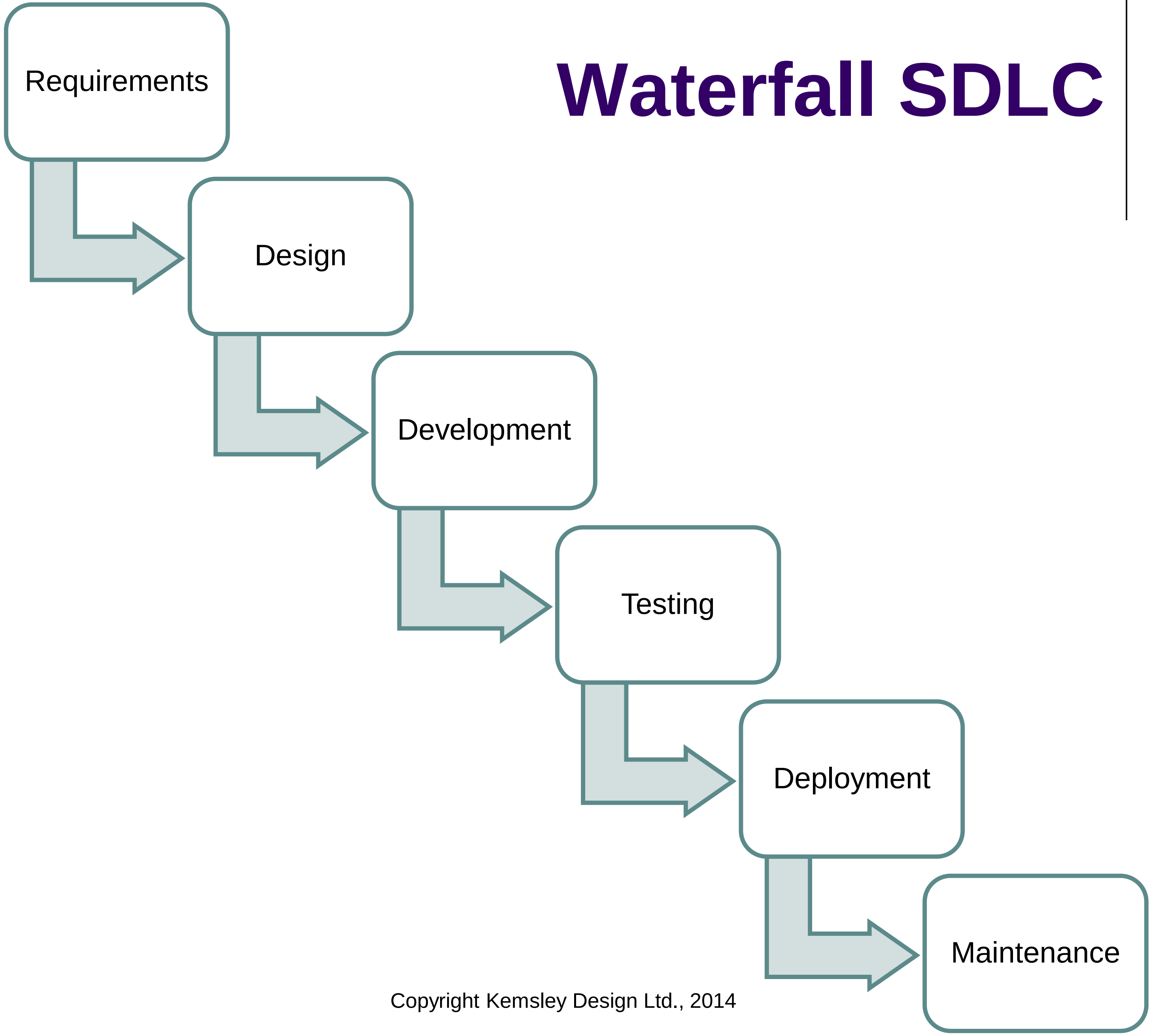
Deployment

- Moving from dev/test to production environment
  - Servers
  - Databases
  - Authentication
  - A million other things
- Performance tuning

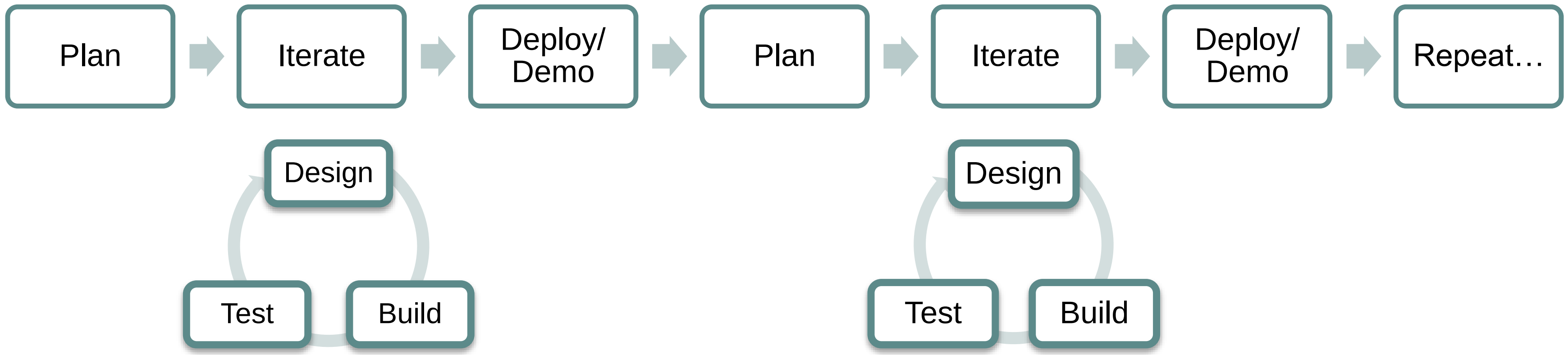
Maintenance

- Supporting users
- Handling system failures
- Identifying bugs and enhancements
  - "Bug" = code does not match design and/or requirements
  - "Enhancement" = requirements do not meet current business need

Waterfall SDLC



Iterative SDLC



Contrasting SDLCs

Waterfall

- Each stage finished and signed off before next stage starts
- Benefit:
  - Requirements/design form a strict contract for outsourced development
- Risk:
  - Requirements change or are incorrect

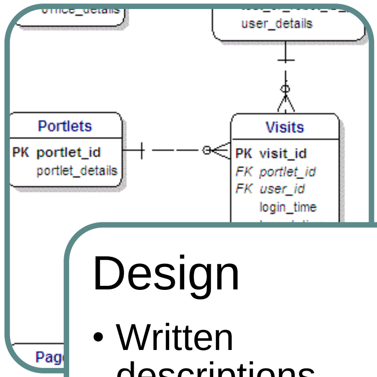
Iterative (e.g., Agile)

- Requirements and design may change at each iteration
- Benefit:
  - Accommodates changing requirements based on interim feedback
- Risk:
  - Reliant on quality of user feedback

Tools Used in the SDLC: Lost In Translation

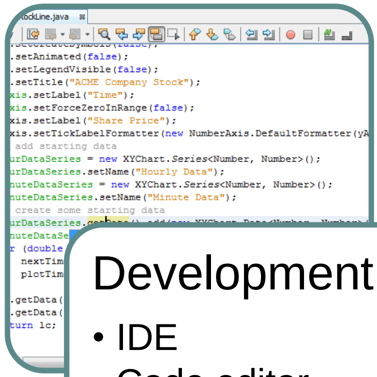
Requirements

- Written descriptions
- Ad hoc diagrams



Design

- Written descriptions
- Domain models (data, process)



```
.setAnimated(false);
.setLegendVisible(false);
.setTitle("ACME Company Stock");
xis.setLabel("Time");
xis.setForceZeroInRange(false);
xis.setLabel("Share Price");
xis.setTickLabelFormatter(new NumberAxis.DefaultFormatter(yA
 add starting data
urDataSeries = new XYChart.Series<Number, Number>();
urDataSeries.setName("Hourly Data");
nuteDataSeries = new XYChart.Series<Number, Number>();
nuteDataSeries.setName("Minute Data");
 create some starting data
urDataSeries.
nuteDataSe
r (double
  nextTim
  plotTim

.getData(
.getData(
turn lc;
```

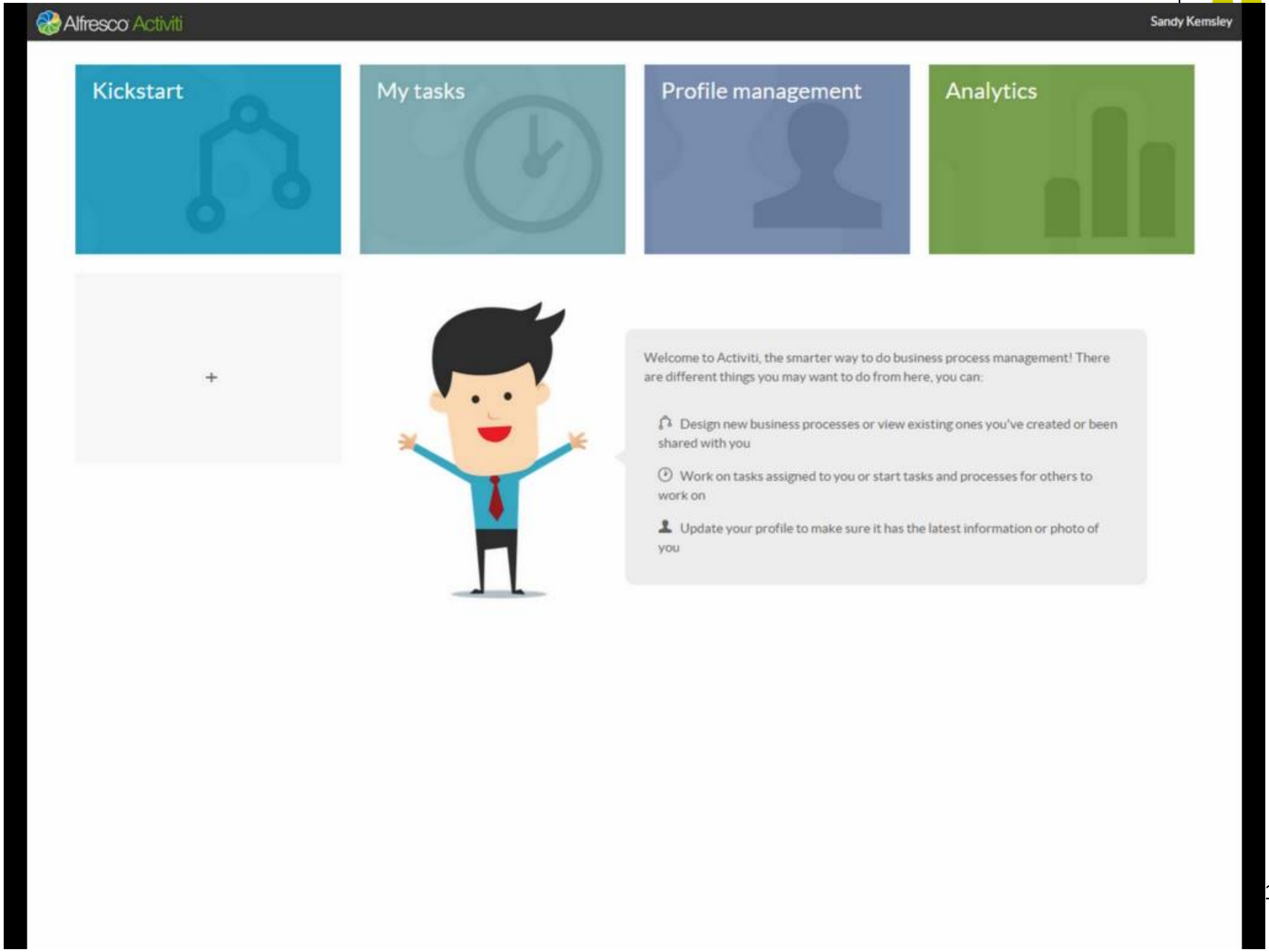
Development

- IDE
- Code editor
- Repository

Model-Driven Development

- Business people and designers use same tool to draw models
- Developers add technical functions to model
- Model is directly executable

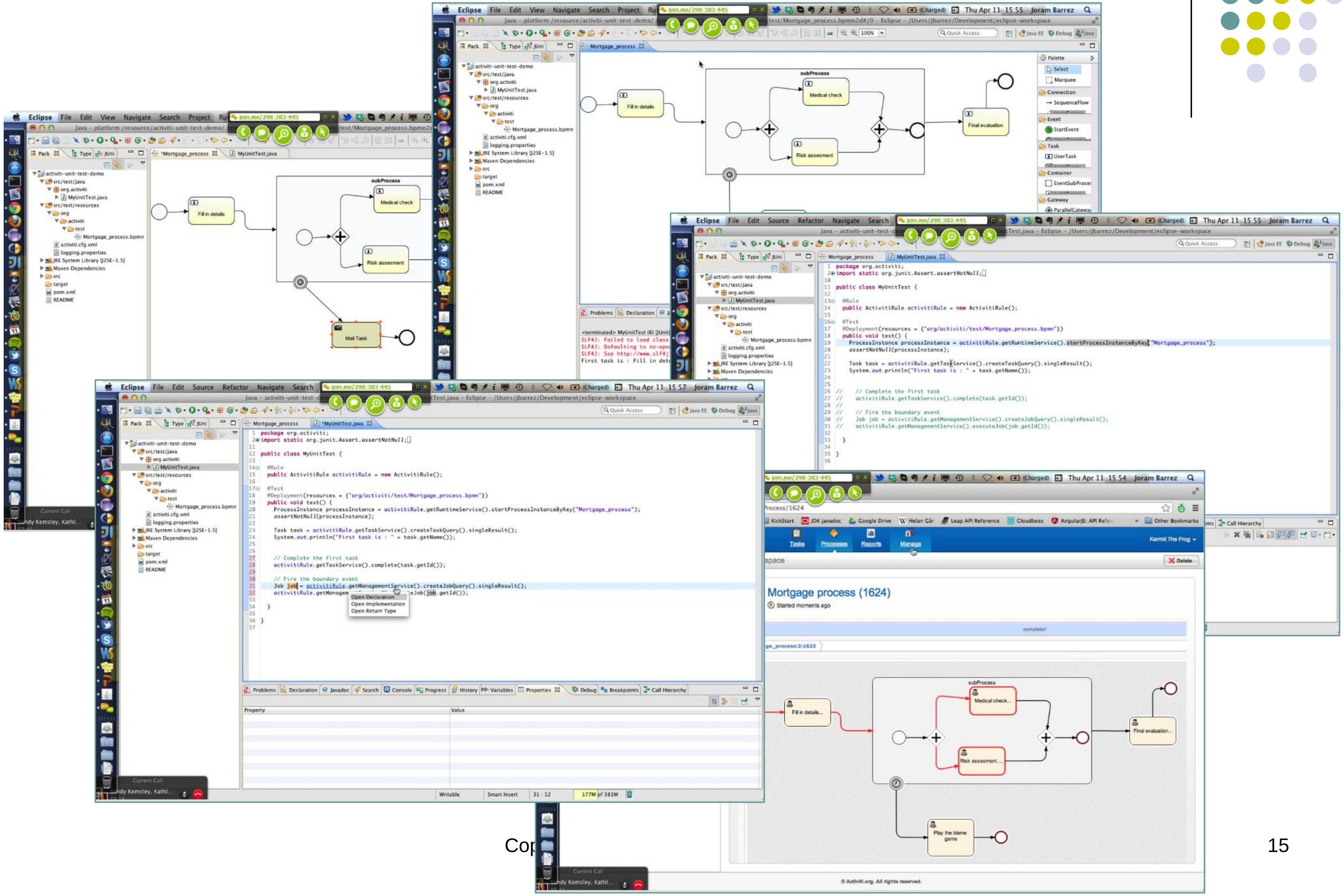
Draw and Execute Directly



Welcome to Activiti, the smarter way to do business process management! There are different things you may want to do from here, you can:

- Design new business processes or view existing ones you've created or been shared with you
- Work on tasks assigned to you or start tasks and processes for others to work on
- Update your profile to make sure it has the latest information or photo of you

Combine IDE and Models



```java
package org.activiti;
import static org.junit.Assert.assertNotNull;

public class MyUnitTest {

  @Rule
  public ActivitiRule activitiRule = new ActivitiRule();

  @Test
  @Deployment(resources = {"org/activiti/test/Mortgage_process.bpmn"})
  public void test() {
    ProcessInstance processInstance = activitiRule.getRuntimeService().startProcessInstanceByKey("Mortgage_process");
    assertNotNull(processInstance);

    Task task = activitiRule.getTaskService().createTaskQuery().singleResult();
    System.out.println("First task is : " + task.getName());


//    // Complete the first task
//    activitiRule.getTaskService().complete(task.getId());
//
//    // Fire the boundary event
//    Job job = activitiRule.getManagementService().createJobQuery().singleResult();
//    activitiRule.getManagementService().executeJob(job.getId());

  }

}
```

```java
package org.activiti;
import static org.junit.Assert.assertNotNull;

public class MyUnitTest {

  @Rule
  public ActivitiRule activitiRule = new ActivitiRule();

  @Test
  @Deployment(resources = {"org/activiti/test/Mortgage_process.bpmn"})
  public void test() {
    ProcessInstance processInstance = activitiRule.getRuntimeService().startProcessInstanceByKey("Mortgage_process");
    assertNotNull(processInstance);

    Task task = activitiRule.getTaskService().createTaskQuery().singleResult();
    System.out.println("First task is : " + task.getName());


    // Complete the first task
    activitiRule.getTaskService().complete(task.getId());

    // Fire the boundary event
    Job job = activitiRule.getManagementService().createJobQuery().singleResult();
    activitiRule.getManagem[illegible]eJob(job.getId());

  }

}
```

Summary

- Think about the entire SDLC, not just coding
- Your code is not useful if:
  - It doesn’t meet the business requirements
  - It doesn’t adhere to design principles
  - It doesn’t work
  - It can’t be supported
  - It can’t be updated
- Model-driven development merges steps within a traditional SDLC

Slides at www.slideshare.net/skemsley

	Sandy Kemsley
	Kemsley Design Ltd.
email:	sandy@kemsleydesign.com
blog:	www.column2.com
twitter:	@skemsley