



Spring ONE

## Where Did All My Beautiful Code Go?

Gregor Hohpe




[gregor@hohpe.com](mailto:gregor@hohpe.com)  
[www.eaipatterns.com](http://www.eaipatterns.com)



*"We hire only the brightest engineers in the industry."*

--Your Company

Google 3



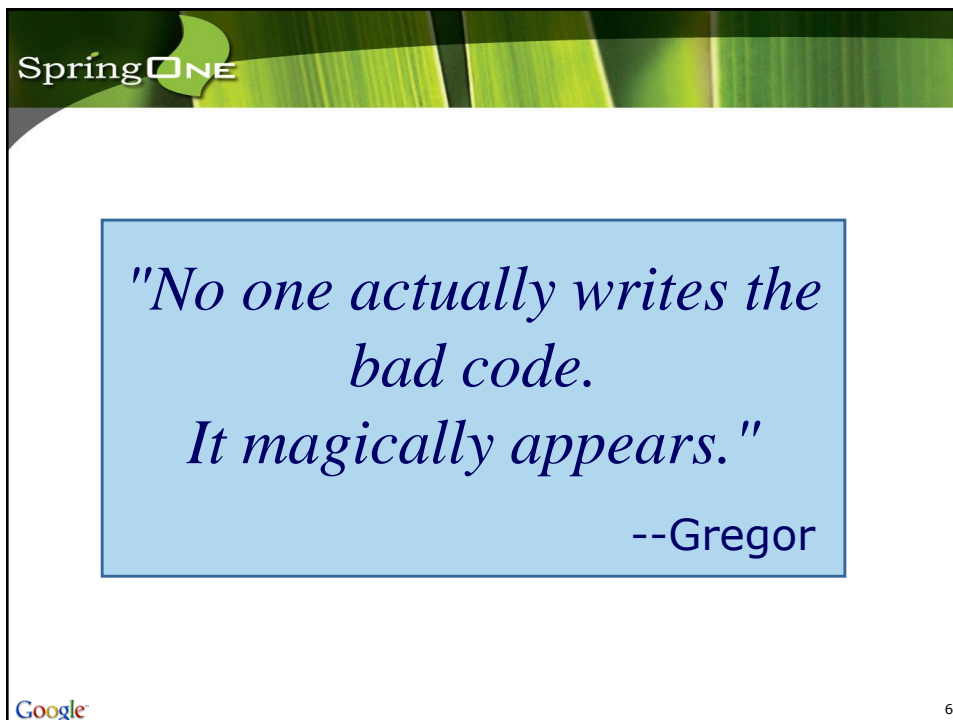
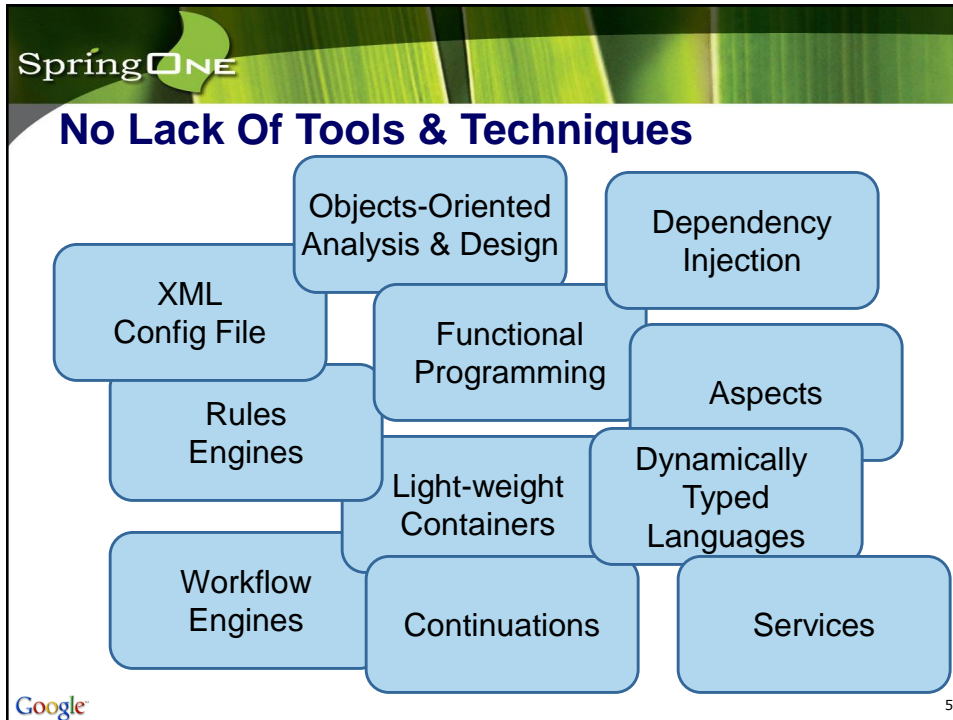
### Still Code Tends To Look Like This...


```
// This code is wrong. It should use the session
// manager instead of the parameter manager.
// [joe] I think this is no longer true
```

```
/**
 * Returns the customer's account. If there are
 * multiple accounts returns the oldest account.
 * TODO make this less confusing by better
 * supporting the notion of multiple accounts per
 * customer.
 */
Account getOnlyAccount() { ... }
```

```
/**
 * Performs a set of arcane checks such as
 * circles etc.
 */
```

Google 4






- 1. Understand your domain**
- 2. Choose your model(s)**
- 3. Choose a language to represent the model**
- 4. Map the model well to the language**
- 5. Protect Your Model**
- 6. Program for humans, not machines**

Google

7

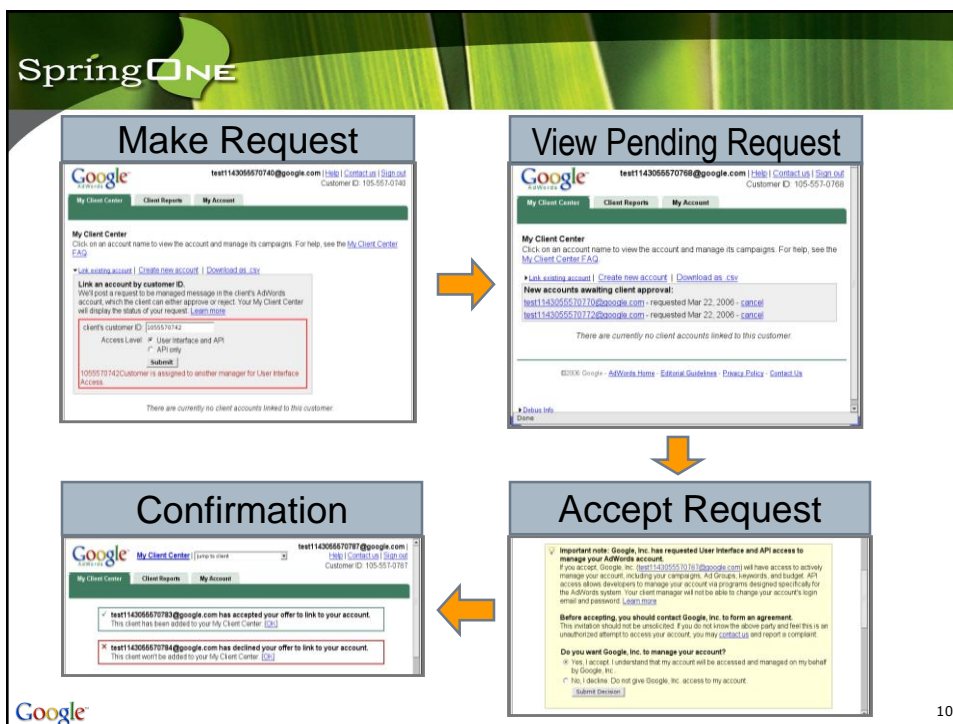
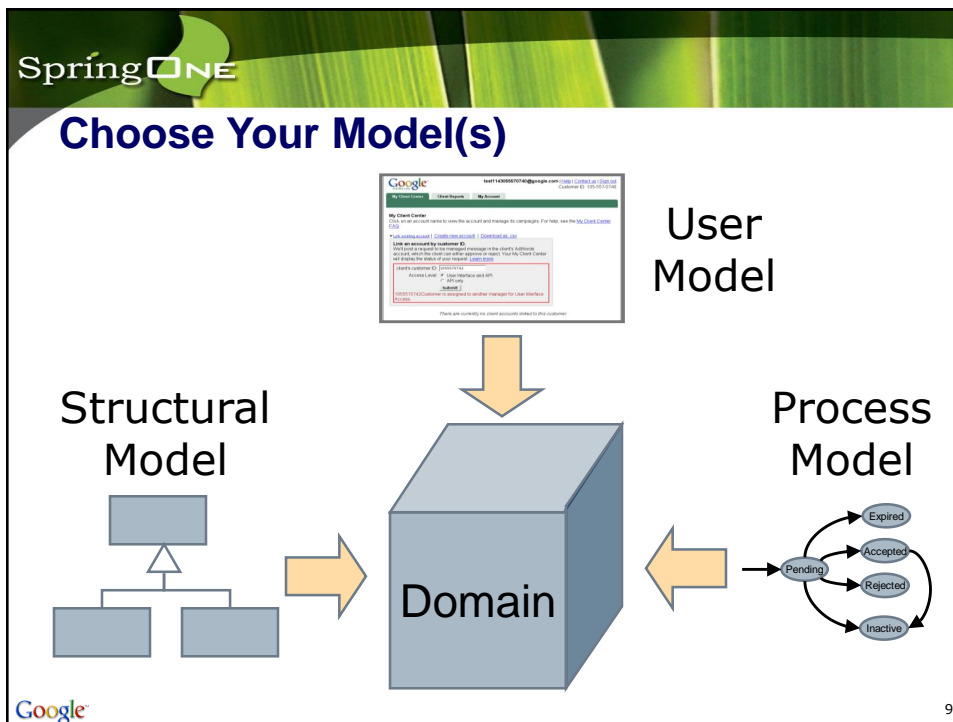


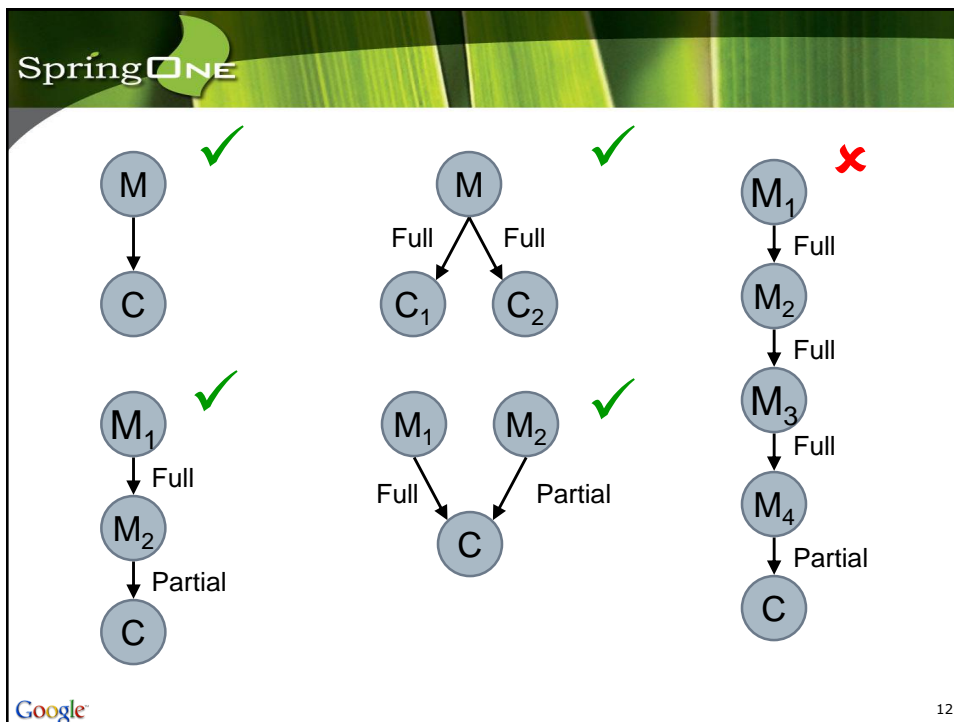
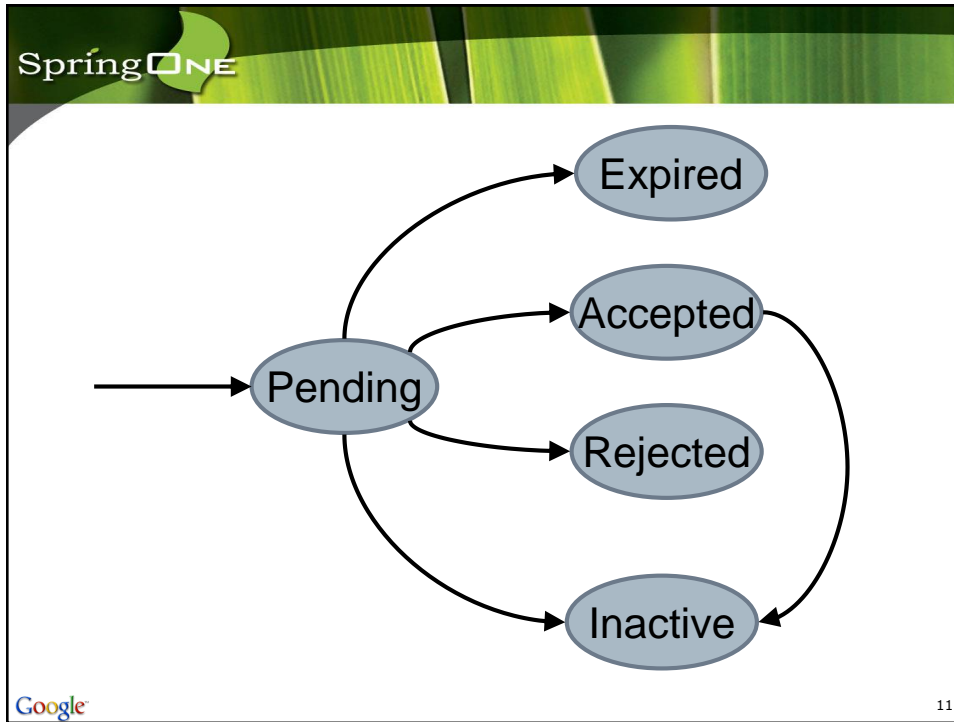
### **A Domain Of Managers And Clients...**

- **A manager can manage one or more clients**
- **A manager has to request access to a client's account first ("invite the client")**
- **A manager can request full access or partial access**
- **Subsequently, the client can accept or refuse**
- **A client can later revoke the manager's access**
- **A client can have a manager for each access type**
- **Managers can manage other managers, up to 3 levels deep**

Google

8





Spring ONE

## Choose A Language

1. A Language whose programming model matches the domain model
2. A general purpose language

```

graph LR
    PM[Process Model] --> Pending((Pending))
    Pending --> Expired((Expired))
    Pending --> Accepted((Accepted))
    Pending --> Rejected((Rejected))
    Pending --> Inactive((Inactive))
    subgraph Languages
        Java[Java]
        OSWorkflow[OSWorkflow]
        BPEL[BPEL]
    end
    PM --> Java
    PM --> OSWorkflow
    PM --> BPEL
  
```

Google

13

Spring ONE

## Flows In a Flow Language

```

<step id="1" name="Pending">
  <action id="1" name="Accept">
    <results>
      <unconditional-result step="2"/>
    </results>
  </action>
  <action id="2" name="Reject">
    <results>
      <unconditional-result step="3"/>
    </results>
  </action>
</step>
<step id="2" name="Accepted">
<step id="3" name="Rejected">
  
```

Google

14



Spring ONE

## Programming Flows In An OO Language

```
seqActivity1 = new SequenceActivity();
seqActivity1.Activities.Add(
    ReceiveRequest);
seqActivity1.Activities.Add(
    ProcessRequest);
seqActivity1.Activities.Add(
    SendConfirmation);
seqActivity1.Name = "sequenceActivity1";
```

```
graph LR;
    A[Receive Request] --> B[Process Request];
    B --> C[Send Confirmation];
```

Google 15

Spring ONE

## Language Trade-Offs

- Switching cost can be high
- Language workbenches try to solve this
- Don't forget your favorite tools
  - Debugger
  - Refactoring
  - Version Control Integration

Google 16



Spring ONE

## Map The Model Well To The Language

- "Invitations expire 30 days after the end of the month in which they were made"

```

TimeZone zone = TimeZone.getTimeZone("Universal");
Calendar calendar = Calendar.getInstance(zone);
calendar.set(Calendar.YEAR, year);
calendar.set(Calendar.MONTH, month-1);
calendar.set(Calendar.DATE, day);
calendar.set(Calendar.HOUR_OF_DAY, 0);
calendar.set(Calendar.MINUTE, 0);
...

```

- Did I say anything about time zone? Hours?
- Month - 1 ??

Google 17

Spring ONE

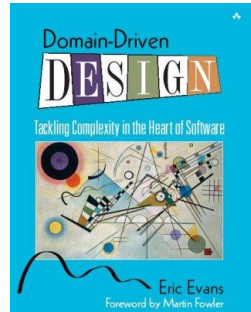
## March Is Not A Number

```

CalendarDate invited = CalendarDate.from(2006, 3, 23);
int allowedDays = 30;
CalendarDate expiryDate =
    completion.month().end().plusDays(allowedDays);

```

- month() is a CalendarInterval (a range of days, not a number)
- end() is a CalendarDate (a specific day, not a point in time)



Google 18

Spring ONE

## Protect Your Model

- No longer a code issue
- Make it beautiful but accessible
- Anticorruption layer
- Validation tools, e.g. imports etc.
- Good test coverage can actually hurt

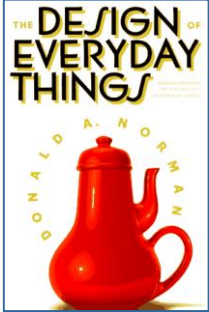
Google

19

Spring ONE

## Program For Humans Not Machines

- People will read your code
- Make it easy for them
- Misunderstandings are not the user's but the designer's fault
- Usability test your interfaces
- Learn from usability design: affordances,...



Google

20




## "Programmers Are People, Too"

- **Kevlin Henney**
  - "Effective Interface Design"
  - Affordances (Don Norman's Design of Everyday Things)
- **Josh Block**
  - "How to Design a Good API and Why it Matters"
  - "Conceptual Weight" of an API
- **Ken Arnold**
  - "Programmers are people, Too"




21




*"Write code worth reading"*


-- Ward Cunningham



22



**Thank You!**



23

The slide features a decorative header with a green leaf pattern and the text "Spring ONE". The main content is the phrase "Thank You!" in a large, bold, blue font. The footer includes the Google logo on the left and the number "23" on the right.

