Advanced Software Engineering Techniques

Course 3 – 17 October 2016

Adrian Iftene
adiftene@info.uaic.ro
Content

- Recapitulation
  - Swebok
  - SCM – Revision control

- Software Engineering Tools – Examples

- Model Driven Development
  - Model Driven Architecture
  - Agile MDD

- Test Driven Development

- Domain Specific Language
  - Eclipse Modeling Framework
R – Swebok project

- **Software Engineering Body of Knowledge**
- Book’s authors Alain Abran, James W. Moore, 2004
- 10 Knowledge Areas:
  - Software configuration management
  - Software engineering management
  - Software engineering process
  - Software engineering tools and methods
R – SCM – Revision control

- The management of changes to documents, programs, and other information stored as computer files
- Version control systems: Microsoft Word, OpenOffice.org, Drupal, Joomla, WordPress, MediaWiki
- Vocabulary: trunk, branch, change, checkout, checkin, merge, conflict
R – SE Tools and Methods

- Requirements
- Software design
- Software construction
- Software testing
**SET – Requirements – IBM Rational**

![Image of Rational RequisitePro interface]

### Table: Business Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
<th>Priority</th>
<th>Status</th>
<th>Cost</th>
<th>Difficulty</th>
<th>Stability</th>
<th>Assigned To</th>
<th>Location</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRULE1</td>
<td>Limit is $1000 a day (24h window)</td>
<td>Medium</td>
<td>Approved</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Database</td>
<td>Withdraw Money</td>
<td>Withdraw Money</td>
</tr>
<tr>
<td>BRULE2</td>
<td>No more than three withdrawals a day (24h window)</td>
<td>Medium</td>
<td>Approved</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Database</td>
<td>Withdraw Money</td>
<td>Withdraw Money</td>
</tr>
<tr>
<td>BRULE3</td>
<td>Account balance must be greater than $0</td>
<td>Medium</td>
<td>Approved</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Database</td>
<td>Withdraw Money</td>
<td>Withdraw Money</td>
</tr>
</tbody>
</table>

**Business rule 44 addresses a 'what-if' scenario.**
SET – Software design – ArgoUML, IBM RSA
SET – Software construction
SET – Software testing tools

- QA Center Performance, QuickTest Professional (QTP), WebLoad Analyzer, Unified TestPro, IBM Rational Tester
Content

» Recapitulation
  ◦ Swebok
  ◦ SCM – Revision control

» Software Engineering Tools – Examples

» Model Driven Development
  ◦ Model Driven Architecture
  ◦ Agile MDD

» Test Driven Development

» Domain Specific Language
  ◦ Eclipse Modeling Framework
Model Driven Development (MDD) is a paradigm for writing and implementing computer programs **quickly, effectively and at minimum cost**

MDD is an approach to software development where extensive models are created before source code is written

A primary example of MDD is the Object Management Group (OMG)’s Model Driven Architecture (MDA) standard
MDA is a software design approach for the development of software systems. It provides a set of guidelines for the structuring of specifications, which are expressed as models.

MDA is a kind of domain engineering, and supports model-driven engineering of software systems. It was launched by the Object Management Group (OMG) in 2001.
MDA

- MDA approach defines system functionality using a platform-independent model (PIM) using an appropriate domain-specific language (DSL)
- PIM is a model of a software system or business system, that is independent of the specific technological platform used to implement it
- DSL is a programming language or specification language dedicated to a particular problem domain, a particular problem representation technique, and/or a particular solution technique
- PSM – the code is generated by transforming the PIM model into a platform-specific model (PSM)
MDA approach

- OMG focuses Model-driven architecture on forward engineering

- One of the main aims of the MDA is to separate design from architecture (allows system developers to choose from the best and most fitting in both domains)

- The design addresses the functional (use case) requirements while architecture provides the infrastructure
MDA tools

- **Creation Tool**: used to elicit initial models
- **Analysis Tool**: used to check models for completeness, inconsistencies, or error and warning conditions
- **Transformation Tool**: used to transform models into other models
- **Composition Tool**: used to compose several source models
- **Test Tool**: used to “test” models
- **Simulation Tool**: used to simulate the execution of a system
- **Metadata Management Tool**: intended to handle the general relations between different models
- **Reverse Engineering Tool**: intended to transform particular legacy or information artifact portfolios into full-fledged models
**CA Gen**

- **CA Gen** is a powerful model-driven environment that will enable your organization to speed delivery and maintenance (platforms: z/OS (CICS and IMS), UNIX, Linux, Windows, .NET and J2EE)

![Mapping Application Development Approaches](image)
AMDD is the agile version of Model Driven Development (MDD)

MDD: The difference with AMDD is that instead of creating extensive models before writing source code you instead create agile models which are just barely good enough that drive your overall development efforts

AMDD is a critical strategy for scaling agile software development beyond the small, co-located team approach that we saw during the first stage of agile adoption
AMDD lifecycle

- Identify the high-level scope
- Identify initial "requirements stack"
- Identify an architectural vision

Modeling is part of iteration planning effort
- Need to model enough to give good estimates
- Need to plan the work for the iteration

Work through specific issues on a JIT manner
- Stakeholders actively participate
- Requirements evolve throughout project
- Model just enough for now, you can always come back later

Develop working software via a test-first approach
- Details captured in the form of executable specifications
Initial Requirements Modeling – identify high-level requirements and the scope of the release (what you think the system should do). We can use:

- a form of usage model (how users will work with the system),
- an initial domain model (identifies fundamental business entity types and the relationships between them)
- an initial user interface model (UI and usability issues)

Initial Architecture Modeling – identify an architecture that has a good chance of working
AMDD – Iteration Modeling

- Thinking Through What You’ll Do This Iteration

Each iteration implement the highest-priority work items

Each new work item is prioritized and added to the stack

Work items may be reprioritized at any time

Work items may be removed at any time

Copyright 2004-2007 Scott W. Ambler
Recapitulation
  ◦ Swebok
  ◦ SCM – Revision control

Software Engineering Tools – Examples

Model Driven Development
  ◦ Model Driven Architecture
  ◦ Agile MDD

Test Driven Development

Domain Specific Language
  ◦ Eclipse Modeling Framework
When you code, alternate these activities:

- add a test, get it to fail, and write code to pass the test
- remove duplication

This inner loop pumps the outer loops of Extreme Programming – Continuous Integration, Daily Deployment, Frequent Releases, and Steering Software Projects.
1. Add a test
2. Run all tests and see if the new one fails
3. Write some code
4. Run the automated tests and see them succeed
5. Refactor code
Repeat
TDD – Benefits & Vulnerabilities

Benefits:
- programmers that wrote more tests tended to be more productive
- It allows a programmer to focus on the task at hand as the first goal is to make the test pass
- total code implementation time is typically shorter
- the code is modularized, flexible, and extensible

Vulnerabilities
- is difficult to use in situations where full functional tests are required to determine success or failure
- The high number of passing unit tests may bring a false sense of security and productivity
Content

- Recapitulation
  - Swebok
  - SCM – Revision control
- Software Engineering Tools
- Model Driven Development
  - Model Driven Architecture
  - Agile MDD
- Test Driven Development
- Domain Specific Language
  - Eclipse Modeling Framework
Domain-specific language

- **DSL** – is a programming language or specification language dedicated to a particular problem domain, a particular problem representation technique, and/or a particular solution technique.

- Creating a domain-specific language (with software to support it) can be worthwhile if the language allows a particular type of problems or solutions to them to be expressed more clearly than pre-existing languages would allow, and the type of problem in question reappears sufficiently often.

- The opposite is:
  - a general-purpose programming language, such as C or Java,
  - or a general-purpose modeling language such as UML.
DSL – Examples (1)

- Logo for children

- Spreadsheet formulas and macros
**DSL – Examples (2)**

- **SQL** for relational database queries
- **LINQ** – a series of language extensions

```csharp
1. public void Linq6()
2. {
3.     int[] numbers = { 5, 4, 1, 3, 9, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100};
5.     var numsPlusOne =
6.         from n in numbers
7.         select n + 1;
8.     Console.WriteLine("Numbers + 1:");
9.     foreach (var i in
11. }
```
YACC grammars for creating parsers

Regular expressions for specifying lexers

- `\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b`
- `<TAG\b[^>]*>(.*?)</TAG>`
- `^\[(\d{19|20})\d\d[- .](0[1-9]|1[012])[0-9]|3[01])$`

The Generic Eclipse Modeling System for creating diagramming languages
In design and implementation
- Visual diagramming language, such as those created by the Generic Eclipse Modeling System
- Programmatic abstractions, such as the Eclipse Modeling Framework

Programming tools
- functional language XSLT, specifically designed for transforming one XML graph into another

Unix shell scripts – a domain-specific language for data organization

MediaWiki templates – support the creation of page templates and inclusion by reference
Metacompilers - useful for generating parsers and code generators for domain specific languages.

TexLanguage - a typesetting language developed by Donald Knuth

- LaTeX, a macro package for Tex which provides higher-level abstractions (though still for general-purpose document formatting)
- BibTeX, a macro package for Tex which specializes in handling citations/bibliographies for scholarly works.
**DSL – Advantages and disadvantages**

- Some of the **advantages**:  
  - Domain experts can understand, validate, modify, and develop DSL programs  
  - Self-documenting code  
  - Enhance **quality, productivity**, reliability, maintainability, portability and **reusability**  
  - Domain-specific languages allow validation at the domain level
- Some of the **disadvantages**:  
  - **Cost** of learning a new language vs. its limited applicability  
  - **Cost** of designing, implementing, and maintaining a DSL  
  - Finding, setting, and maintaining proper scope  
  - **Difficulty** of balancing trade-offs between domain-specificity and general-purpose programming language constructs  
  - **Loss** of processor efficiency
EMF started out as an implementation of the Object Management Group’s (OMG) Meta Object Facility (MOF) specification.

EMF is a Java open source framework and code-generation facility for building tools and other applications based on a structured model.

EMF provides an efficient reflective API and allows you to work with dynamic, non-generated, models.
EMF

- EMF can generate Java source code that will allow you to create, query, update, serialize, deserialize, validate, and track changes to instances of your models.

- EMF supports generating code from XML Schema, UML class diagrams (Rational Rose or UML2), and annotated Java interfaces.
Example: Generating an EMF Model

Model generated looks like:

- Library
  - name : String

- <<enumeration>>
  - BookCategory
    - Mystery
    - ScienceFiction
    - Biography

- Writer
  - name : String
  - writers : 0..*

- Book
  - title : String
  - pages : int = 100
  - category : BookCategory
  - books : 0..*
  - author : 1
  - books : 0..*
GEMFM – Prerequisites

- EMF Updates sites: http://download.eclipse.org/modeling/emf/updates/releases/
- Eclipse → Help → Install new software
EMF in Eclipse

- Help -> About Eclipse -> Installation details -> Plug-ins

![Eclipse Plugin Details](image_url)

<table>
<thead>
<tr>
<th>Signed</th>
<th>Provider</th>
<th>Plug-in Name</th>
<th>Version</th>
<th>Plug-in Id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF - Eclipse Modeling Framework</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Capabilities Example</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.activities</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ant Tasks</td>
<td>2.4.0.v20090615</td>
<td>org.eclipse.emf.ant</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Cheat Sheets</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.cheatsheets</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Code Generation</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.codegen</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ecore Code Generator</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.codegen</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ecore Code Generator UI</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.codegen.ui</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Code Generation UI</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.ui</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ecore</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Common</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.common</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Common UI</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.common.ui</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Model Converter</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.converter</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Data Binding</td>
<td>1.1.0.v20090615</td>
<td>org.eclipse.emf.databinding</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Edit Data Binding</td>
<td>1.1.0.v20090615</td>
<td>org.eclipse.emf.datatransfer.data</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Documentation</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.doc</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ecore</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Change Model</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.change</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Change Edit</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.change.edit</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ecore Edit</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.edit</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Sample Ecore Editor</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.editor</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF XML/XML Persistence</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.xml</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ecore</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Change Model</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.change</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Change Edit</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.change.edit</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ecore Edit</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.edit</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Sample Ecore Editor</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.editor</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF XML/XML Persistence</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.xml</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ecore</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Change Model</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.change</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Change Edit</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.change.edit</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ecore Edit</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.edit</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Sample Ecore Editor</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.editor</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF XML/XML Persistence</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.xml</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ecore</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Change Model</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.change</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Change Edit</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.change.edit</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ecore Edit</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.edit</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Sample Ecore Editor</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.editor</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF XML/XML Persistence</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.xml</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ecore</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Change Model</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.change</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Change Edit</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.change.edit</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Ecore Edit</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.edit</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF Sample Ecore Editor</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.editor</td>
</tr>
<tr>
<td></td>
<td>Eclipse Modeling Project</td>
<td>EMF XML/XML Persistence</td>
<td>2.5.0.v20090615</td>
<td>org.eclipse.emf.ecore.xml</td>
</tr>
</tbody>
</table>

38
EMF – Import model from Rose (1)

EMF – Import model from Rose (2)

Select an option:
- Ecore model (MDL)
- Rose class diagram (MDL)
- XML Schema

Model Import

Select a 'mdl' file, load and specify any path map symbols, load the model, and choose a file name for the generator model.

File:
- Load

Path Map

Specify which packages to generate and which to reference from other generator models.

Generator model file name:
- library.genmodel

Package Selection

Package | File Name
---|---
org.eclipse.example.library | library.ecore

Referenced generator models:

Add...
EMF – Import model from Rose (3)
library.mdl

(object Class "Book"
  quid "40C81E700047"
  class_attributes (list
    (object ClassAttribute "title"
      quid "40C81E770119"
      type "String")
    (object ClassAttribute "pages"
      quid "40C81E7E03A3"
      type "int"
      initv "100")
    (object ClassAttribute "category"
      quid "40C81E890344"
      type "BookCategory")
  )
)
Generate the EMF Model Code (1)
Generate the EMF Model Code (2)
The result (1)
The result (2)
Example (2) – Webpage

- Eclipse 3.7 (Indigo), Eclipse 4.5 (Mars), Eclipse 4.6 (Eclipse Neon)
- Eclipse EMF Tutorial: [http://www.vogella.de/articles/EclipseEMF/article.html](http://www.vogella.de/articles/EclipseEMF/article.html)
Links

- AMDD: [http://www.agilemodeling.com/essays/amdd.htm](http://www.agilemodeling.com/essays/amdd.htm)
- [http://www.eclipse.org/modeling/emf/docs/2.x/tutorials/clibmod/clibmod_emf_2.0.html](http://www.eclipse.org/modeling/emf/docs/2.x/tutorials/clibmod/clibmod_emf_2.0.html)
- Eclipse EMF Tutorial: [http://www.vogella.de/articles/EclipseEMF/article.html](http://www.vogella.de/articles/EclipseEMF/article.html)
Bibliography

- **Swebok Book:**
  [http://se.sjtu.edu.cn/sites/se/gb/CCSE/Swebok_Ironman_June_23_%202004.pdf](http://se.sjtu.edu.cn/sites/se/gb/CCSE/Swebok_Ironman_June_23_%202004.pdf)

- **Benefits of MDD:**
Software Engineering Links (1)

- .Net
  - [http://dofactory.com/net/design–patterns](http://dofactory.com/net/design–patterns)
- Tim Downey – Web Development
- O’Reilly – Design Patterns
- Code smells: [https://sourcemaking.com/refactoring/extract-method](https://sourcemaking.com/refactoring/extract-method)
- CodeScool: [https://www.codeschool.com/](https://www.codeschool.com/)
Software Engineering Links (2)

- http://it-ebooks.info/
- https://sourcemaking.com/
- http://www.codeproject.com/
- https://www.safaribooksonline.com/library/view/design-patterns-explained/0321247140/
- http://www.growing-object-oriented-software.com/
- http://www.mkyong.com/
- http://www.tutorialspoint.com/
- http://www.oodesign.com/
- https://www.codecademy.com/
- https://www.thenewboston.com/
Software Engineering Links (3)

- Head First into Design Patterns, de la editura O'Reilly, scrisa de Eric Freeman si Elisabeth Robson.  
  https://www.amazon.com/Head-First-Design-Patterns-Freeman/dp/0596007124
- http://martinfowler.com/  
  https://www.tutorialspoint.com/index.htm
- https://drive.google.com/open?id=0B8hbVmS3aWf4WGZXTFpVUkpD Wms
- * Effective Java, Joshua Bloch  
  * Effective C++, Scott Meyers  
  * Thinking in Java, Bruce Eckel  
  * Thinking in C++, Bruce Eckel.
- 1. Effective Modern C++  
  2. Effective C++, Third Edition
- http://stackoverflow.com/  
  https://developer.android.com/  
  https://www.tutorialspoint.com/android
Software Engineering Links (4)

- Andrew Hunt and David Thomas, The Pragmatic Programmer, Addison Wesley, 1999
- Martin Fowler, Refactoring, Addison Wesley, 2000
- Tiberiu Covaci, et al., Microsoft Certification Toolkit, 2010
- Dominic Betts, et. al, Exploring CQRS and Event Sourcing, Microsoft, 2012
- Steve MacConnell, Code Complete 2, Microsoft, 2005
- Venkat Subramaniam and Andy Hunt, Practices of an Agile Developer, Pragmatic Bookshelf, 2006
- [https://www.coursera.org](https://www.coursera.org)
- [http://beginnersbook.com](http://beginnersbook.com)