Software Engineering at the crossroads

Le Génie Logiciel à la croisée des chemins

Work in progress

JBezivin@gmail.com
JBezivin@twitter
1. In search of the ultimate silver bullet
   1. Introduction
   2. Software Engineering 1.0
   3. How Model Driven Engineering Missed the Boat

2. From SE 1.0 to SE 2.0
   1. Problem and Solution Spaces
   2. Domain Engineering
   3. Support Engineering
   4. Conclusion

Santa Claus does not exist

There is NO silver bullet
Some bad news and some good news

INTRODUCTION
A View of 20th and 21st Century Software Engineering

Barry Boehm, ICSE2006, Shanghai
Paradigm/Artifact changes \{step = 15y.\}


Procedural Technology  Object Technology  Component Technology  Model Driven Engineering

Procedures, Pascal, C, ...  Objects, Classes, Smalltalk, C++, ...  Components, Packages, Frameworks, Patterns, EJB, J2EE  Models, Metamodels, UML, MOF, ...

Procedural refinement  Object composition  Model transformation
Climbing the steps
Software engineering: Approaching half-time?

1965 2015 2065

Structured Programming
Object Oriented Programming
Agile Development

Jeff Sutherland: Scrum (OOPSLA95)

http://bertrandmeyer.com/2013/04/04/the-origin-of-software-engineering/
ThoughtWorks, Technology Radar, May 2013

Languages & Frameworks

**Adopt**
- Clojure
- CSS frameworks
- Jasmine paired with Node.js
- Scala
- Sinatra

**Trial**
- CoffeeScript
- Dropwizard
- HTML5 for offline applications
- JavaScript as a platform
- JavaScript MV* frameworks
- Play Framework 2
- Require.js & NPM
- Scratch, Alice, and Kodu

**Assess**
- ClojureScript
- Gremlin
- Lua
- Nancy
- OWIN
- RubyMotion
- Twitter Bootstrap

**Hold**
- Backbone.js
- Component-based frameworks
- Handwritten CSS
- Logic in stored procedures

http://www.thoughtworks.com/radar (Martin Fowler & Co°)
Hype after Hype

- Are we condemned to jump from hype to hype like a fashion industry? (1)
- What is the hidden meaning (if any) in the evolution of our discipline?

(1) Ivar Jacobson
Looking at the past to guess the future

5 years 5 years

50 years 50 years
Software Engineering is not in good health

SOFTWARE ENGINEERING 1.0
Not dead, but at least critically ill

The NATO Conferences of 1968 and 1969 were motivated by the belief that software development should be "based on the types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering."

Surprisingly the conferences did not discuss what these foundations and disciplines were, or how they could be emulated. There has been little discussion of this topic in the intervening forty years and more. Some important lessons have been neglected.

From Michael Jackson’s Web site

Software engineering is gravely hampered today by immature practices:

- The prevalence of fad's more typical of fashion industry than of an engineering discipline
- The lack of a sound, widely accepted theoretical basis
- The huge number of methods and method variants, with differences little understood and artificially magnified
- The lack of credible experimental evaluation and validation
- The split between industry practice and academic research

The “New Deal” for Software development
What has changed in the past 50 years?

- Expressions like “CAD” or “Computer Assisted” or “Computer Aided” have lost all their discriminant meaning in engineering.
- Most engineering fields are now using computers and software.
- Time to adapt our vision.
“Software Professionals” vs. “End User Developers”

In advanced countries, about 1% of the population are software professionals (i.e. people that make a living from software production and management)

315 millions habitants in USA

3,3 millions software professionals

315,091,138

Software Professionals and End Users

Software Professionals (1%)

End Users, including End User Developers (99%)

Total inversion:
Percentage of non software professional using computers and software applications at work, home, etc.
Software vs. Traditional Engineers


Will have to imagine new ways for working together in the next decades

<table>
<thead>
<tr>
<th>Role</th>
<th>#</th>
<th>SE Title</th>
<th>#</th>
<th>TE Title</th>
<th>Ratio of SE to TE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Practitioners</td>
<td>611,900</td>
<td>Software Engineers</td>
<td>1,157,020</td>
<td>Traditional Engineers</td>
<td>53%</td>
</tr>
<tr>
<td>Managers</td>
<td>264,790</td>
<td>Computer and Information Systems Managers</td>
<td>413,750</td>
<td>Engineering Managers + Construction Managers</td>
<td>64%</td>
</tr>
<tr>
<td>Educators</td>
<td>16,495</td>
<td>Computer Science (practical)</td>
<td>29,310</td>
<td>Engineering Teachers</td>
<td>56%</td>
</tr>
<tr>
<td>Technicians</td>
<td>457,320</td>
<td>Computer Programmers</td>
<td>516,170</td>
<td>Engineering Technicians</td>
<td>88%</td>
</tr>
</tbody>
</table>
Workload Transfer due to Excel

3 M Software Professionals (1%)

Workload transfer (1979-2013)

300 M End Users (99%)
Including 55M Excel users (about 18%)
Everybody’s a programmer

Visicalc (1979)

Excel (much later)

(...Charles Simonyi...)

(...Intentional Programming...)

www.bricklin.com
Model Driven Engineering

The last silver bullet fired blank

HOW MDE PARTLY MISSED THE BOAT
Separating the platform independent and dependent parts of a system (PIM/PSM)

We don't want anymore to pay such a high price for simply moving our information system to a new middleware platform (COM, CORBA, Java, HTML, XML, DotNet, etc.) when our business system stays stable. We are prepared to pay a last price for building the abstract models of our business and services that will guarantee us against technological obsolescence. From there, any platform provider will also have to provide the mapping solutions from standard business models before we buy.

November 2000
The first promise/commitment of MDA™ was on sustainability:

- “Developers gain the ultimate in flexibility, the ability to regenerate code from a stable, platform independent model as the underlying infrastructure shifts over time”.
- “ROI flows from the reuse of application and domain models across the software lifespan--especially during long-term support and maintenance, the most expensive phase of an application's life”.
- “Durability”, “Perennity”

The MDA™ did not deliver on sustainability.

Reasons are multiple:

- Complexity of UML
- Evolution of UML (versions)
- UML profiles
- UML tools not based on the UML metamodel (no dogfooding)
- Bad interoperability of UML tools
- Versions of XMI

As a result, Java code is probably more sustainable than most UML models

- In direct violation of initial PIM/PSM separation objective (MDA)
Any artifact or situation in IS management may be precisely described and operated by models/metamodels

- Product & Process
- Code & Data
- Problem & Solution
- Static & Dynamic
- PIM & PSM (& CIM)
- Primitive & Derived
- Executable & Non executable
- Proprietary & Normative
- Atomic & Composite
- Basic & Correspondence & Transformation
- Visual & Textual
- Descriptive & Prescriptive
- Formal & Informal

- Requirement models
- Product line models
- Feature models
- Process & agent models
- Trace models
- Object & Component models
- Service models
- Complex event models
- Legacy models
- Software architecture models
- Enterprise architecture models
- Modeling in the large & modeling in the small
- etc.

Huge productivity gains possible with this homogeneous representation scheme
But we learnt many things from MDE

1. Representation principle
   - Any model $M$ represents a system $S$

2. Multiple view principle
   - A system $S$ may be represented by several models

3. Conformance principle
   - Any model $M$ conforms to the language of its metamodel $MM$

4. 3-level principle
   - Any metamodel $MM$ conforms to a common metamodel $MMM$

5. Transformation principle
   - The most important operation applicable to models is a transformation

6. HOT principle
   - A transformation is a model

7. Weaving principle
   - Abstract correspondences between models are represented as models

8. Megamodel principle
   - Model elements may be considered as models

9. Unification principle
   - All models specialize a common abstract model

10. Technical Space Framework
    - Any model has a given representation defined by its technical space (no MOF/ECORE lock-in)
Not all models are software models, but most of them are
ME meets OSS

- The Normative period (1996-2004)
- The Open Source period (2004-2010)

Mission
The Eclipse Modeling Project will focus on the evolution and promotion of model-based development technologies within the Eclipse community. It will unite projects falling into this classification to bring holistic model-based development capabilities to Eclipse.

DTC = Domain Technical committee
PTC = Platform Technical committee
The long history of modeling languages

No global **consolidated** history of Modeling Languages

(Non-exhaustive list)

### Programming Languages
- Assembler
- Fortran
- COBOL
- PL/1
- Prolog
- Ada
- Smalltalk
- Algol60
- Pascal
- C
- C++
- Java
- C#
- Ruby
- Python
- Javascript
- F#
- Scala
- Go
- Dart

### Modeling Languages
- Sara
- PSR/PSA
- SREM
- SADT
- Petri
- SART
- JSD
- DFD
- VDM
- B
- Z
- OMT
- UML
- SBVR
- SysML

*DS*
Definition Framework

- **Model Engineering (ME)** promotes the systematic use of models, metamodels and model transformations to achieve industrial goals.
- **Model Driven Engineering (MDE)** is the application of ME principles and tools to any given engineering field.
- **Model Driven Software Engineering (MDSE)**
- **Model Driven Development (MDD)**
- **Model Driven Code Generation (MDCG)**
- **Model Driven Reverse Engineering (MDrevE)**
- **But also**
  - Model Driven Business Engineering (MDbizE)
  - Model Driven System Engineering (MDsysE)
  - Model Driven Data Engineering
  - Model Driven Web Engineering
  - Model Driven Requirement Engineering
  - Model Driven Civil Engineering
  - Model Driven Biological Engineering
  - etc.
MDE is not only for code generation

Initially MDA was for just software engineering, But the scope was progressively extended

- Software engineering
- Data engineering
- System engineering
- Business engineering
- Enterprise engineering
- Telecommunication engineering
- Building engineering
- Electrical engineering
- Mechanical engineering
- Automotive engineering
- Aeronautical engineering
- Biological engineering
- Automotive engineering
- Health engineering
- Financial engineering
- etc.

Broadening application spectrum
A possible scenario for MDE

Visibility

Technology trigger

2010 2020 2030 2040

Time

Second tentative

WE ARE HERE

MDE is too important to be confined to pure software engineering
Beyond technical spaces

PROBLEM AND SOLUTION SPACES
Focus on Engineering

Scientists study the world as it is; engineers create the world that has never been.

Theodore von Kármán
The two engineering spaces

Problems lie here

Domain Engineering

Tools to solve problems may be found here

Support Engineering
Problems and Solutions

- **Support Engineering (vertical?)**
  - Process engineering
  - Product (line) engineering
  - Software language engineering
  - Model engineering
  - Service engineering
  - Data engineering
  - Program engineering
  - Event engineering
  - Constraint engineering
  - System engineering
  - Requirement engineering
  - Ontology engineering
  - OSS engineering

- **Domain Engineering (horizontal?)**
  - Civil engineering
  - Building engineering
  - Electrical engineering
  - Mechanical engineering
  - Business engineering
  - Biological engineering
  - Automotive engineering
  - Health engineering
  - Enterprise Engineering
Problem Spaces

DOMAIN ENGINEERING
Old and New engineering fields

- Software engineering
- Electrical engineering
- Civil engineering
- Chemical engineering

- Financial engineering
- Bioengineering
- Aerospace engineering
Domain Engineering

Problem spaces

Solution spaces (Support Engineering)

Domain Engineering

Product Engineering

Process Engineering
Many features common to all domain engineering fields

- Based on support engineering
  - Product & Process focus
  - Including HR and team management
    - Human in the loop
    - Engineers in control

- Chain
  1. Building Abstract Models
  2. Verification/Validation
  3. Putting in Production
  4. Putting in Operation

- Need for a strong model repository (e.g. Dassault Syst. Catia)
  - Scaling up to millions of parts
  - Cooperative concurrent access
  - Point of view mechanisms
  - Strong zooming mechanisms
Electrical Engineering

1. Building abstract models
2. Validation Verification
3. Putting in Production
4. Augmenting, Changing the world
Construction Engineering

Building abstract models → Validation Verification → Putting in Production → Augmenting, Changing the world
Complexity of the Domain Engineering Landscape

- Civil Engineering
- Electrical Engineering
- Automotive Engineering
- Architecture Engineering
- Medical Engineering
- Chemical Engineering
- Biological Engineering
- Telephone Engineering
- Military Engineering
- Financial Engineering
- Business Engineering
- Enterprise Engineering
- Ecology Engineering
- Agricultural Engineering
- Communication Engineering
- Other Engineering Fields
Transfer of expertise between engineering fields

Architectural engineering

Software engineering
Beyond Technical Spaces

SUPPORT ENGINEERING
Specialized engineering fields

- Language Engineering
  - Software Language Engineering
    - Grammar Engineering
    - Model Engineering
    - Ontology Engineering
    - XML Engineering
Complexity of the Support Engineering Landscape

- Language Engineering
- Program Engineering
- Ontology Engineering
- Model Engineering
- Web Engineering
- Service Engineering
- Transformation Engineering
- Rule Engineering
- Complex Event Engineering
- Data Engineering
- Process Engineering
- Product Engineering
- HR Engineering
- Team Engineering
- Software Engineering
- OSS Engineering
Process engineering encompasses a vast range of industries, such as chemical, petrochemical, mineral processing, advanced material, food, pharmaceutical, biotechnological, and software industries.

See also Concurrent Engineering
Team and Product management

- Team Management Engineering
  - Software Team Management Engineering
    - Agile Methods
  
- Product Lifecycle Management (PLM)
  - Product Line Engineering (incl. variability)
  
- Software Product Line Engineering
Data Engineering

- Open Data
- Linked Data
- Data Viz
- Data Mining
- etc.
Program Engineering

- Short name: “programming”
- Long tradition of excellence
- Noble and visible part of SE
- Very difficult
- Many iterations and branches
  - Structured Programming
  - OO Programming
  - Functional Programming
  - Etc.

- Good definitions allow avoiding sterile, futile, and non productive discussions

«Mal nommer les choses, c'est ajouter au malheur du monde» Albert Camus
[To misname things is to add misery to the world]
Composite Engineering Fields

- Software Engineering
  - Program Engineering
  - Model Engineering
  - Language Engineering
  - Method Engineering
  - Etc.

But also:
- OSS Engineering
- Document Engineering
- Requirement Engineering
- Formal Method Engineering
- Usability Engineering
- HR Engineering
- Education Engineering
- Team Mgmt Engineering
- Legal Engineering
DogFooding: Software Tools are Software Too
Software Engineering is Engineering

CONCLUSIONS
Towards a Unified Global Theory of Engineering

“Predictions are very hard, especially about the future”

- Yes we need to resurrect Software Engineering.
  - The expression “Software Engineering” was coined in 1965.
  - Need to refund SE2.0 on solid grounds, taking into account what has been learnt in half-a-century.

- The second part of the life of SE (2015-2065) will probably be more in rupture than in continuity

- Change of focus: from “Software Engineering” to “Engineering Software”?
  - Wikipedia: “Software engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the design, development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software”
  - More relevant is the increasing need for the application of software to engineering
MDE is dead, Long life MDE
Thanks

• Questions?
• Comments?

JBezivin@gmail.com
JBezivin@twitter

« Qu'on ne me dise pas que je n'ai rien dit de nouveau : la disposition des matières est nouvelle ...»
(Pascal, Pensées, 1669)

[Do not tell me that I did not say anything new: arrangement of the material is new]
Teaching Kids to Program?

3 M Software Professionals (1%)

Any significant workload transfer?

300 M End Users (99%)

A false good solution?