

# Méthodes de conception et de validation de logiciel (DUGL)

**Mathieu Acher**

<http://www.mathieuacher.com>

*Associate Professor*

*University of Rennes 1*

# Disclaimer

- Supports en Français / Anglais
  - <http://www.mathieuacher.com/teaching/DUGL>
- Je suis enseignant-chercheur en génie logiciel / software engineering
  - Comment synthétiser des variantes?

# Objectifs

- Méthodes de conception et validation de logiciel
  - En fait: génie logiciel / software engineering
  - Comment développer des systèmes logiciels de plus en plus complexe?
- #1 Prendre conscience de la complexité des systèmes logiciels actuels et à venir
  - Les enjeux et l'impact sur le métier
- #2 Modélisation
  - UML, SysML
- #3 Design patterns, refactoring, test
  - OO avancé
- #4 Méthodes

# Modalités d'Evaluation

- Contrôle continu
  - Basé sur le contenu du cours et des TDs
- Projet
  - Mise en pratique des technique de modélisation avec UML, design patterns, tests, refactoring, etc.



# Aujourd'hui / Today

- Méthodes de conception et de validation de logiciel
  - En fait: génie logiciel / software engineering
  - Comment développer des systèmes logiciels de plus en plus complexe?
- **#1 Prendre conscience de la complexité des systèmes logiciels actuels et à venir**
  - Les enjeux et l'impact sur le métier

# Outline

---

- ① Issues in Software Engineering
- ② Evolution in Software Engineering
- ③ State of the Practice
- ④ Modeling in Software Engineering

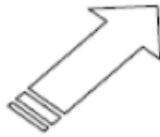
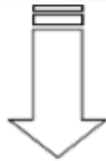
# Outline

---

- ① Issues in Software Engineering
- ② Evolution in Software Engineering
- ③ State of the Practice
- ④ Modeling in Software Engineering

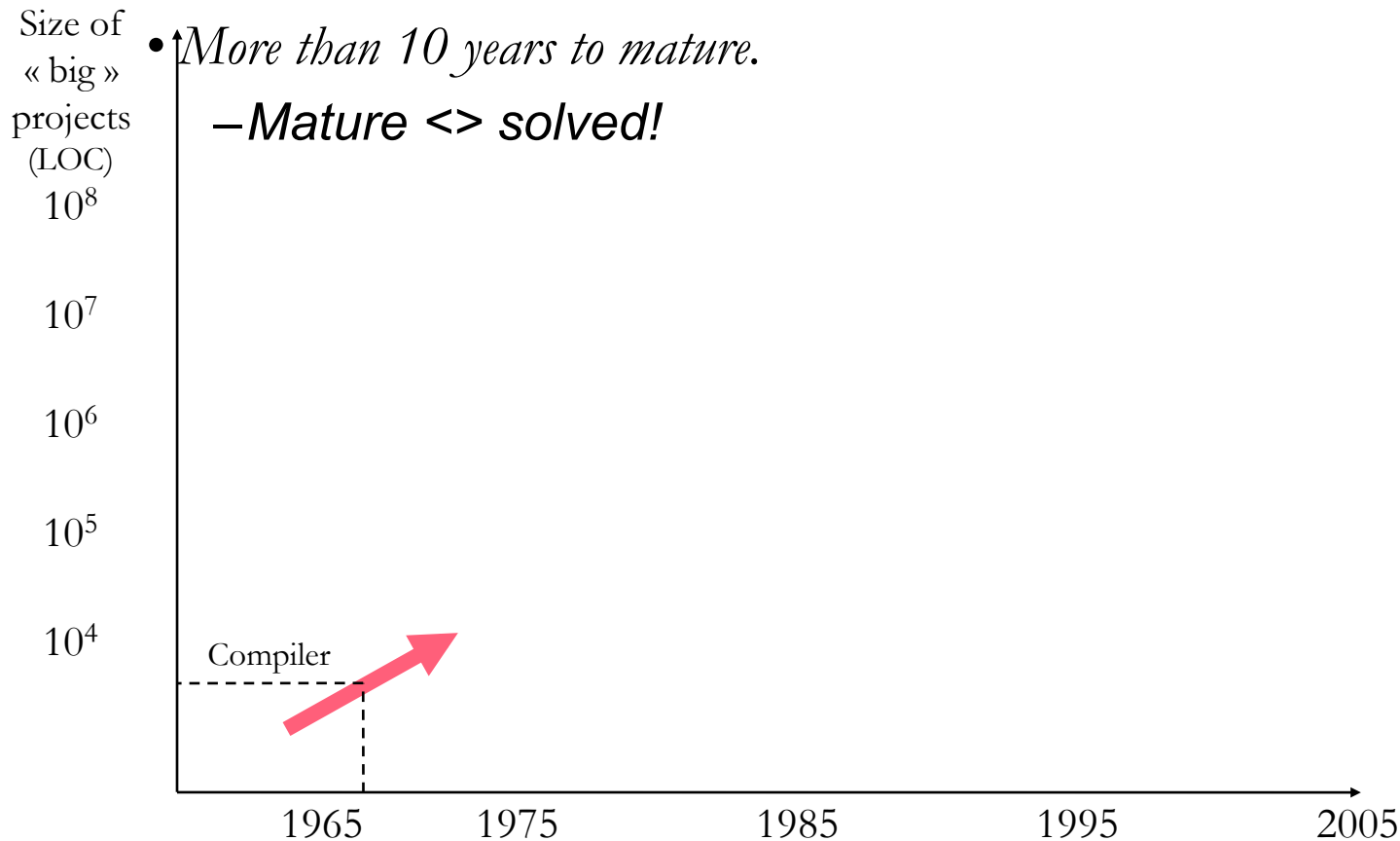
# Software Complexity

---



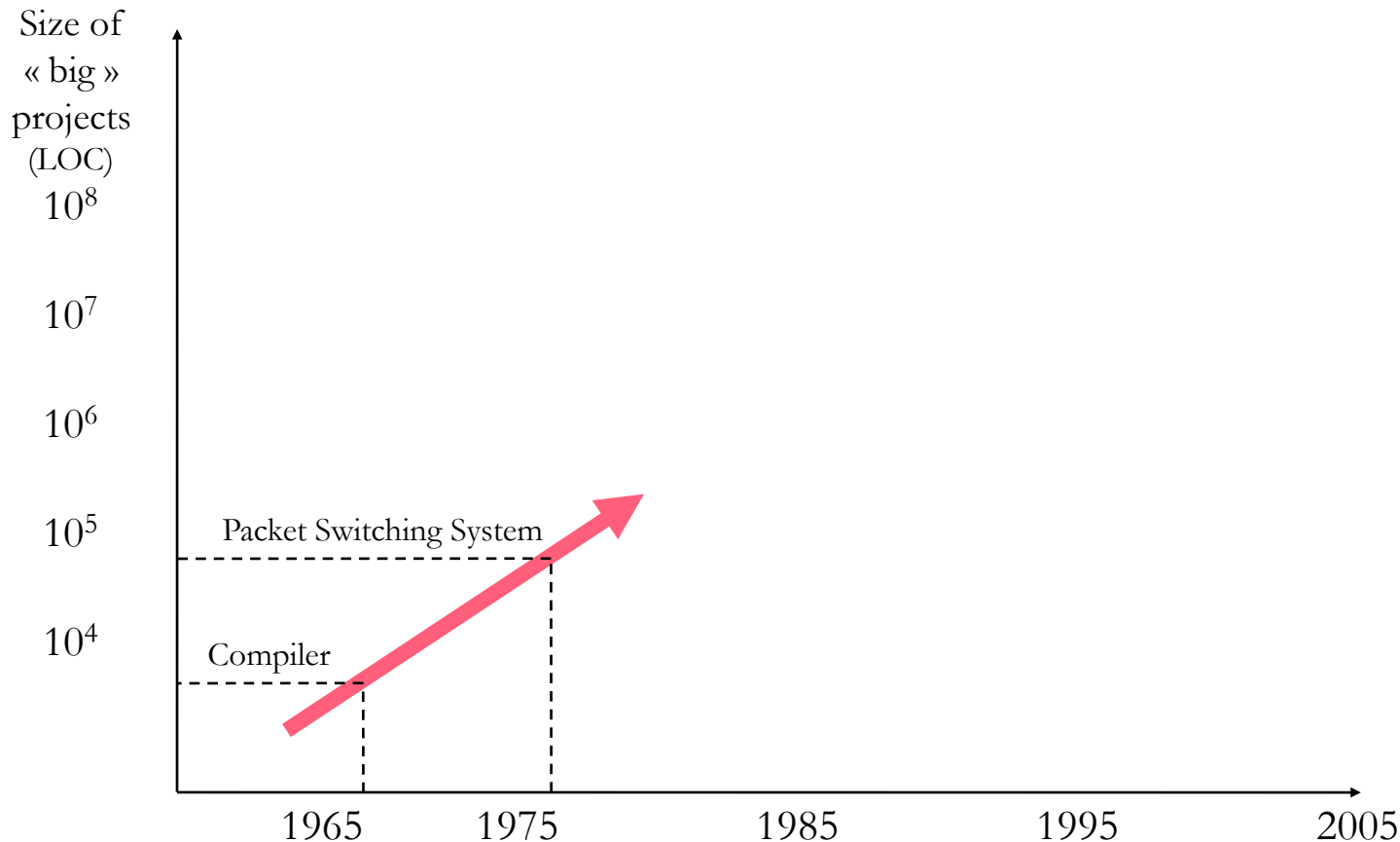
# Problems addressed in SE

- 1960's: Cope with inherent complexity of software (Correctness)
  - Milestone: Floyd 'assigning meaning to programs'



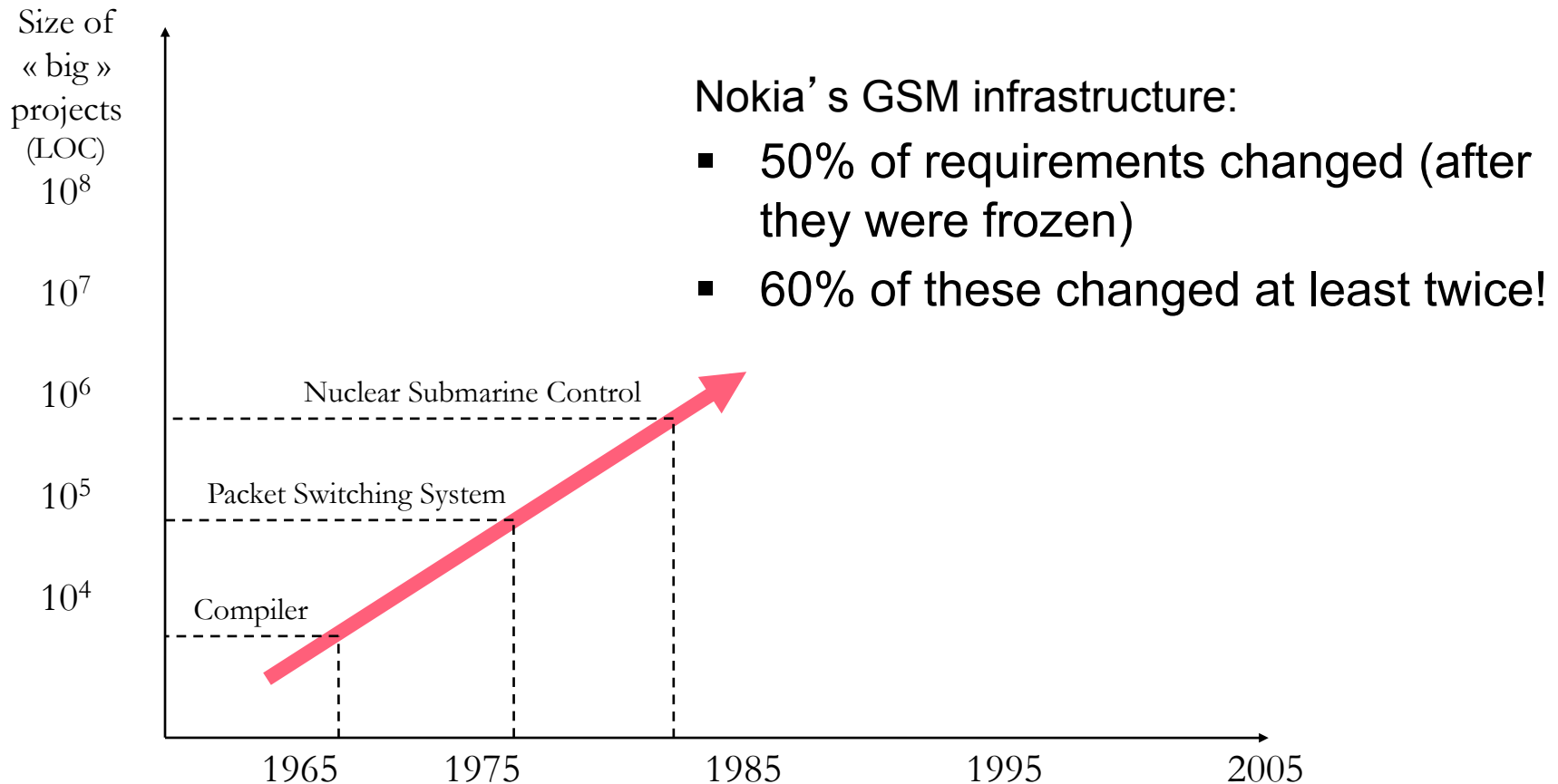
# Problems addressed in SE

- 1970's: Cope with project size
  - Milestone: Parnas, Yourdon: *modularity & structure*
    - *More than 10 years to mature*



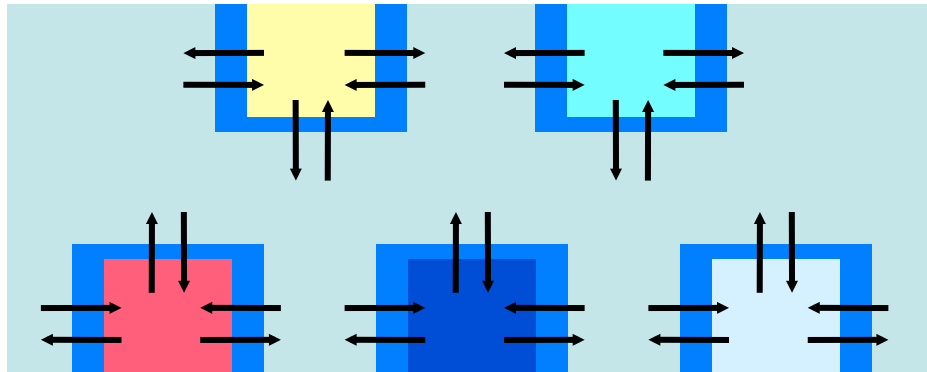
# Problems addressed in SE

- 1980's: Cope with variability in requirements
  - Milestone: Jackson, Meyer: *modeling, object orientation*
    - *More than 10 years to mature*



# OO approach: frameworks

---

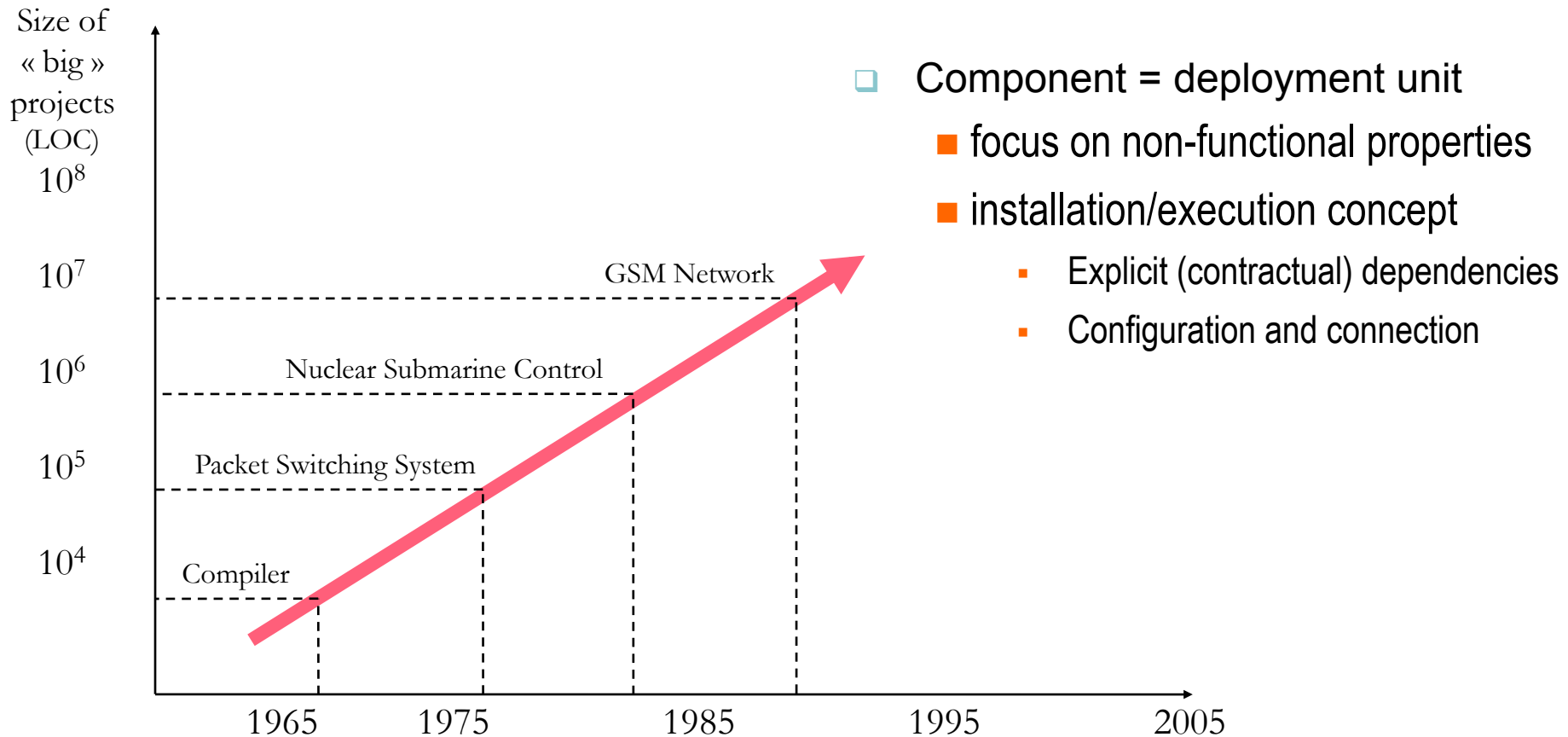




# Problems addressed in SE

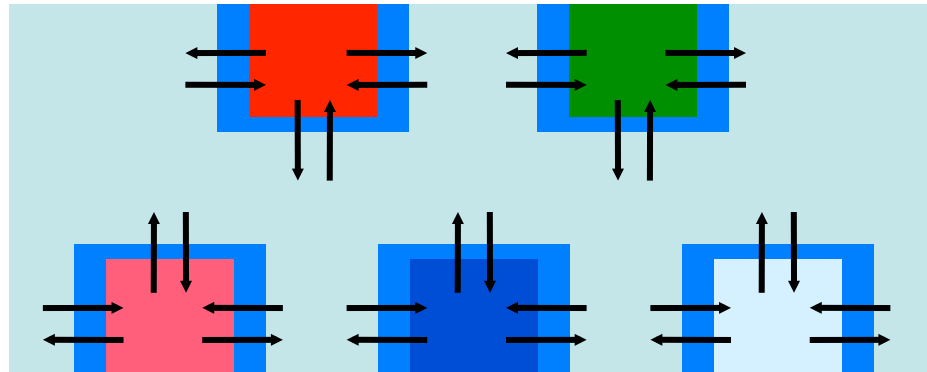
- 1990's: Cope with distributed systems and mass deployment:

- Milestone: MS (COM), Szyperski: *product-lines & components*



# OO approach: Models and Components

---



- Frameworks

- Changeable software, from distributed/unconnected sources even after delivery, by the end user
- Guarantees ?

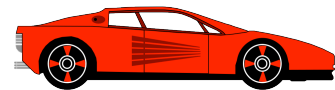
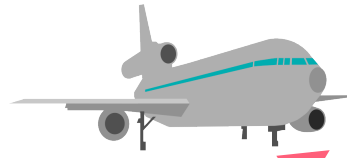
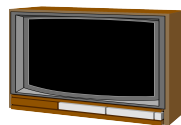
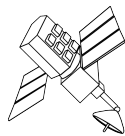
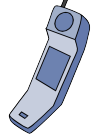
Functional , synchronization, performance, QoS

# Problems addressed in SE

- 2000's: pervasive software integration, accelerating technological changes (platforms)

■ Milestone: ?

Size of  
« big »  
projects  
(LOC)  
 $10^8$



Windows 2000

GSM Network

Nuclear Submarine Control

Packet Switching System

Compiler

1965

1975

1985

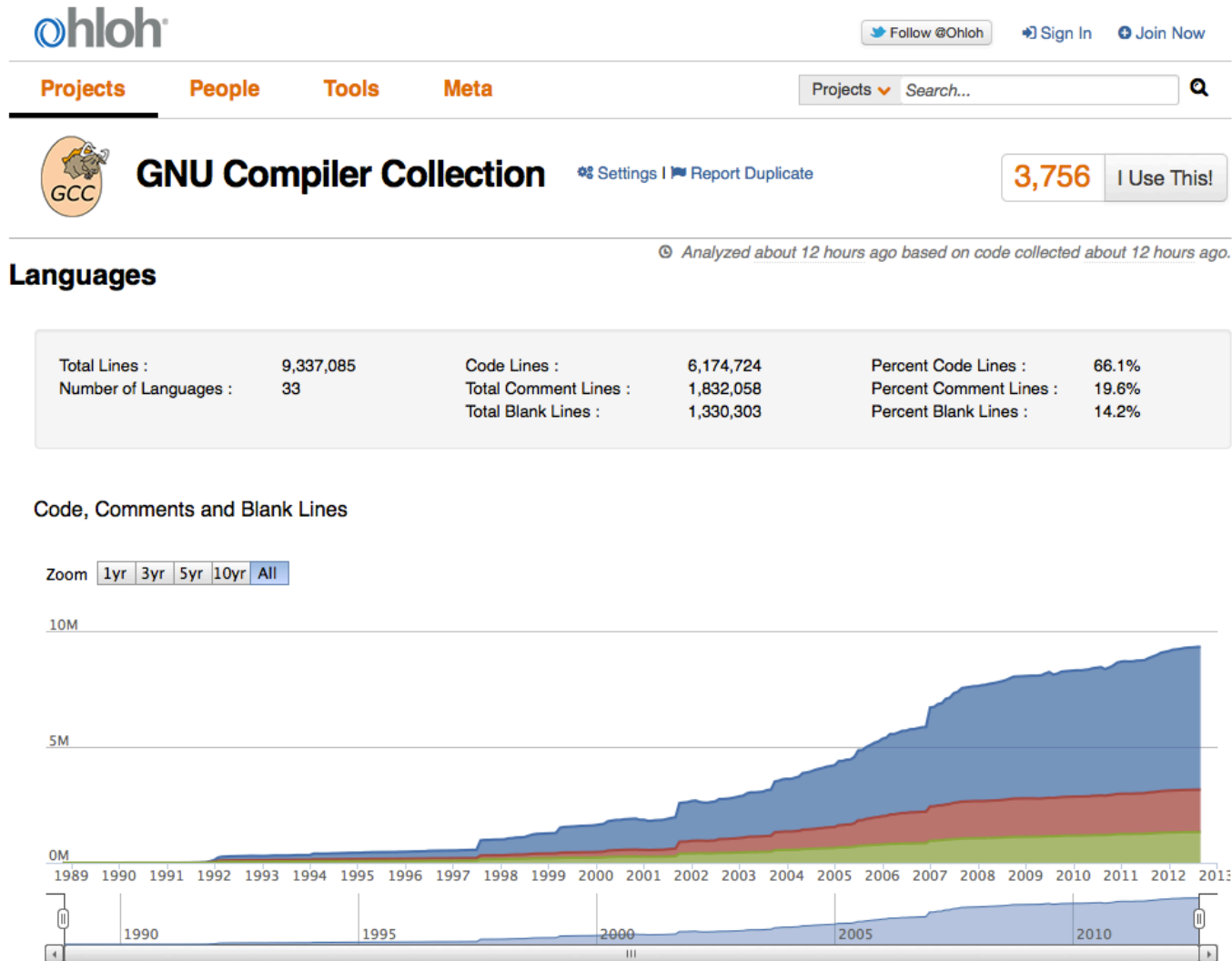
1995

2005

**CHANGE**

**AHEAD**

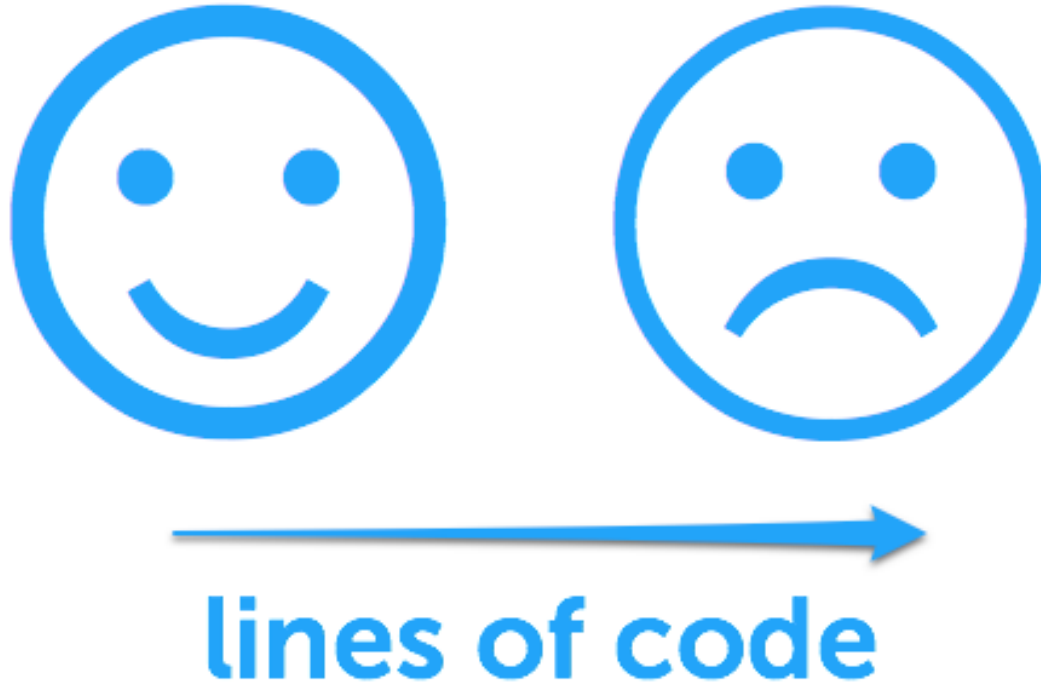
# Software Complexity



See <http://www.ohloh.net/p/gcc>. Retrieved 2012-09-16.

# Software Complexity

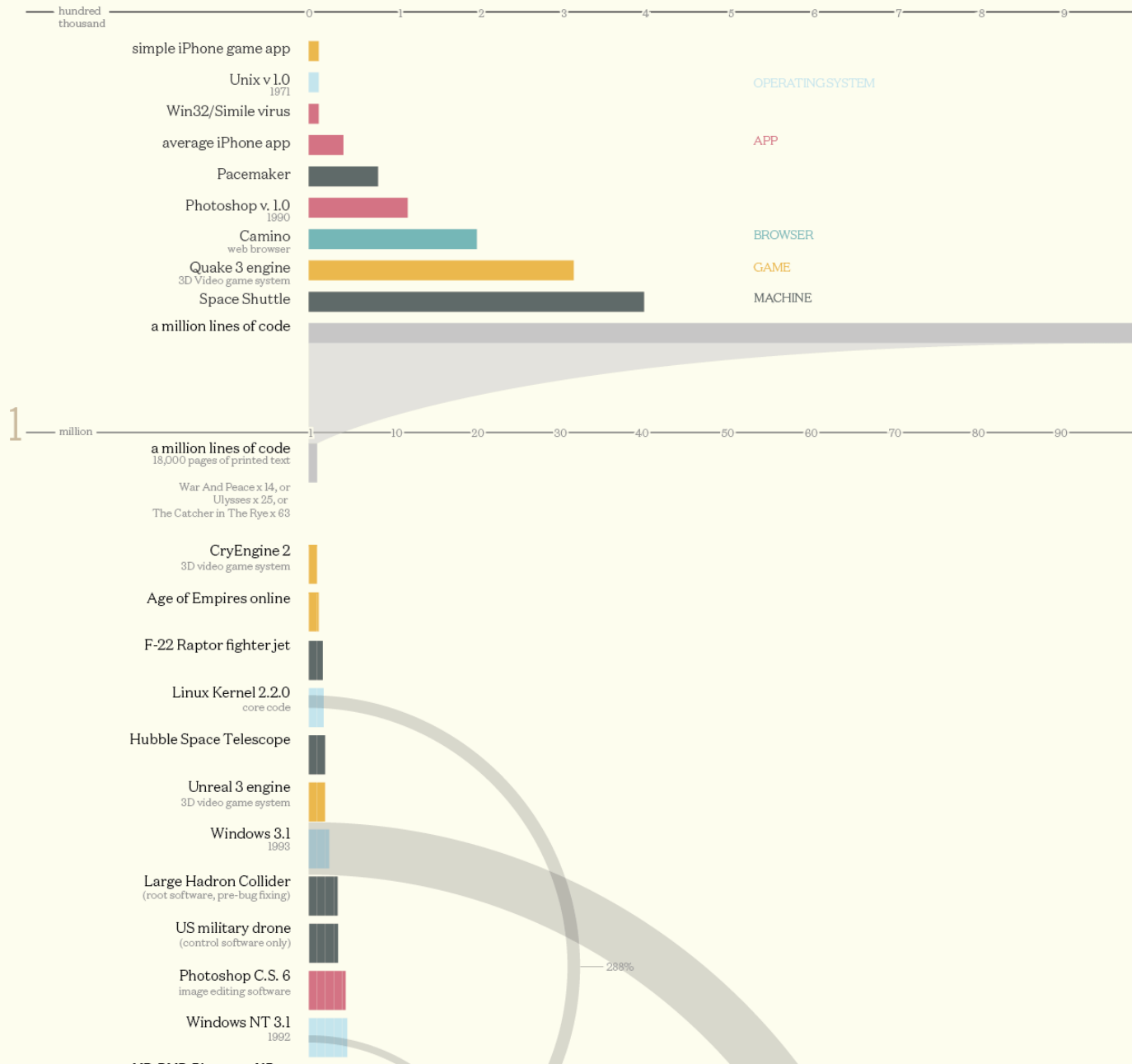
---



**But also...**

# Codebases

Millions of lines of code



5

needed to repair HealthCare.gov  
apparently

Mars Curiosity Rover  
Martian ground vehicle probe

Linux kernel 2.6.0  
2003

Google Chrome  
latest

World of WarCraft  
server only

Boeing 787  
avionics & online support systems only

Windows NT 3.5  
1993

Firefox  
latest version

10

Chevy Volt  
electric car

Intuit Quickbooks  
accounting software

Windows NT 4.0  
1996

Android  
mobile device operating system

Mozilla Core  
core code at heart of all Mozilla's software

MySQL  
database language

Boeing 787  
total flight software

Linux 3.1  
latest version

Apache Open Office  
open-source office software

F-35 Fighter jet  
2013

25

Microsoft Office 2001

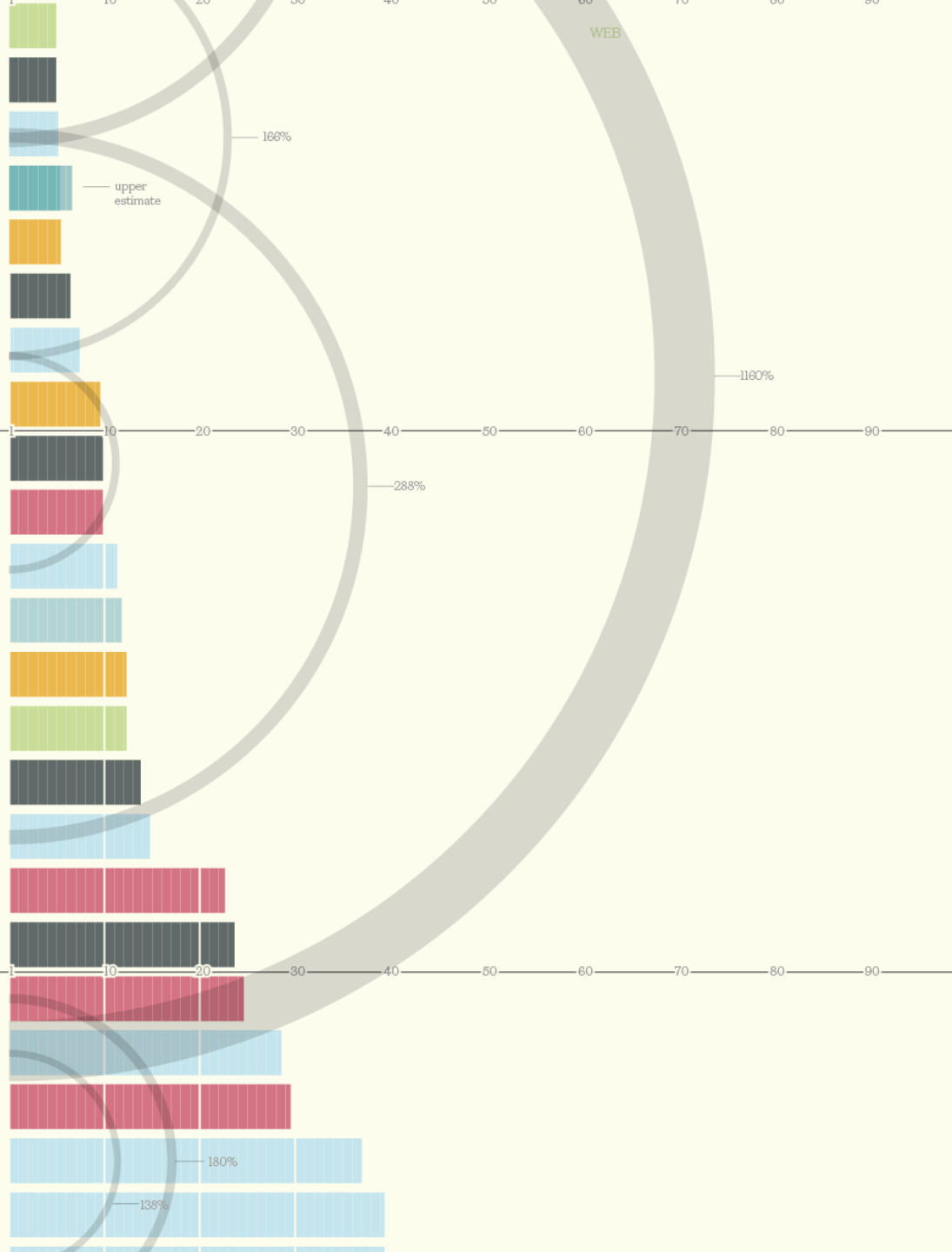
Windows 2000

Microsoft Office for Mac  
2006

Symbian  
mobile operating system

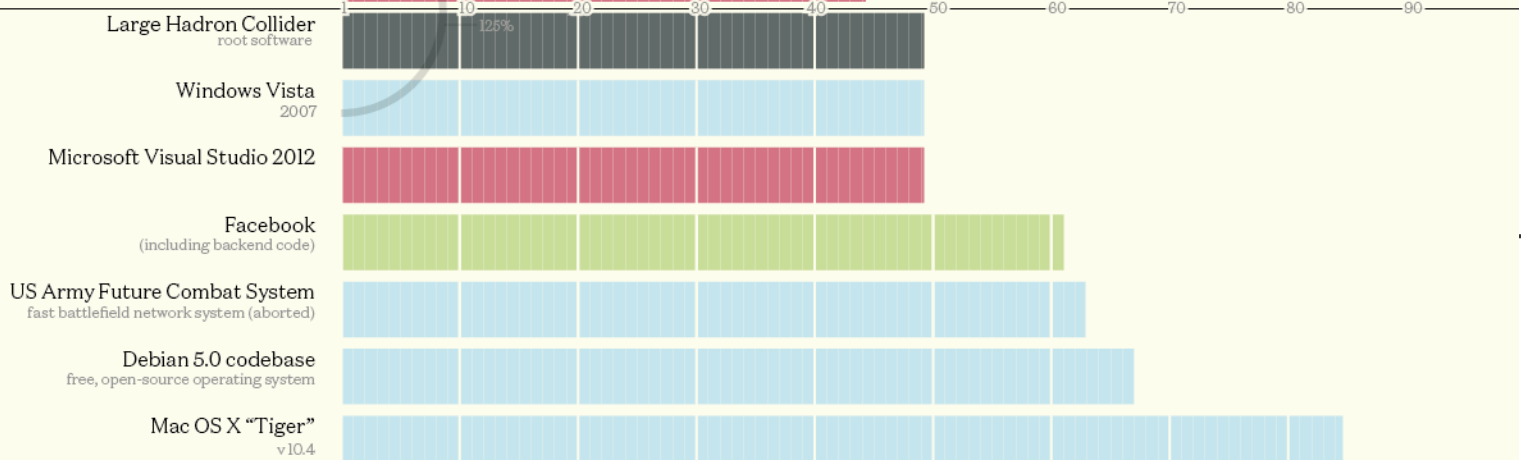
Windows 7  
2009

Windows XP

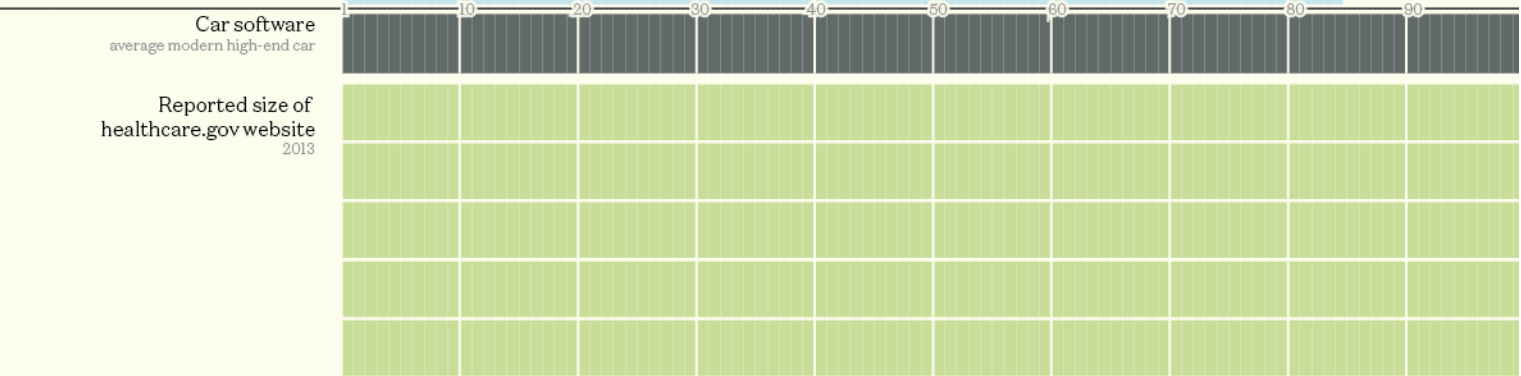




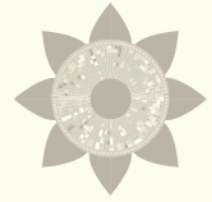
50



100



concept & design: David McCandless  
**informationisbeautiful.net**  
 research: Pearl Doughty-White, Miriam Quick



work in progress  
 v0.62 // Oct 2013

sources NASA, Quora, Ohloh, Wired & press reports  
 note some guess work, rumours & estimates  
 data bit.ly/KIB\_linescode

Language	Code Lines	Comment Lines	Comment Ratio	Blank Lines	Total Lines	Total Percentage
C	2,300,710	476,978	17.2%	452,773	3,230,461	34.6%
C++	1,206,025	250,128	17.2%	252,971	1,709,124	18.3%
Java	743,003	699,939	48.5%	179,887	1,622,829	17.4%
Ada	729,322	335,302	31.5%	252,886	1,317,510	14.1%
Autoconf	450,574	756	0.2%	71,979	523,309	5.6%
HTML	214,572	6,279	2.8%	43,661	264,512	2.8%
Fortran (Fixed-format)	113,138	2,326	2.0%	15,909	131,373	1.4%
Make	112,507	3,917	3.4%	14,123	130,547	1.4%
Go	66,921	11,083	14.2%	4,904	82,908	0.9%
Assembly	51,774	13,375	20.5%	10,080	75,229	0.8%
XML	49,875	675	1.3%	6,062	56,612	0.6%
Objective-C	28,137	5,215	15.6%	8,279	41,631	0.4%
shell script	19,657	5,823	22.9%	4,417	29,897	0.3%
Fortran (Free-format)	17,068	3,305	16.2%	1,686	22,059	0.2%
Perl	16,549	3,869	18.9%	2,463	22,881	0.2%
TeX/LaTeX	12,823	6,358	33.1%	1,639	20,820	0.2%
Scheme	11,023	1,010	8.4%	1,205	13,238	0.1%
Automake	10,775	1,210	10.1%	1,626	13,611	0.1%
Modula-2	4,326	983	18.5%	826	6,135	0.1%
Objective Caml	2,930	578	16.5%	389	3,897	0.0%
XSL Transformation	2,896	450	13.4%	576	3,922	0.0%
AWK	2,318	569	19.7%	376	3,263	0.0%
CSS	2,049	171	7.7%	453	2,673	0.0%
Python	1,735	410	19.1%	404	2,549	0.0%
Pascal	1,044	141	11.9%	218	1,403	0.0%
C#	879	506	36.5%	230	1,615	0.0%
DCL	698	154	18.1%	15	867	0.0%
JavaScript	655	404	38.1%	144	1,203	0.0%
Tcl	392	113	22.4%	72	577	0.0%
Haskell	154	0	0.0%	17	171	0.0%
CMake	134	31	18.8%	25	190	0.0%
Matlab	57	0	0.0%	8	65	0.0%
DOS batch script	4	0	0.0%	0	4	0.0%
<b>Totals</b>	<b>6,174,724</b>	<b>1,832,058</b>		<b>1,330,303</b>	<b>9,337,085</b>	

**- Interoperability**

See <http://www.ohloh.net/p/gcc>. Retrieved 2012-09-16.

## Report: Healthcare website failed test ahead of rollout

By **Ed Payne**, **Matt Smith** and **Tom Cohen**, CNN

October 23, 2013 -- Updated 0103 GMT (0903 HKT)



### Report: Obamacare site failed early test

#### STORY HIGHLIGHTS

- **NEW:** Top White House official part of "tech surge" on Obamacare
- Obamacare "is not failing" despite website woes, White House spokesman says
- Obama says HealthCare.gov problems are "going to get fixed"
- Secretary Sebelius expected to testify at a congressional hearing next week

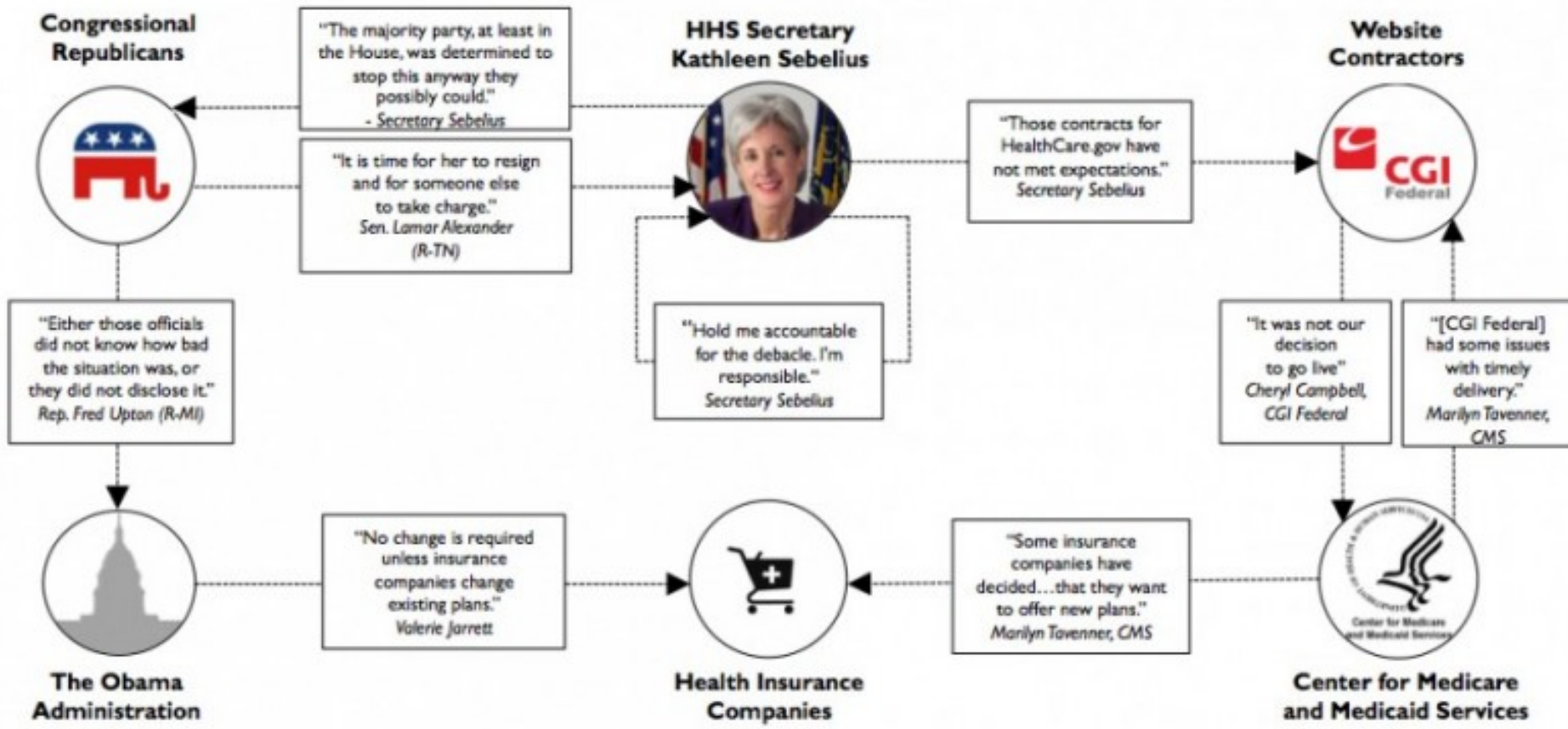
**Washington (CNN)** -- The President's healthcare sign-up web page was supposed to handle tens of thousands of people at once. But in a trial run days before its launch, just a few hundred users flatlined the site.

Despite the problems, federal health officials pushed aside the crash cart and rolled out [HealthCare.gov](#) on October 1 as planned, [The Washington Post](#) reported.

The result? The website crashed shortly after midnight as a couple thousand people tried to start the process, two people familiar with the project told the Post.

# Requirements engineering/Management problem

## ACA Finger-Pointing Flowchart



<http://www.washingtonpost.com/blogs/wonkblog/wp/2013/11/01/thirty-one-things-we-learned-in-healthcare-govs-first-31-days/>

# Thirty-one things we learned in HealthCare.gov's first 31 days

Scalability problem

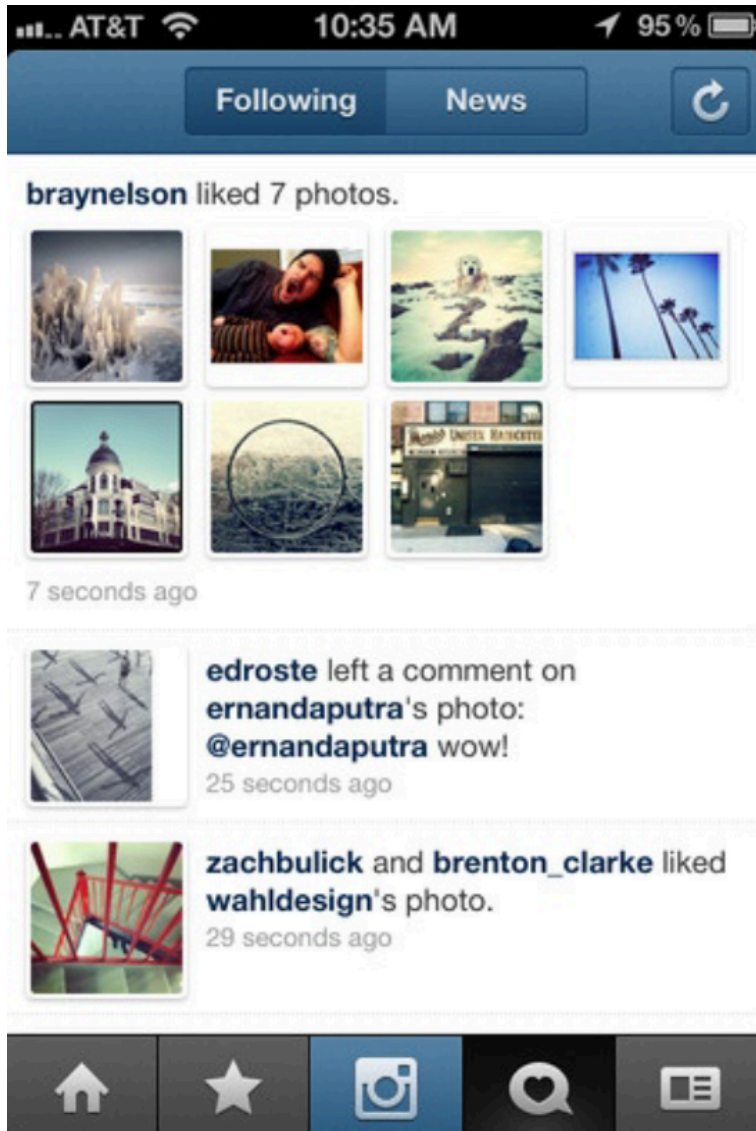
Technical problems (e.g., inaccurate data, cancellation failures)

Testing issues

<http://www.washingtonpost.com/blogs/wonkblog/wp/2013/11/01/thirty-one-things-we-learned-in-healthcare-govs-first-31-days/>



# Instagram Story



# Instagram Story

---

« Instagram is an app that **only took 8 weeks** to build and ship, but was a product of over a year of work. »

# Instagram Story

---

« While I was there working in marketing, I started doing more and more engineering at night on simple ideas that helped me learn how to program (**I don't have any formal CS degree or training**) »



# Instagram Story

---

« We spent 1 week prototyping a version that focused solely on photos.

It was pretty awful. So we went back to creating a native version of Burbn. We actually got an entire version of Burbn done as an iPhone app, but it felt cluttered, and overrun with features. It was really difficult to decide to start from scratch, but we went out on a limb, and basically cut everything in the Burbn app except for its photo, comment, and like capabilities. What remained was Instagram. »

# Instagram Story

---

« So 8 weeks later, we gave it to our friends, beta tested, bug fixed, etc. and this Monday we decided it was ready to ship. »

# Instagram Story

---

« Who is responsible for Instagram's UI design?

For better or for worse, I've done most of the pixel pushing in our app. ;) »

# Instagram Story

---

- 30+ millions d'utilisateur en 2 ans
- 25k inscriptions le premier jour
  - « best & worst day of our lives so far »
  - « favicon » cause des milliers d'erreurs 404
    - « 404-ing on Django, causing tons of errors »
- Un seul serveur au lancement
  - Moins puissant qu'un MacBook Pro
- La suite: passage à l'échelle, cloud (EC2) et ingénierie du logiciel

<https://speakerdeck.com/mikeyk/scaling-instagram>

<http://zoompf.com/blog/2012/04/instagram-and-optimizing-favicons>

# Instagram Story

---

- Sur la trentaine de composants, 4 seulement ont été écrits à partir de zéro
  - App iOS, App Android, Android Push Notification Service et Redis Query analyzer



node2dm



Fabric



# Instagram Story (key lessons)

---

- Sélection et intégration de multiples librairies
- Open source community
  - Apprendre, partager, demander, répondre, etc.
- Auto-apprentissage
  - « Product guys » sont maintenant à même de rivaliser...
- Agilité, développement incrémental

# Software Complexity

**NOKIA**  
Connecting People Know our past. Create the future...

1982 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994

**- Reusability**

**19 - Durability**

1998 1999

2000 2001 2002

2003 2004

2005

2006

**symbian OS**

**iPhone OS 4.0 Software**

**BlackBerry 6**

**- Variability**



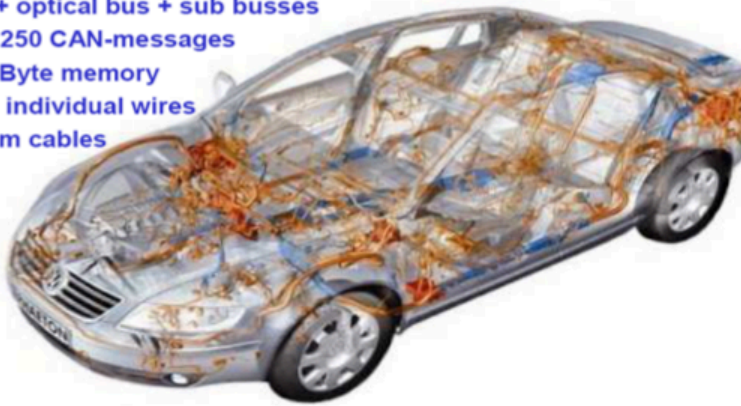
# Software Complexity

- **Critical**
- **Real-time**
- **Embedded**



## Phaeton

- ◆ 61 networked ECUs
- ◆ 3 bus systems + optical bus + sub busses
- ◆ 2500 signals in 250 CAN-messages
- ◆ more than 50 MByte memory
- ◆ more than 2000 individual wires
- ◆ more than 3800m cables





# Software Complexity

---

- *"The avionics system in the F-22 Raptor [...] consists of about 1.7 million lines of software code."*
- *"F-35 Joint Strike Fighter [...] will require about 5.7 million lines of code to operate its onboard systems."*
- *"Boeing's new 787 Dreamliner [...] requires about 6.5 million lines of software code to operate its avionics and onboard support systems."*
- *"if you bought a premium-class automobile recently, it probably contains close to 100 million lines of software code. [...] All that software executes on 70 to 100 microprocessor-based electronic control units (ECUs) networked throughout the body of your car."*
- *"Alfred Katzenbach, the director of information technology management at Daimler, has reportedly said that the radio and navigation system in the current S-class Mercedes-Benz requires over 20 million lines of code alone and that the car contains nearly as many ECUs as the new Airbus A380 (excluding the plane's in-flight entertainment system)."*
- *"TBM claims that approximately 50 percent of car warranty costs are now related to electronics and their embedded software"*

*"This Car Runs on Code", By Robert N. Charrette, IEEE Spectrum, Feb. 2009, see <http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code>*

# Long term availability...

## AIRBUS A300 Life Cycle

Program began in 1972, production stopped in 2007

**2007-1972 = 35 years...**

Support will last until 2050

**2050-1972 = 78 years !!**

**On board software development  
for very long lifecycle products**

*From the OPEES ITEA2 project (2009-2012)*

# Software Complexity

---

- **Distributed**
- **Large-scale**

## ➤ **Google (2012 Update from Larry Page, CEO):**

- *Over 850,000 Android devices are activated daily through a network of 55 manufacturers and more than 300 carriers.*
- *Google Chrome browser has over 200 million users.*
- *Google launched Gmail in 2004 and now is used by more than 350 million people.*
- *YouTube has over 800 million monthly users who upload an hour of video per second.*

See <http://investor.google.com/corporate/2012/ceo-letter.html>

# Software Complexity

---

- **Google, Building for Scale:**
  - 6,000 developer / 1,500+ projects
  - Each product has custom release cycles
    - few days to few weeks
  - 1 (!! ) code repository
  - No binary releases
    - everything builds from HEAD
  - 20+ code changes per minute
    - 50% of the code changes every month



Google

Innovation Factory:  
Testing, Culture, &  
Infrastructure

Patrick Copeland, Google  
ICST 2010

Source: <http://googletesting.blogspot.com/search/label/Copeland>

# Software Complexity

---

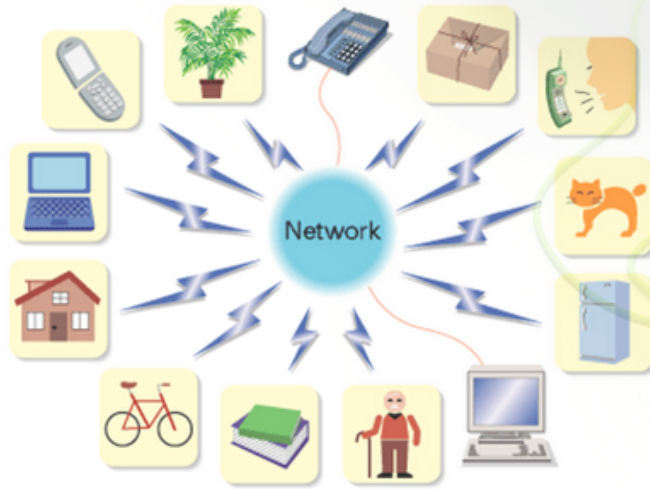
## ➤ Free Mobile (10/01/2012):

- *“A 9h45, le site mobile.free.fr cumulait déjà plus d'un million d'accès simultanés alors que le site proposait seulement une page invitant à patienter” !!*

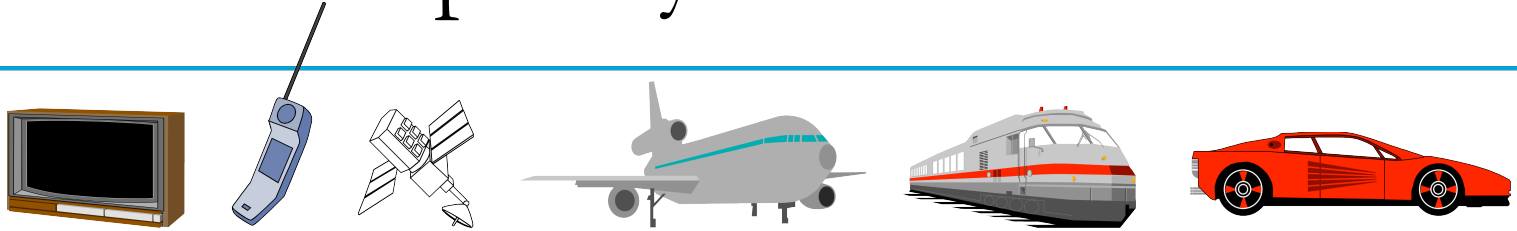


# Software Complexity

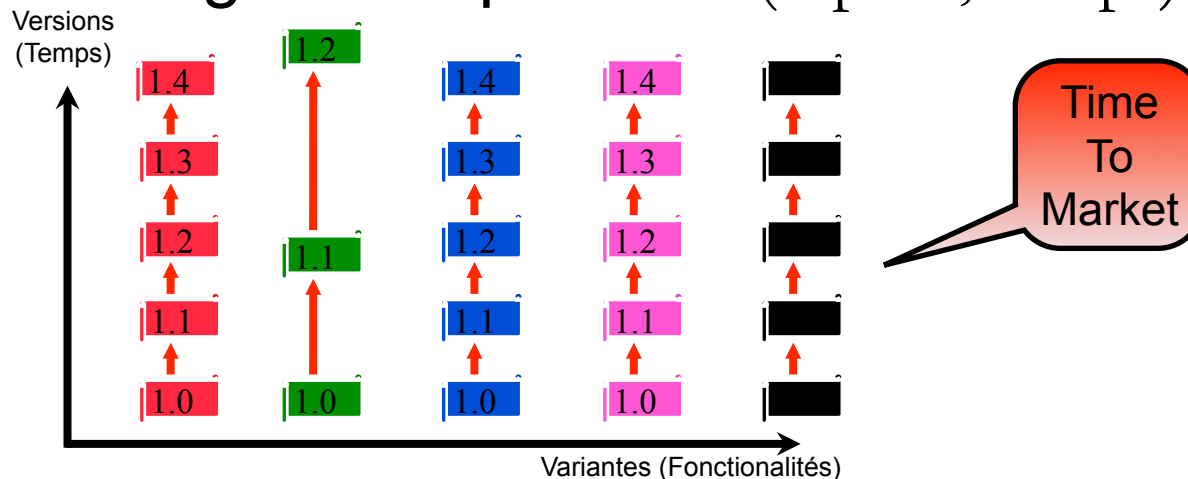
- Autonomic Computing
- Cloud Computing
- PaaS, SaaS, IoS, IoT...



# Software Complexity

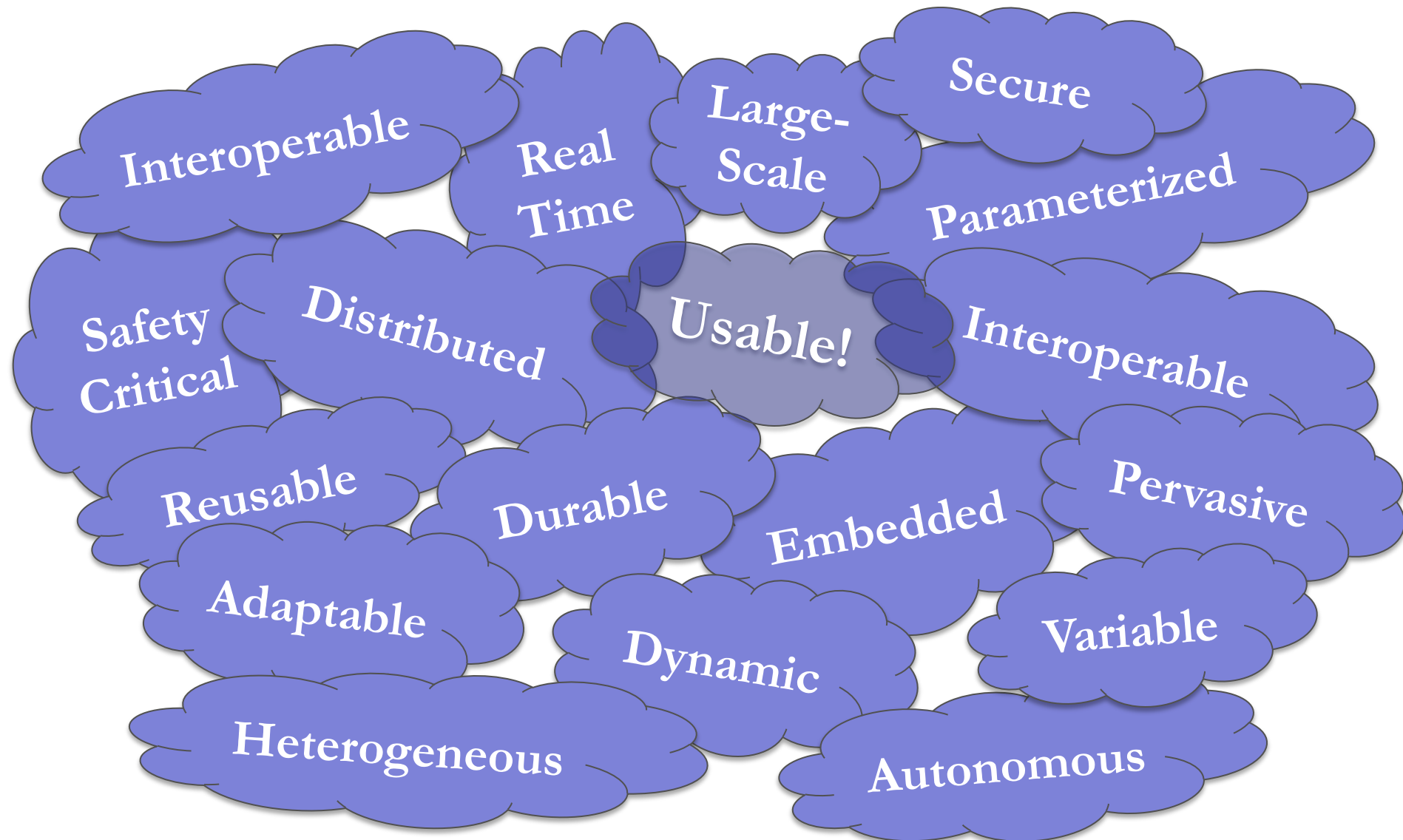


- Importance des aspects non fonctionnels
  - systèmes répartis, parallèles et asynchrones
  - qualité de service : fiabilité, latence, performances...
- Flexibilité accrue des aspects fonctionnels
  - notion de lignes de produits (espace, temps)



# Software Complexity: Some Dimensions

---





# Défaillances

---

- Catastrophe humaine ou financière:

- Therac-25 (1985-1987) – radiologie et contrôle d'injection de substances radioactives
- Iran Air Flight 655 (1988) – guerre d'Irak et missile américain – système radar
- London Ambulance System (1992) – central et dispatch ambulances
- Ariane 5 (1996)
- Mars Climate Orbiter (1999) – sonde spatiale – unité de mesure
- Bourse de Londres (Taurus, 1993) – SI qui n'a pas pu être déployé
- SI du FBI (2005) – SI qui n'a pas pu être déployé
- Un canon-robot anti-aérien sud-africain tue neuf soldats dans un mode tout automatique qui avait été rajouté (2007)

- Image de marque :

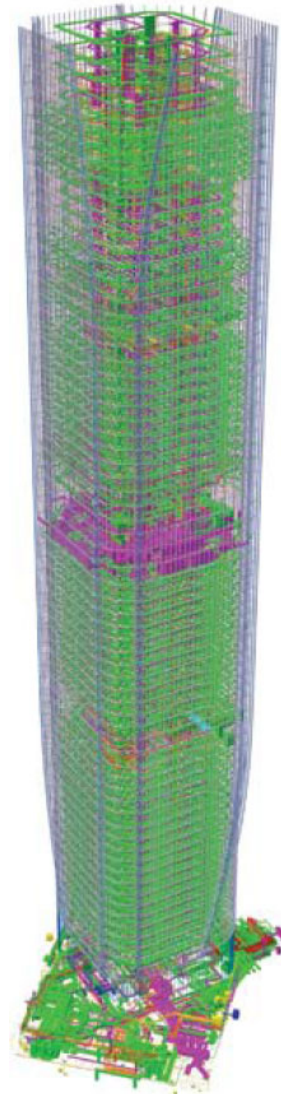
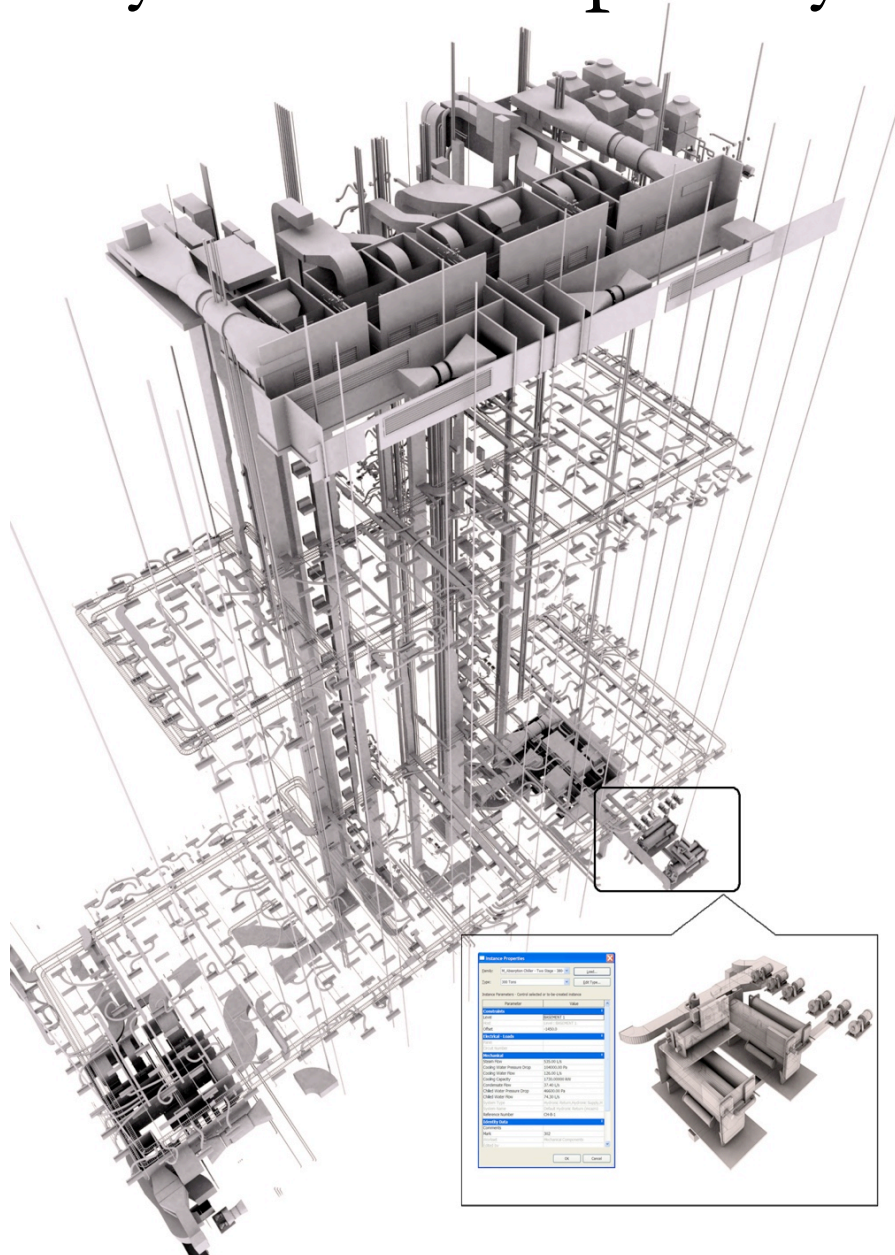
- FT et Bouygues en 2004 – crash des serveurs – indisponibilité 48h
- Playstation 3 : le système de jeux en ligne croyait qu'il y avait un 29 février 2010, ce qui bloqua le fonctionnement des jeux en ligne toute la journée du 1 mars 2010
- iPhone 4 : problème de réveil après le passage à l'heure d'hiver à l'automne 2010, après le passage à 2011

- Succès financier: Windows ;)

- Sans conséquence mais énervant : bugs @ Irisa, ...

- D'autres : [http://en.wikipedia.org/wiki/Computer\\_bug](http://en.wikipedia.org/wiki/Computer_bug)

# System Complexity



# Failures in System Engineering

---





# Failures in System Engineering

---



# Outline

---

- ① Issues in Software Engineering
- ② Evolution in Software Engineering
- ③ State of the Practice
- ④ Modeling in Software Engineering

# Software Engineering

---

The production of operational software satisfying defined standards of quality...

... includes programming, but is more than programming!

The five components of Software Engineering [Meyer]:

- **Describe:** *requirements, design, specification, documentation...*
- **Implement:** *modeling, programming*
- **Assess:** *testing and other V&V techniques*
- **Manage:** *plans, schedules, communication, reviews*
- **Operate:** *deployment, installation...*

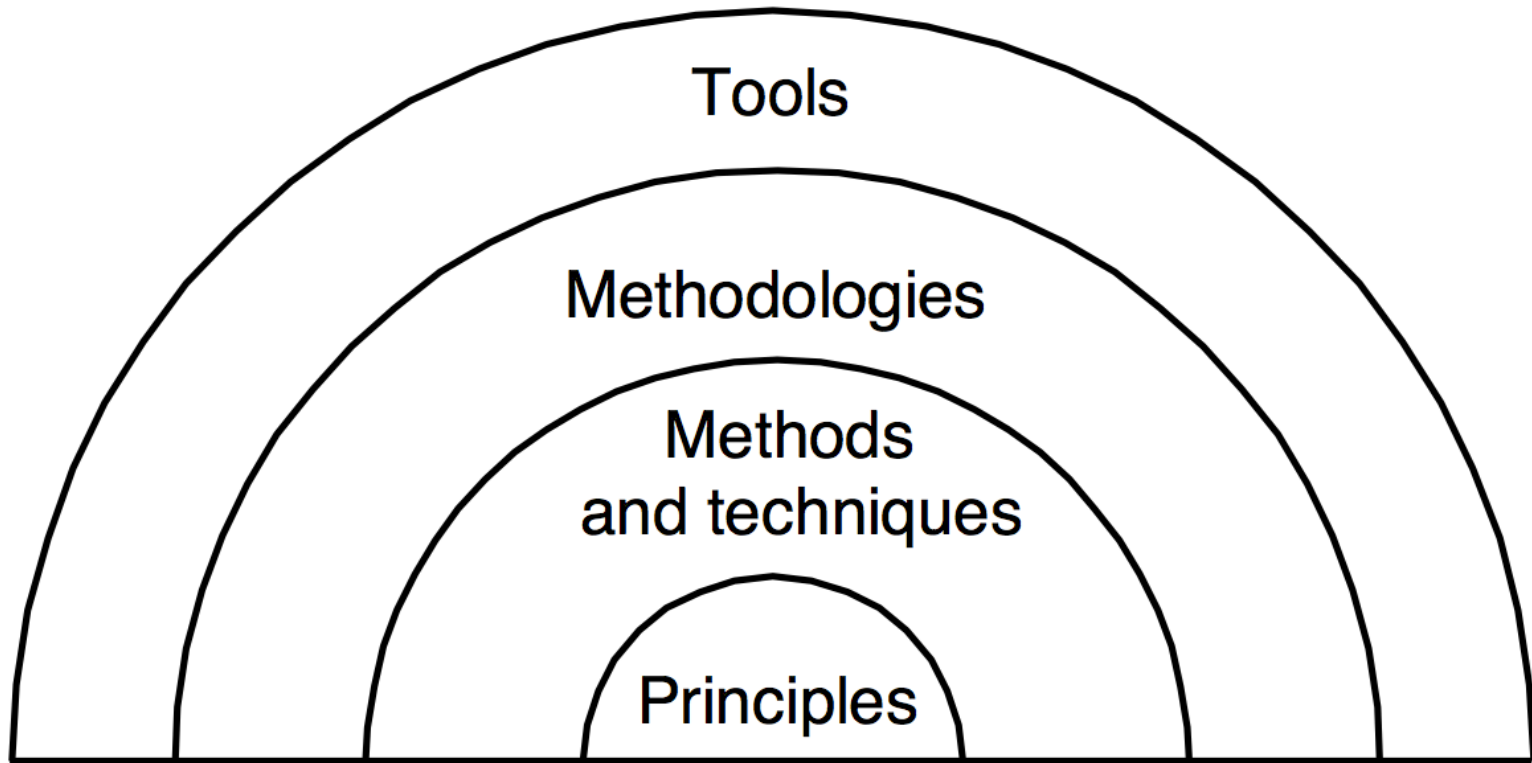
# Software Engineering: definition

---

- is a **profession** dedicated to designing, implementing, and modifying software so that it is of higher quality, more affordable, maintainable, and faster to build.
- is a **systematic approach** to the analysis, design, assessment, implementation, test, maintenance and re-engineering of a software by applying engineering to the software.
- first appeared in the 1968 NATO Software Engineering Conference (to provoke thought regarding the perceived "software crisis" at the time).

# Software Engineering: Basics

---



Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli.  
*Fundamentals of Software Engineering, 2nd edition.* 2002.

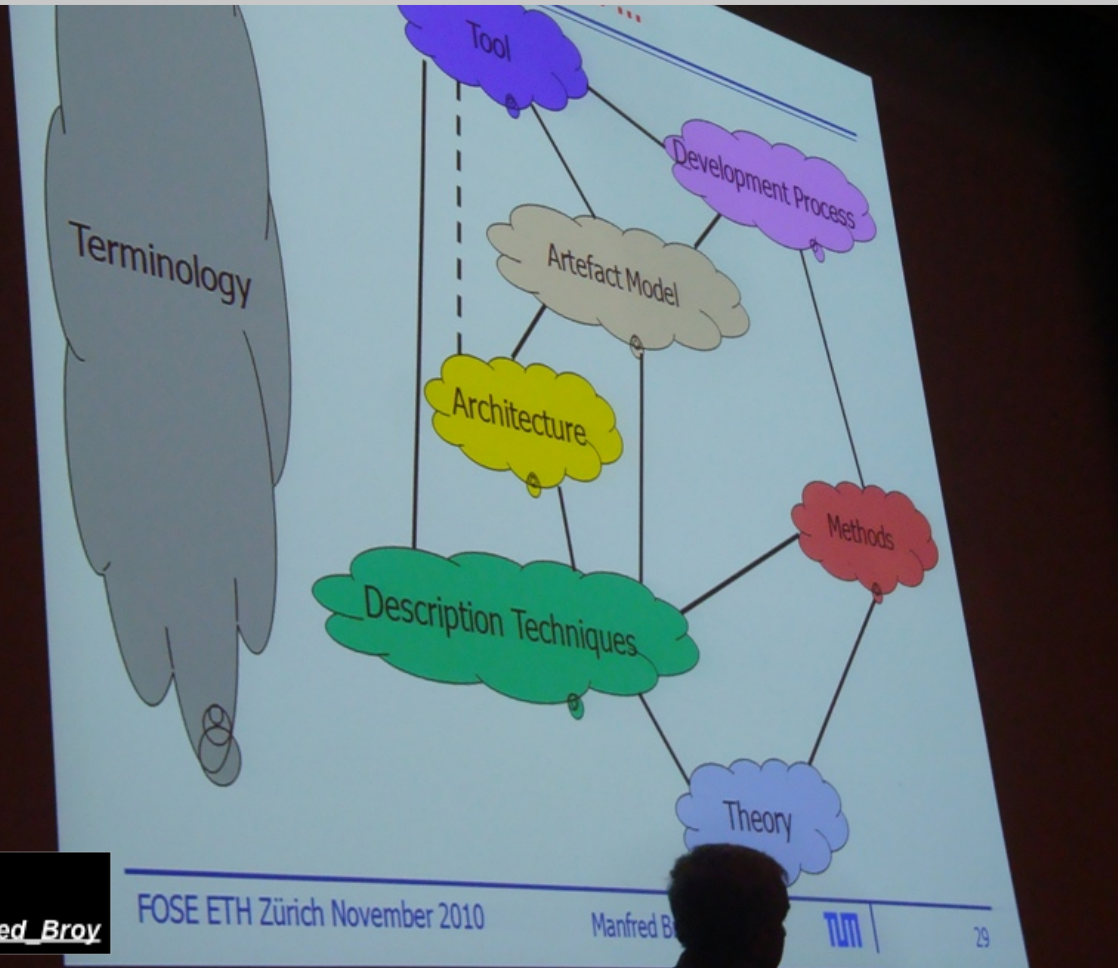


## What we need?



**Manfred Broy**

[http://en.wikipedia.org/wiki/Manfred\\_Broy](http://en.wikipedia.org/wiki/Manfred_Broy)



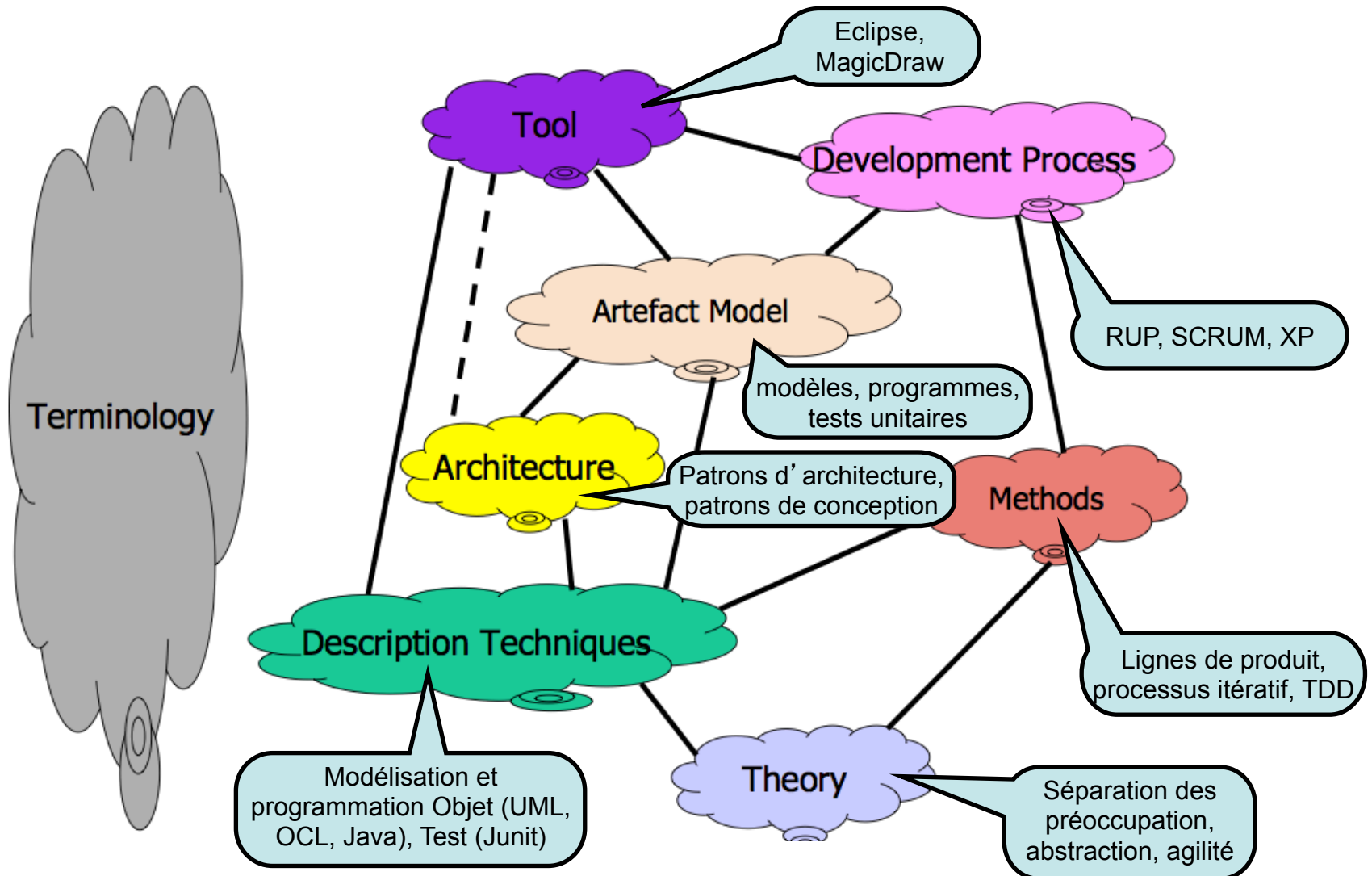
FOSE ETH Zürich November 2010

Manfred Broy



29

# Software Engineering: Basics



# Des logiciels complexes...

---

## ■ Logiciels de grande taille

- des millions de lignes de code
- des équipes nombreuses
- durée de vie importante
- des lignes de produits
- plateformes technologiques complexes
- évolution continue

## ■ Logiciels complexes

## ■ Logiciels critiques

## ■ ...

# Aujourd'hui / Today

- Méthodes de conception et de validation de logiciel
  - En fait: génie logiciel / software engineering
  - Comment développer des systèmes logiciels de plus en plus complexe?
- **#1 Prendre conscience de la complexité des systèmes logiciels actuels et à venir**
  - Les enjeux et l'impact sur le métier

# Exemple très concret (yet another)

- I insist: software engineering is difficult!
- Even on a very basic (apparently) example
- It will be our « running » example for the rest of the courses



**Jeu d'échecs aka Chess**

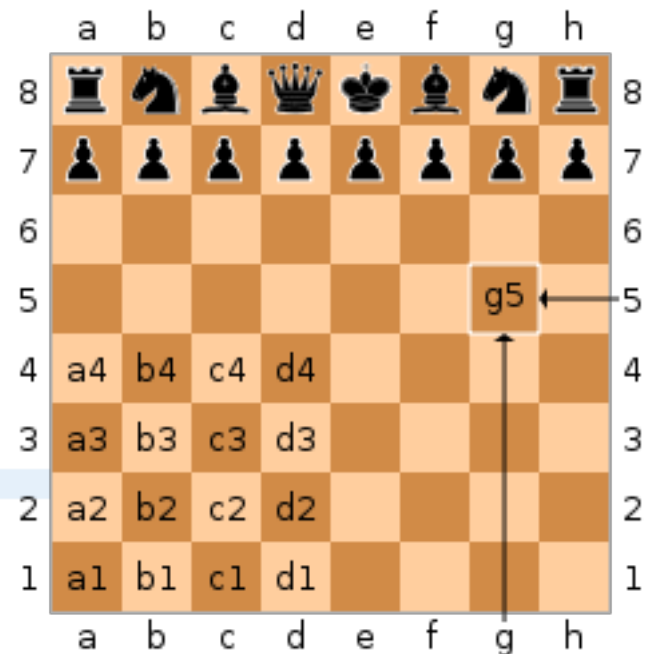
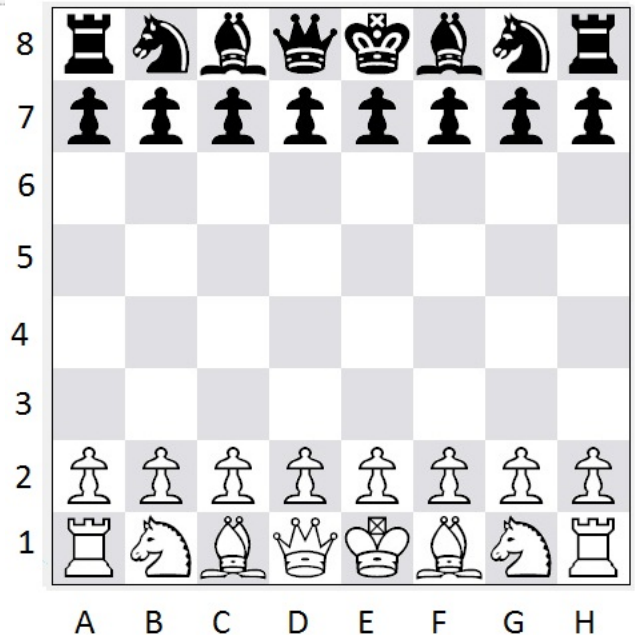
# Day #0

- Spécification du client
  - « Développer un jeu d'échecs pour jouer tout seul »
  - Quel langage?
  - Java
  - Combien?
  - 1000\$
  - OK. Let us go

```

1  /**
2   * @author Should I give my name for this code?
3   *
4   */
5  public class ChessDay0 {
6
7
8     private int[][] board ;
9
10    public ChessDay0() {
11        board = new int[8][8] ;
12
13
14        for (int j = 0; j < 8 ; j++) {
15
16            // second row (for white)
17            board[1][j] = 0 ; // 0 is a pawn yes
18
19            // second row (for black)
20            board[6][j] = 0 ; // 0 is a pawn yes
21
22        }
23
24
25        board[0][0] = 1 ; // rook (white)
26        board[0][7] = 1 ; // rook (white)
27
28        //..
29
30        board[7][0] = 1 ; // rook (black)
31        board[7][7] = 1 ; // rook (black)
32
33        //..
34
35    }
36
37
38    public static void main(String[] args) {
39        System.out.println(new ChessDay0());
40
41    }
42
43 }
44

```





# Day #1 (refactoring)

The image shows an IDE with two Java files: ChessDay0.java and ChessDay1.java. The ChessDay1.java file contains the following code:

```
1 /**
2  * @author Should I give my name for this code?
3  *
4  */
5 public class ChessDay1 {
6
7     private int[][] board ;
8
9     public ChessDay1() {
10        board = new int[8][8] ;
11
12        for (int j = 0; j < 8; j++) {
13
14            // second row (for white)
15            board[1][j] = 0 ; // 0 is a pawn ye
16
17            // second row (for black)
18            board[6][j] = 0 ; // 0 is a pawn ye
19
20        }
21
22        board[0][0] = 1 ; // rook (white)
23        board[0][7] = 1 ; // rook (white)
24
25        //...
26
27        board[7][0] = 1 ; // rook (bl
28        board[7][7] = 1 ; // rook (bl
29
30        //...
31
32    }
33
34    public static void main(String[] args) {
35        System.out.println(new ChessDay1());
36    }
37 }
```

The 'Extract Method' dialog is open, showing the following configuration:

- Method name: `initBoard`
- Access modifier:  public  protected  default  private
- Declare thrown runtime exceptions
- Generate method comment
- Replace additional occurrences of statements with method
- Method signature preview: `private void initBoard()`

A context menu is open over the code, with the following options:

- Undo
- Revert File
- Save
- Open Declaration
- Open Type Hierarchy
- Open Call Hierarchy
- Show in Breadcrumb
- Quick Outline
- Quick Type Hierarchy
- Open With
- Show In
- Cut
- Copy
- Copy Qualified Name
- Paste
- Quick Fix
- Source
- Refactor
- Surround With
- Local History
- References
- Declarations
- Add to Snippets...
- Debug As
- Move...
- Change Method Signature...
- Extract Method...
- Extract Interface...
- Use Supertype Where Possible...
- Extract Class...
- Introduce Parameter Object...

```

initBoard();

}

/**
 *
 */
private void initBoard() {
    for (int j = 0; j < 8 ; j++) {

        // second row (for white)
        board[1][j] = 0 ; // 0 is a pawn yes

        // second row (for black)
        board[6][j] = 0 ; // 0 is a pawn yes

    }

    board[0][0] = 1 ; // rook (white)
    board[0][7] = 1 ; // rook (white)

    //..

    board[7][0] = 1 ; // rook (black)
    board[7][7] = 1 ; // rook (black)

    //..
}

```

Day #1 (refactoring)

Piece	King	Queen	Rook	Bishop	Knight	Pawn
Number	1	1	2	2	2	8
Symbols	 	 	 	 	 	 

# Day #1 (refactoring)

```

PieceType.java
MDIChess > src > (default package) > P
1+ /**
4
5- /**
6  * @author macher1
7  *
8  */
9 public enum PieceType {
10
11     PAWN,
12     ROOK,
13     KING,
14     QUEEN,
15     BISHOP,
16     KNIGHT
17
18 }
19
20

ChessDay0.java  ChessDay1.java
MDIChess > src > (default package) > C
5 public class ChessDay1 {
6
7
8     private PieceType[][] board ;
9

```

```

private void initBoard() {
    for (int j = 0; j < 8 ; j++) {

        // second row (for white)
        board[1][j] = PieceType.PAWN ;
        // second row (for black)
        board[6][j] = PieceType.PAWN ;

    }

    board[0][0] = PieceType.ROOK ; // white
    board[0][7] = PieceType.ROOK ; // white

    //..

    board[7][0] = PieceType.ROOK ; // black
    board[7][7] = PieceType.ROOK ; // black
}

```

```
r n b q k b . r
p p p p p p p p
. . . . n . .
. . . . . . . .
. . . P . . . .
. . . . N . . .
P P P . P P P P
R N B Q K B . R
```

Thinking...

```
white KQkq
r . b q k b . r
p p p p p p p p
. . n . . n . .
. . . . . . . .
. . . P . . . .
. . . . N . . .
P P P . P P P P
R N B Q K B . R
```

My move is : Nc6  
White (3) :

**Day #3 – Nice demo**

**I want my 1000\$**

# Day #3

- Le client
  - « J'avais demandé une version graphique... Et on ne peut même pas sauvegarder une partie. Ou lire des parties. Et vous n'avez pas pris en compte la règle du pat! »
  
- Rappel (spécification du client à l'origine)
  - « Développer un jeu d'échecs pour jouer tout seul »

# Restons calme. Analyse

- « **J'avais demandé une version graphique...**
  - Mauvaise analyse des besoins initiaux (cahier des charges)
  - Impact sur le code non négligeable: il va très certainement falloir adapter le code existant (en plus d'implémenter de nouvelles fonctionnalités)
    - Affichage graphique des pièces et des cases
    - Écouter des « évènements » et réagir en conséquence
    - Interdire des interactions (sur des coups impossibles)
  - Mais est-ce que le code est fait pour?

# Retour au code

```
public void displayBoard() {  
  
    for (int i = 0; i < 8; i++) {  
        for (int j = 0; j < 8; j++) {  
            if (board[i][j] == PieceType.ROOK) {  
                System.out.println('R');  
            }  
            // ...  
            else { // square without piece  
                System.out.println('.');  
            }  
            //  
            //...  
        }  
    }  
}
```

**Graphique! Evolution nécessaire...**

System.out?

Couleur des pièces?

Encore une série de if-then-else?

Comment garder la représentation textuelle malgré tout?

# Imaginons un code monolithique (une seule classe)

Graphique?

Sauvegarder une partie?

Charger une partie?

```
ChessDay1
  main(String[]) : void
  board : PieceType[][]
  ChessDay1()
  canMove(int, int, int, int) : boolean
  displayBoard() : void
  get50MoveRulePlyCount() : int
  getCapturedPieces() : List<PieceType>
  getCapturedPieces(boolean) : List<PieceType>
  getCurrentMoveNumber() : int
  getHistory() : String
  getMaterialCount(boolean) : int
  getPossibleMoves() : List<int[][]>
  getPossibleSquares(int, int) : List<int[][]>
  getThreats(int, int) : List<int[][]>
  getUnCapturedPieces(boolean) : List<PieceType>
  initBoard() : void
  isBlackCastleableKingside() : boolean
  isBlackCastleableQueenside() : boolean
  isBlockable() : boolean
  isCheck() : boolean
  isCheckmate() : boolean
  isDoubleCheck() : boolean
  isEnPassantFile(int) : boolean
  isStalemate() : boolean
  isWhiteCastleableKingside() : boolean
  isWhiteCastleableQueenside() : boolean
  isWhiteToPlay() : boolean
  recordMove(int, int, int, int) : boolean
  redo() : boolean
  reset() : void
  setBlackCastleableKingside(boolean) : void
  setBlackCastleableQueenside(boolean) : void
  setWhiteCastleableKingside(boolean) : void
  setWhiteCastleableQueenside(boolean) : void
  undo() : boolean
```



# Restons calme. Analyse

- « J'avais demandé une version graphique...
- Et on ne peut même pas sauvegarder une partie
- Ou lire des parties.
- **Et vous n'avez pas pris en compte la règle du pat!**  
»

# Imaginons un code monolithique (une seule classe)

Règle du pat?

Dans quelle méthode?

```
ChessDay1
  main(String[]) : void
  board : PieceType[][]
  ChessDay1()
  canMove(int, int, int, int) : boolean
  displayBoard() : void
  get50MoveRulePlyCount() : int
  getCapturedPieces() : List<PieceType>
  getCapturedPieces(boolean) : List<PieceType>
  getCurrentMoveNumber() : int
  getHistory() : String
  getMaterialCount(boolean) : int
  getPossibleMoves() : List<int[][]>
  getPossibleSquares(int, int) : List<int[][]>
  getThreats(int, int) : List<int[][]>
  getUnCapturedPieces(boolean) : List<PieceType>
  initBoard() : void
  isBlackCastleableKingside() : boolean
  isBlackCastleableQueenside() : boolean
  isBlockable() : boolean
  isCheck() : boolean
  isCheckmate() : boolean
  isDoubleCheck() : boolean
  isEnPassantFile(int) : boolean
  isStalemate() : boolean
  isWhiteCastleableKingside() : boolean
  isWhiteCastleableQueenside() : boolean
  isWhiteToPlay() : boolean
  recordMove(int, int, int, int) : boolean
  redo() : boolean
  reset() : void
  setBlackCastleableKingside(boolean) : void
  setBlackCastleableQueenside(boolean) : void
  setWhiteCastleableKingside(boolean) : void
  setWhiteCastleableQueenside(boolean) : void
  undo() : boolean
```

# Day #23

- Le client (retour)
  - « C'est mieux mais la règle des 50 coups ne fonctionne pas. Le fou peut se déplacer comme une dame. Il est possible de manger le roi. Il y a parfois des échecs et mat qui n'en sont pas. »
  
- Bugs

# Imaginons un code monolithique (une seule classe)

Ajouter la règle des 50 coups?

## TESTS

Comment s'assurer que le fou n'a  
pas le comportement d'une dame?  
Qu'on ne peut pas manger le roi?  
etc.

```
ChessDay1
  S main(String[]) : void
  board : PieceType[][]
  C ChessDay1()
  canMove(int, int, int, int) : boolean
  displayBoard() : void
  get50MoveRulePlyCount() : int
  getCapturedPieces() : List<PieceType>
  getCapturedPieces(boolean) : List<PieceType>
  getCurrentMoveNumber() : int
  getHistory() : String
  getMaterialCount(boolean) : int
  getPossibleMoves() : List<int[][]>
  getPossibleSquares(int, int) : List<int[][]>
  getThreats(int, int) : List<int[][]>
  getUnCapturedPieces(boolean) : List<PieceType>
  initBoard() : void
  isBlackCastleableKingside() : boolean
  isBlackCastleableQueenside() : boolean
  isBlockable() : boolean
  isCheck() : boolean
  isCheckmate() : boolean
  isDoubleCheck() : boolean
  isEnPassantFile(int) : boolean
  isStalemate() : boolean
  isWhiteCastleableKingside() : boolean
  isWhiteCastleableQueenside() : boolean
  isWhiteToPlay() : boolean
  recordMove(int, int, int, int) : boolean
  redo() : boolean
  reset() : void
  setBlackCastleableKingside(boolean) : void
  setBlackCastleableQueenside(boolean) : void
  setWhiteCastleableKingside(boolean) : void
  setWhiteCastleableQueenside(boolean) : void
  undo() : boolean
```

I don't  
make  
mistakes

# Testing

Visual Studio

Package Load Failure

Package 'ArtOfTest.WebAiiVSIP.WebAiiVSIP, ArtOfTest.WebAiiVSIP, version=1.0.0.0, Culture=neutral, PublicKeyToken=14176981d1873d86' failed to load properly ( GUID = {5EDEC6D-8025-40CF-987C-B8EABA2C7451} ). Please contact package vendor for assistance. Application restart is recommended, due to possible environment corruption. Would you like to disable loading this package in the future? You may use 'devenv /resetskipkpgs' to re-enable package loading.

Yes

No

```
ChessDay1
  S main(String[]) : void
  board : PieceType[][]
  C ChessDay1()
  Move(int, int, int) : boolean
  displayBoard() : void
  getBoardRulePLYCount() : int
  getCapturedPieces() : List<PieceType>
  getCapturedPieces(boolean) : List<PieceType>
  getCurrentMoveNumber() : int
  getHistory() : String
  getMaterialCount(boolean) : int
  getPossibleMoves() : List<int[][]>
  getPossibleSquares(int, int) : List<int[][]>
  getThreats(int, int) : List<int[][]>
  getUnCapturedPieces(boolean) : List<PieceType>
  initBoard() : void
  isBlackCastleableKingside() : boolean
  isBlackCastleableQueenside() : boolean
  isBlockable() : boolean
  isChecked() : boolean
  isCheckedmate() : boolean
  isDoubleCheck() : boolean
  isEnPassantFile(int) : boolean
  isStalemate() : boolean
  isWhiteCastleableKingside() : boolean
  isWhiteCastleableQueenside() : boolean
  isWhiteToPlay() : boolean
  recordMove(int, int, int, int) : boolean
  redo() : boolean
  reset() : void
  setBlackCastleableKingside(boolean) : void
  setBlackCastleableQueenside(boolean) : void
  setWhiteCastleableKingside(boolean) : void
  setWhiteCastleableQueenside(boolean) : void
  undo() : boolean
```

Plus simple de contrôler le comportement d'un *Fou* dans une classe dédiée (et non dans une série de méthodes avec des if-then-else de partout)

```
/**
 * @param i
 * @param j
 * @param x
 * @param y
 * @return true if the piece at i, j can move to x, y
 */
public boolean canMove(int i, int j, int x, int y) {

    PieceType p = board[i][j];
    PieceType p2 = board[x][y];

    if (p == PieceType.ROOK) {

    }

    else if (p == PieceType.BISHOP) {

    }

}
```

```
ChessDay1
  S main(String[]) : void
  board : PieceType[][]
  C ChessDay1()
  canMove(int, int, int, int) : boolean
  displayBoard() : void
  get50MoveRulePlyCount() : int
  getCapturedPieces() : List<PieceType>
  getCapturedPieces(boolean) : List<PieceType>
  getCurrentMoveNumber() : int
  getHistory() : String
  getMaterialCount(boolean) : int
  getPossibleMoves() : List<int[][]>
  getPossibleSquares(int, int) : List<int[][]>
  getThreats(int, int) : List<int[][]>
  getUnCapturedPieces(boolean) : List<PieceType>
  initBoard() : void
  isBlackCastleableKingside() : boolean
  isBlackCastleableQueenside() : boolean
  isBlockable() : boolean
  isCheck() : boolean
  isCheckmate() : boolean
  isDoubleCheck() : boolean
  isEnPassantFile(int) : boolean
  isStalemate() : boolean
  isWhiteCastleableKingside() : boolean
  isWhiteCastleableQueenside() : boolean
  isWhiteToPlay() : boolean
  recordMove(int, int, int, int) : boolean
  redo() : boolean
  reset() : void
  setBlackCastleableKingside(boolean) : void
  setBlackCastleableQueenside(boolean) : void
  setWhiteCastleableKingside(boolean) : void
  setWhiteCastleableQueenside(boolean) : void
  undo() : boolean
```

Testabilité d'un système



On aimerait:

\* donner en entrée des mouvements de Fou

\* vérifier qu'ils sont effectivement interdits ou autorisés

```
/**
 * @param i
 * @param j
 * @param x
 * @param y
 * @return true if the piece at i, j can move to x, y
 */
public boolean canMove(int i, int j, int x, int y) {

    PieceType p = board[i][j];
    PieceType p2 = board[x][y];

    if (p == PieceType.ROOK) {

    }

    else if (p == PieceType.BISHOP) {

    }

}
```

```
ChessDay1
  S main(String[]) : void
  board : PieceType[][]
  C ChessDay1()
  canMove(int, int, int, int) : boolean
  displayBoard() : void
  get50MoveRulePlyCount() : int
  getCapturedPieces() : List<PieceType>
  getCapturedPieces(boolean) : List<PieceType>
  getCurrentMoveNumber() : int
  getHistory() : String
  getMaterialCount(boolean) : int
  getPossibleMoves() : List<int[][]>
  getPossibleSquares(int, int) : List<int[][]>
  getThreats(int, int) : List<int[][]>
  getUnCapturedPieces(boolean) : List<PieceType>
  initBoard() : void
  isBlackCastleableKingside() : boolean
  isBlackCastleableQueenside() : boolean
  isBlockable() : boolean
  isCheck() : boolean
  isCheckmate() : boolean
  isDoubleCheck() : boolean
  isEnPassantFile(int) : boolean
  isStalemate() : boolean
  isWhiteCastleableKingside() : boolean
  isWhiteCastleableQueenside() : boolean
  isWhiteToPlay() : boolean
  recordMove(int, int, int, int) : boolean
  redo() : boolean
  reset() : void
  setBlackCastleableKingside(boolean) : void
  setBlackCastleableQueenside(boolean) : void
  setWhiteCastleableKingside(boolean) : void
  setWhiteCastleableQueenside(boolean) : void
  undo() : boolean
```

Testabilité d'un système



## On aimerait:

- \* donner en entrée des mouvements de Fou
- \* vérifier qu'ils sont effectivement interdits ou autorisés

- Testability
  - degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met.”
  - Controllability + Observability
- **Controllability** ability to manipulate the **software's input** as well as to place this software into a particular **state**
- **Observability** deals with the possibility to observe the outputs and state changes
- How to improve Testability?
  - Improve Modularity (Refactoring, Design patterns)

```
ChessDay1
  S main(String[]) : void
  board : PieceType[][]
  C ChessDay1()
  canMove(int, int, int, int) : boolean
  displayBoard() : void
  get50MoveRulePlyCount() : int
  getCapturedPieces() : List<PieceType>
  getCapturedPieces(boolean) : List<PieceType>
  getCurrentMoveNumber() : int
  getHistory() : String
  getMaterialCount(boolean) : int
  getPossibleMoves() : List<int[][]>
  getPossibleSquares(int, int) : List<int[][]>
  getThreats(int, int) : List<int[][]>
  getUnCapturedPieces(boolean) : List<PieceType>
  initBoard() : void
  isBlackCastableKingside() : boolean
  isBlackCastableQueenside() : boolean
  isBlockable() : boolean
  isCheck() : boolean
  isCheckmate() : boolean
  isDoubleCheck() : boolean
  isEnPassantFile(int) : boolean
  isStalemate() : boolean
  isWhiteCastableKingside() : boolean
  isWhiteCastableQueenside() : boolean
  isWhiteToPlay() : boolean
  recordMove(int, int, int, int) : boolean
  redo() : boolean
  reset() : void
  setBlackCastableKingside(boolean) : void
  setBlackCastableQueenside(boolean) : void
  setWhiteCastableKingside(boolean) : void
  setWhiteCastableQueenside(boolean) : void
  undo() : boolean
```



0:00:55

0:00:41

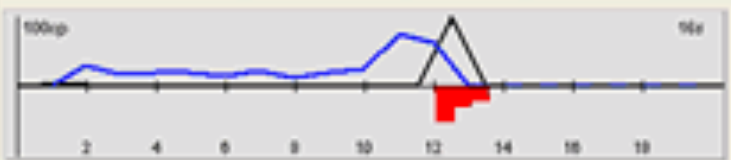


Freechess - Fritz 5.32  
Level=10<sup>th</sup> av. 26.06.2006

```
1634kB, ChesslibPowerBook2005.ctg, PentiumIII  
e-0 c5 0 2.f3 5 e6 0 3.d4 2 cxd4 0 4.fxd4  
3 a6 0 5.fxc3 3 fxc6 0 6.fxe3 2 gxc7 0 7.gd2 3  
f5 0 8.f3 2 f4 0 9.a3 3 f7 0 10.g4 4 h6 0  
11.0-0-0 12 b5 0 12.fxb1 10 b4 -0.50/10 16  
13.fxb4 0
```

Day #97  
1000\$

Fritz 5.32    ± (-0.28)    909kN/s  
13...fxb4 (1/45)    depth=11/11    0:00:12



```
± (-0.28) depth: 8/26 00:00:01 823kN  
13...fxb4 14.fg2 fxb8 15.h4 d5 16.g5 f6g8 17.exd5 fxd5 18.fxc  
= (-0.22) depth: 9/30 00:00:02 2268kN  
13...fxb4  
= (-0.26) depth: 9/33 00:00:04 3468kN  
13...fxb4 14.f14 f6g6 15.fxe3 f6a5 16.fxc6 dxc6 17.fxd4 e5 18.f  
± (-0.28) depth: 10/33 00:00:09 8528kN
```

"100,000 nodes per second and he just goes on thinking."



**New offer: 100000\$**

**I want to play online**

**I want to broadcast live games**

**Chess AI engines**

**Variants of Chess**

**iOS and Android compatible**

**Very robust solution**



# The Chess Variant Pages: Index

[What's New!](#) ([All Languages](#)) | [PLAY!](#) | [About Us](#) | [Help Us](#) | [Contact Us](#) | [Post Your Own Game](#)


Google    
 Search WWW  Search www.chessvariants.org

[Languages](#) | [Other Information](#) | [Advanced Search](#) | [Subject Index](#)

[0-9](#) - [A](#) - [B](#) - [C](#) - [D](#) - [E](#) - [F](#) - [G](#) - [H](#) - [I](#) - [J](#) - [K](#) - [L](#) - [M](#) - [N](#) - [O](#) - [P](#) - [Q](#) - [R](#) - [S](#) - [T](#) - [U](#) - [V](#) - [W](#) - [X](#) - [Y](#) - [Z](#)



[Recognized chess variants](#): What are considered to be the `best chess variants'?

 [Member favorites](#): What have members of these pages chosen as their favorites?

### Boards with an unusual shape



[Three dimensional chess variants](#): [Star Trek and others](#)



[Hexagonal chess variants](#)



[Round boards](#)



[Other boards with an unusual shape](#)



[Chess variants with usual equipment](#): Those variants that can be played with (mainly) a usual 8 by 8 board, and usual pieces.

- [Different moving pieces.](#)
- [Moving your opponents pieces.](#)
- [Multi-move variants.](#)
- [Rules about the board.](#)
- [Winning in a different way.](#)
- [Different opening setups](#)
- [Capturing in a different way](#)
- [Other variants](#)
- [Modest variants](#)



[Oriental chess variants](#): [Xiangqi](#) (Chinese Chess), [Shogi](#) (Japanese Chess), variants, other oriental chess variants



[Historic chess variants](#): The first predecessor of chess: Chaturanga, Shatranj, and other chess variants from ancient times



[Multi player variants](#): Chess with 3, 4, 6, 7 or any number of players.



[Small chess variants](#): Chess on board of size 4 by 8, 6 by 6, etc.



[Large chess variants](#): Chess on boards of 9 by 9 or larger.



[Chess variants with unequal armies](#): Variants where white and black have a different set of pieces.

### Other chess variants:

- [Wargames and Hierarchical games.](#)
- [Chess with cards.](#)
- [Chess with dice.](#)
- [Crossovers.](#) Variants that borrow ideas or rules from, or combine with other games.
- [Chess with incomplete information.](#) Kriegspiel and its variants.
- [Other variants.](#) Everything that didn't fit in another category: with several nice ones!

# Variants of chess game

Not 8\*8

Different moves

New pieces

New rules

New graphics

New AI engines

...

**In this monolithic code?!**

```
ChessDay1
  S main(String[]) : void
  board : PieceType[][]
  C ChessDay1()
  canMove(int, int, int, int) : boolean
  displayBoard() : void
  get50MoveRulePlyCount() : int
  getCapturedPieces() : List<PieceType>
  getCapturedPieces(boolean) : List<PieceType>
  getCurrentMoveNumber() : int
  getHistory() : String
  getMaterialCount(boolean) : int
  getPossibleMoves() : List<int[][]>
  getPossibleSquares(int, int) : List<int[][]>
  getThreats(int, int) : List<int[][]>
  getUnCapturedPieces(boolean) : List<PieceType>
  initBoard() : void
  isBlackCastleableKingside() : boolean
  isBlackCastleableQueenside() : boolean
  isBlockable() : boolean
  isCheck() : boolean
  isCheckmate() : boolean
  isDoubleCheck() : boolean
  isEnPassantFile(int) : boolean
  isStalemate() : boolean
  isWhiteCastleableKingside() : boolean
  isWhiteCastleableQueenside() : boolean
  isWhiteToPlay() : boolean
  recordMove(int, int, int, int) : boolean
  redo() : boolean
  reset() : void
  setBlackCastleableKingside(boolean) : void
  setBlackCastleableQueenside(boolean) : void
  setWhiteCastleableKingside(boolean) : void
  setWhiteCastleableQueenside(boolean) : void
  undo() : boolean
```



```
r n b q k b . r
p p p p p p p
. . . . n . .
. . . P . . .
. . . . N . .
P P P . P P P P
R N B Q K B . R
```

Thinking...

```
white KQkq
r . b q k b . r
p p p p p p p
. . n . . n . .
. . . P . . .
. . . . N . .
P P P . P P P P
R N B Q K B . R
```

My move is : Nc6  
White (3) :

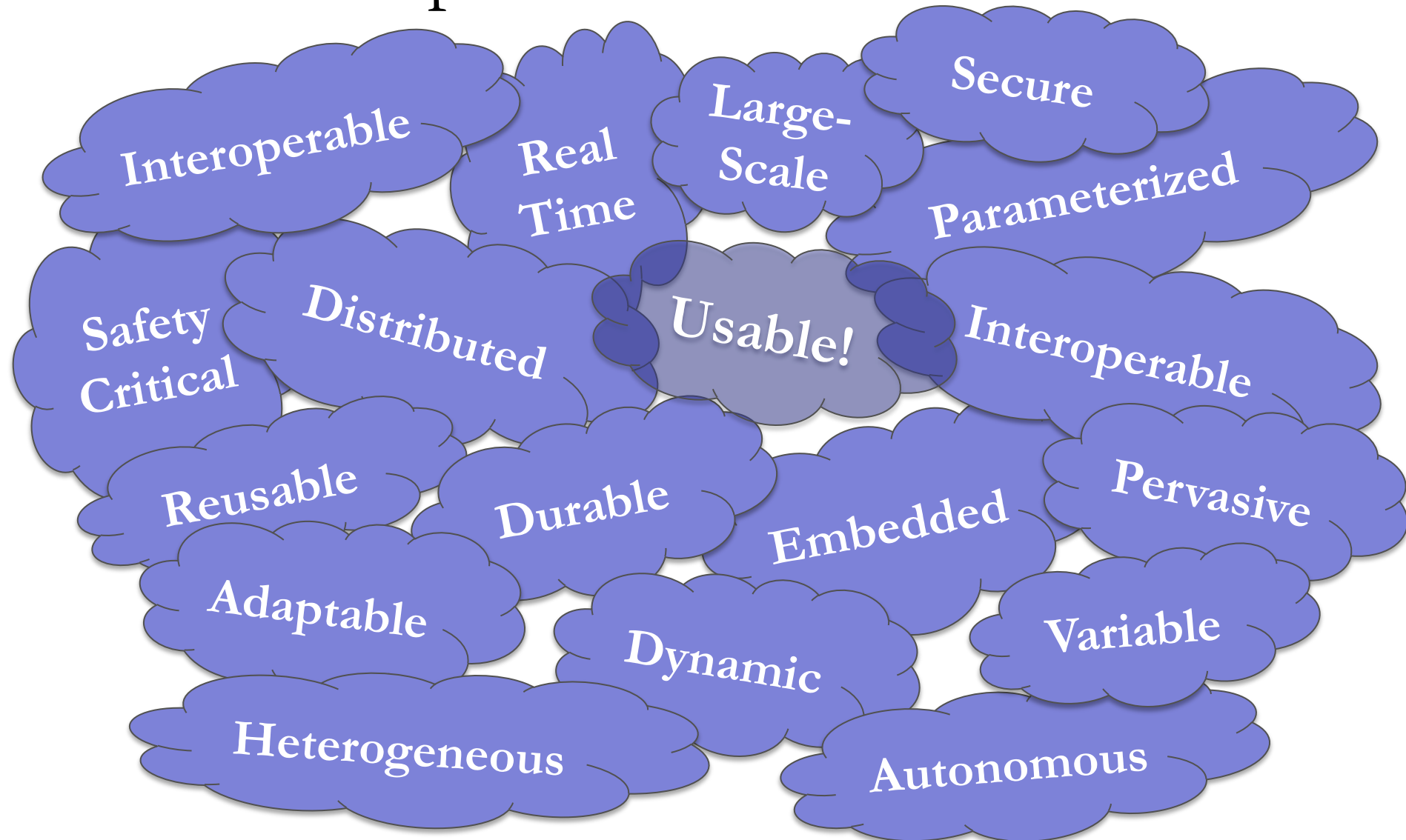


**Broadcasting?**

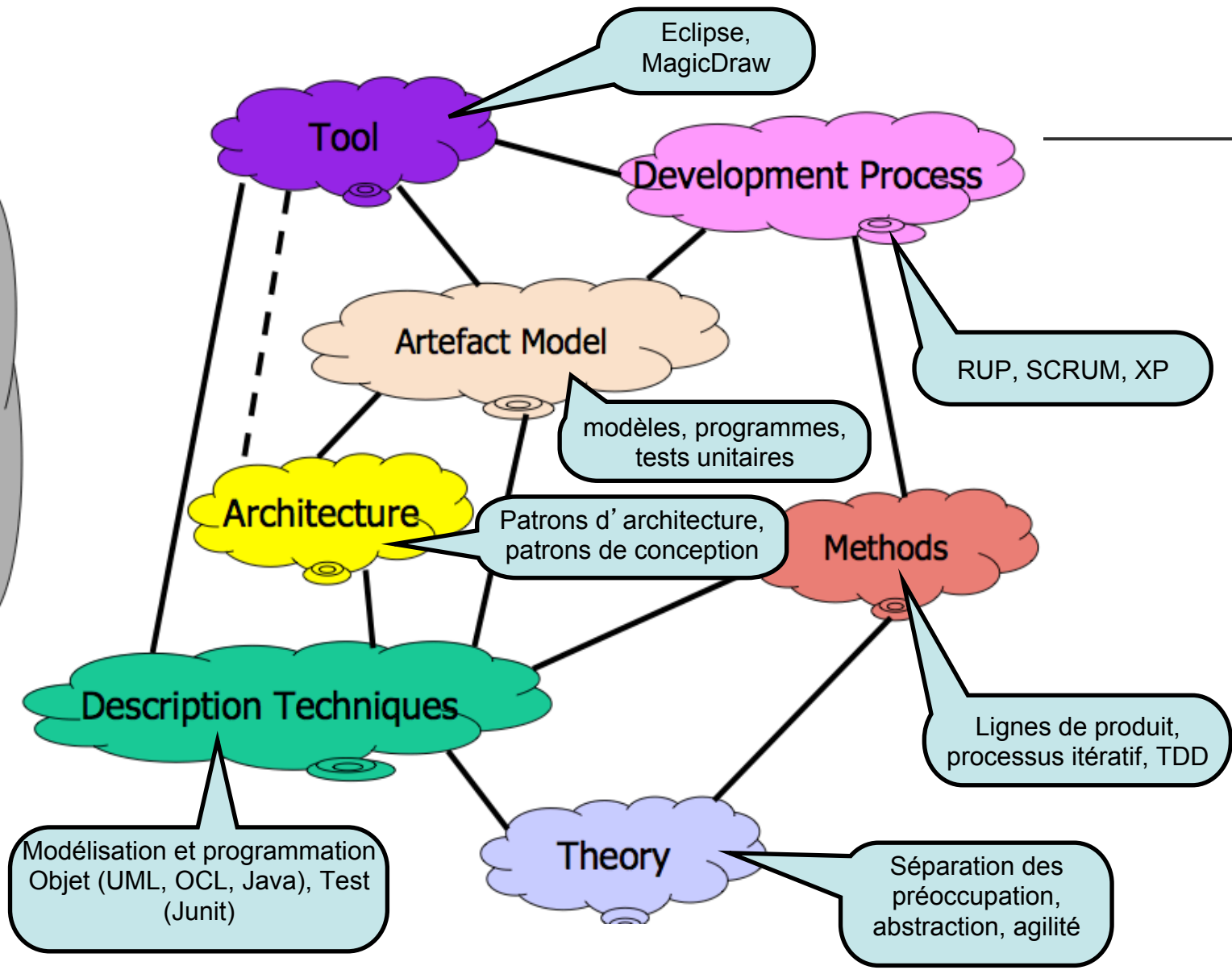
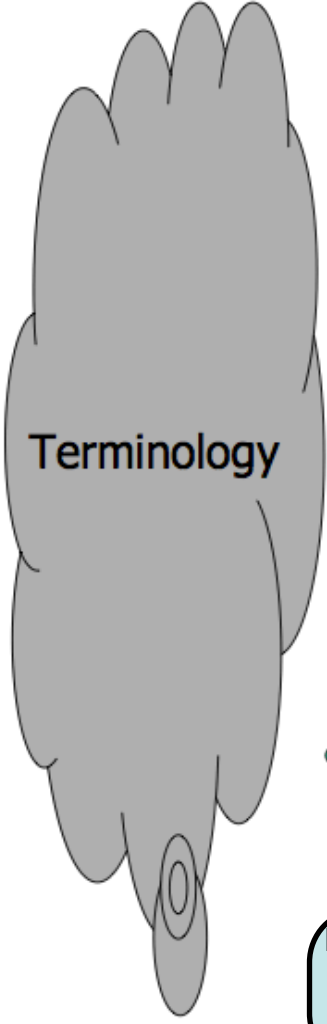
**1,3 billions of  
people live in  
India**

# Software Complexity: Even Applies for a Basic Example like Chess Game

---







# Conclusion

---

- **Le métier d'ingénieur logiciel est complexe:**  
*principes, techniques, méthodes, et outils pour décrire, implémenter, vérifier, gérer, et rendre opérationnel un système logiciel*
- Réponse de l'ingénierie du logiciel par l'utilisation de la modélisation
  - séparation des préoccupations
  - montée en abstraction
  - agilité des développements