Archeologia del Software

Ricostruire l'evoluzione di sistemi software

Marco D'Ambros e Michele Lanza

- Facoltà di informatica -
Università della Svizzera Italiana
Lugano (CH)

Università Bicocca, 27 Novembre 2006

Struttura della presentazione

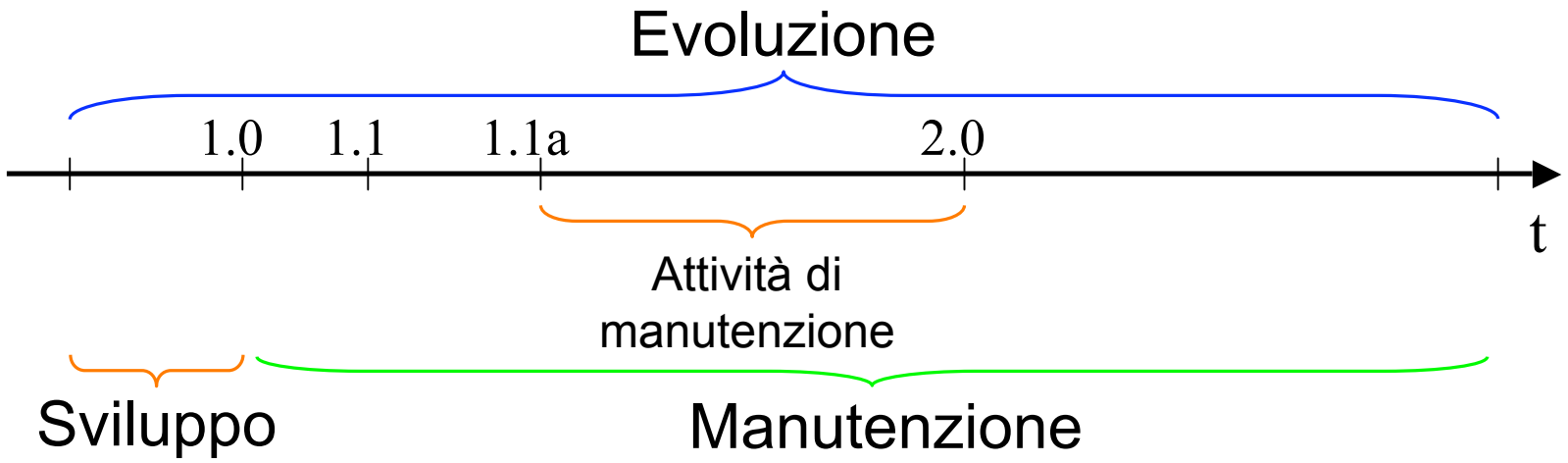
- Introduzione
  - Definizioni, motivazioni, problemi
- Il nostro approccio
  - Recupero dati e visualizzazione
  - BugCrawler e i diversi aspetti dell'evoluzione
    - Demo
  - Evolution radar e dipendenze logiche
    - Demo
- Conclusioni e direzioni da esplorare

Il software evolve!

- Il Software deve essere continuamente modificato per mantenere la sua utilità [1]
  ➡ Manutenzione del software
- Questo comporta una serie di problemi:
  - Crescita delle dimensioni
  - Crescita dell'entropia e della complessità
    - Peggioramento (erosione) di architettura, design e decomposizione in moduli
    - Crescita dell'interdipendenza fra moduli (coupling)
    - Decrescita della coesione

[1] M. Lehman, L. Belady. *A Model of Large Program Development.* IBM Systems Journal,1976.

Evoluzione ed analisi: Definizioni



- **Analisi dell'evoluzione**: analisi retrospettiva delle informazioni "storiche" disponibili su un sistema
- **Obiettivo**: identificare potenziali difetti nell'architettura, design e struttura logica del sistema
  - Successivamente soggetti a re-ingegnerizzazione

Perchè il termine archeologia?

- La struttura di un software è definita nel codice… e l'evoluzione?
- Tutto quello che è stato registrato durante lo sviluppo fa parte della storia del sistema
  - Sistemi di controllo di versione (cvs, subversion)
  - Bug tracking system (Bugzilla)
  - Mailing list
- Questi sistemi sono progettati per lo sviluppo, non per l'analisi retrospettiva
  - Le informazioni sono incomplete e non strutturate
- L'evoluzione va ricostruita, da cui il termine **Archeologia**
  - Processare le informazioni disponibili e salvarle in maniera strutturata
  - Analisi di log files cvs
  - Associare bug con entità software cvs

Motivazioni

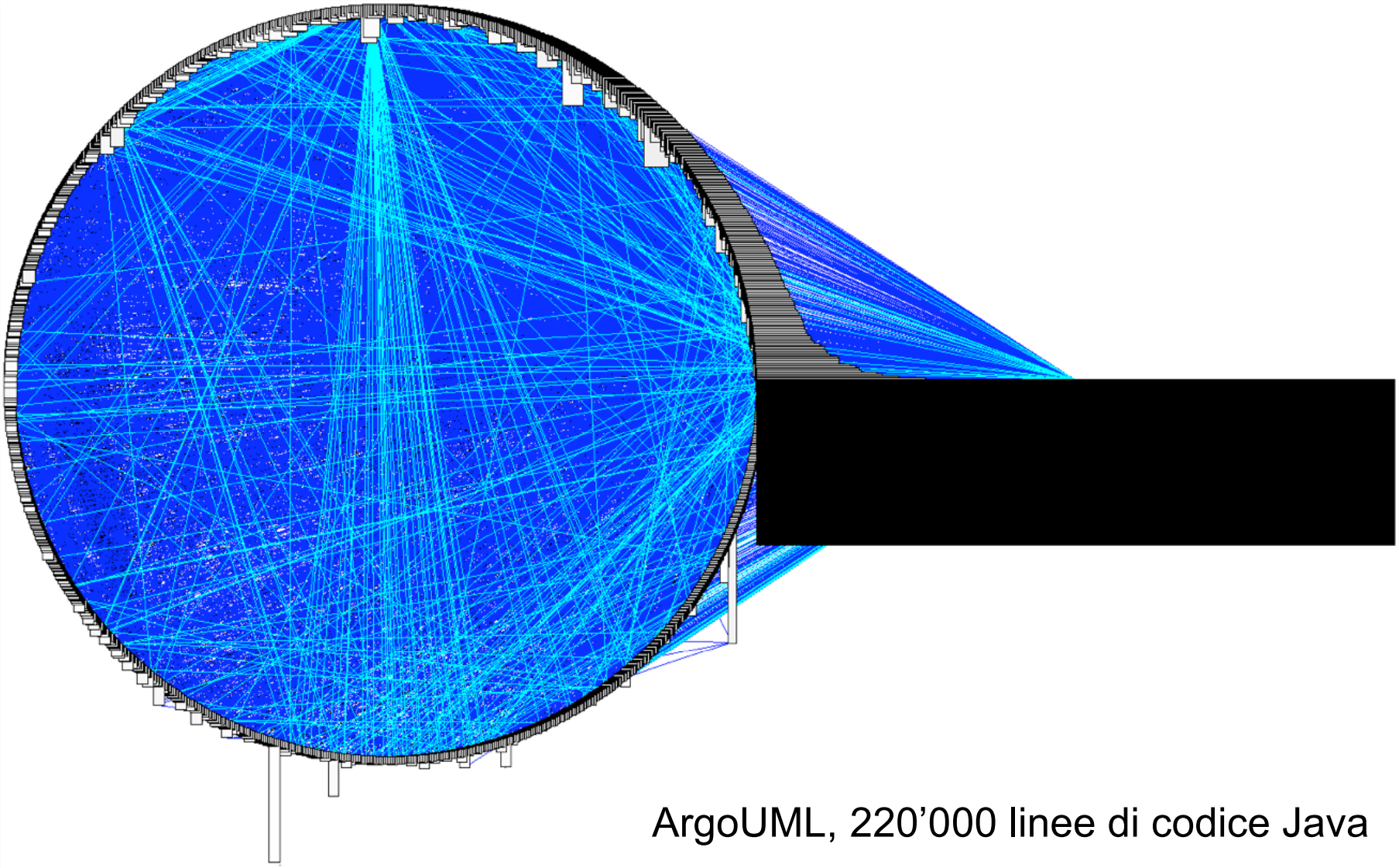
- Perchè è un problema rilevante?

$$\frac{\text{Costi legati a manutenzione e evoluzione}}{\text{Costo totale del software}} > 90\% \quad [2]$$

- Perchè studiare l'evoluzione per trovare difetti nella struttura?
  - Complementare all'analisi strutturale
  - Studiare più versioni dello stesso sistema fornisce maggiori risultati
  - Alcuni pattern sono visibili solo nell'evoluzione
- Ma quanto è (può diventare) complesso il software?

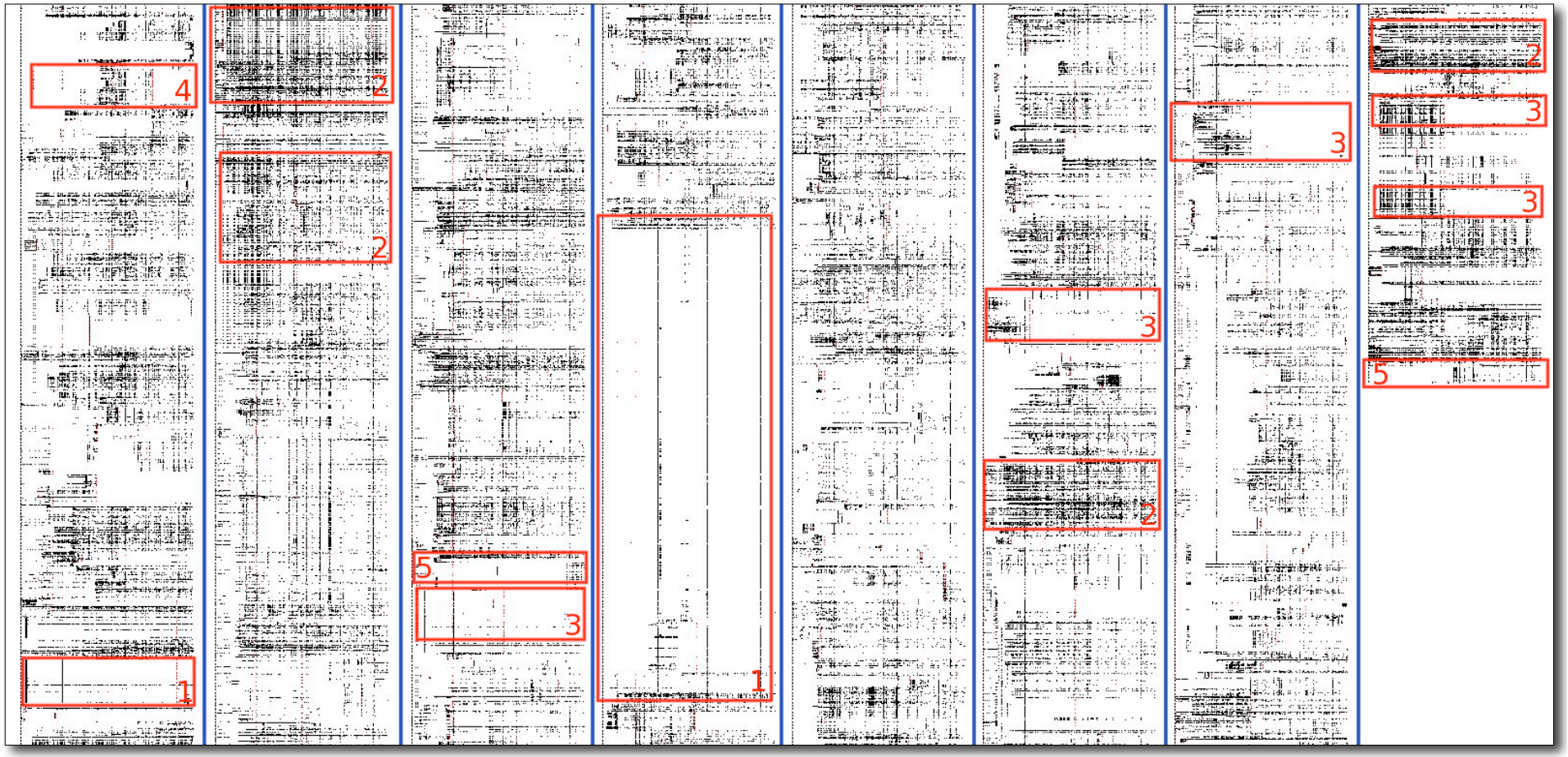
[2] L. Erlikh. *Leveraging legacy system dollars for e-business.* IT Professional 2, 3 (May. 2000)

Il software è complesso



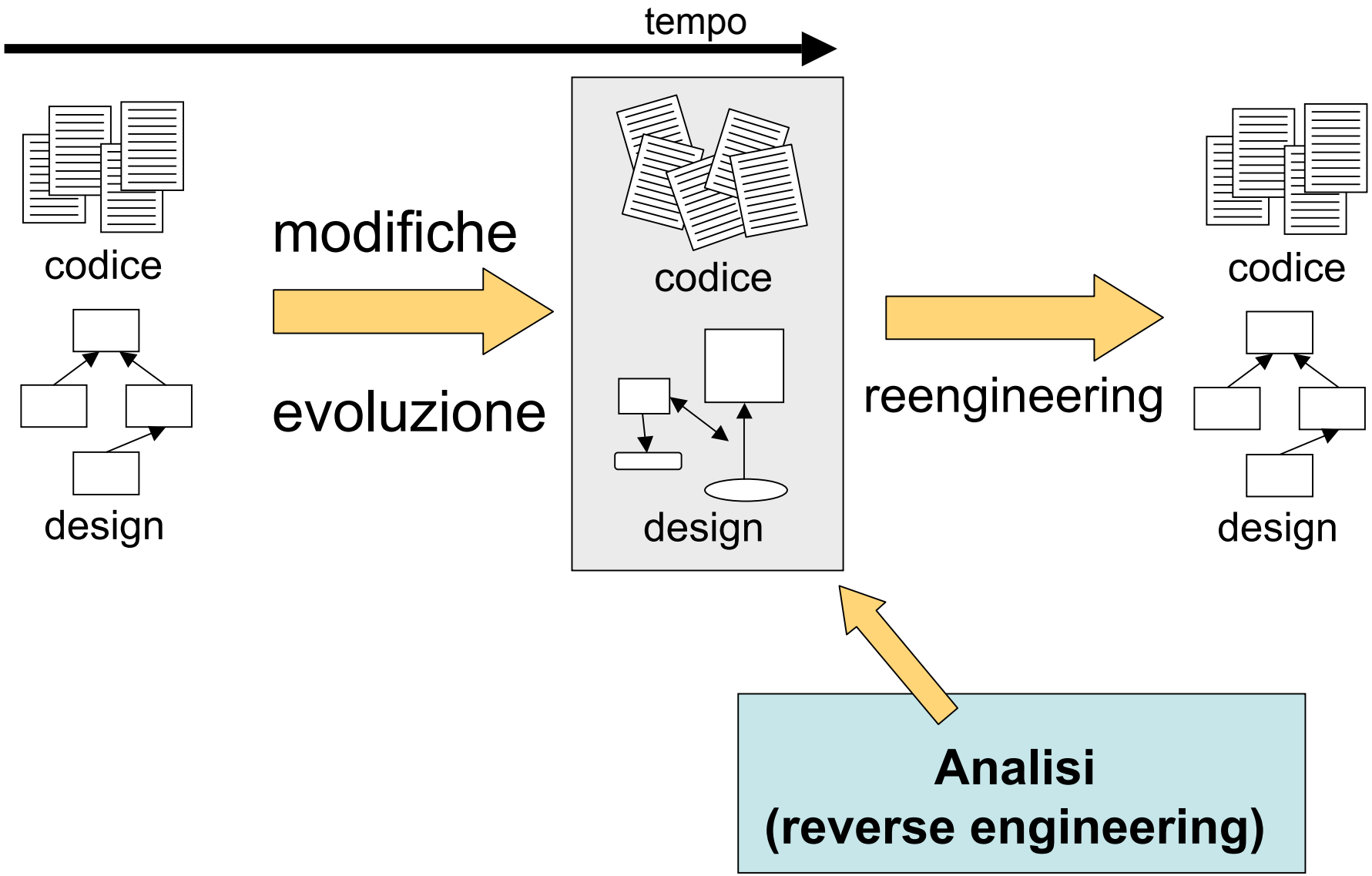
ArgoUML, 220’000 linee di codice Java

L'Evoluzione è ancora più complessa

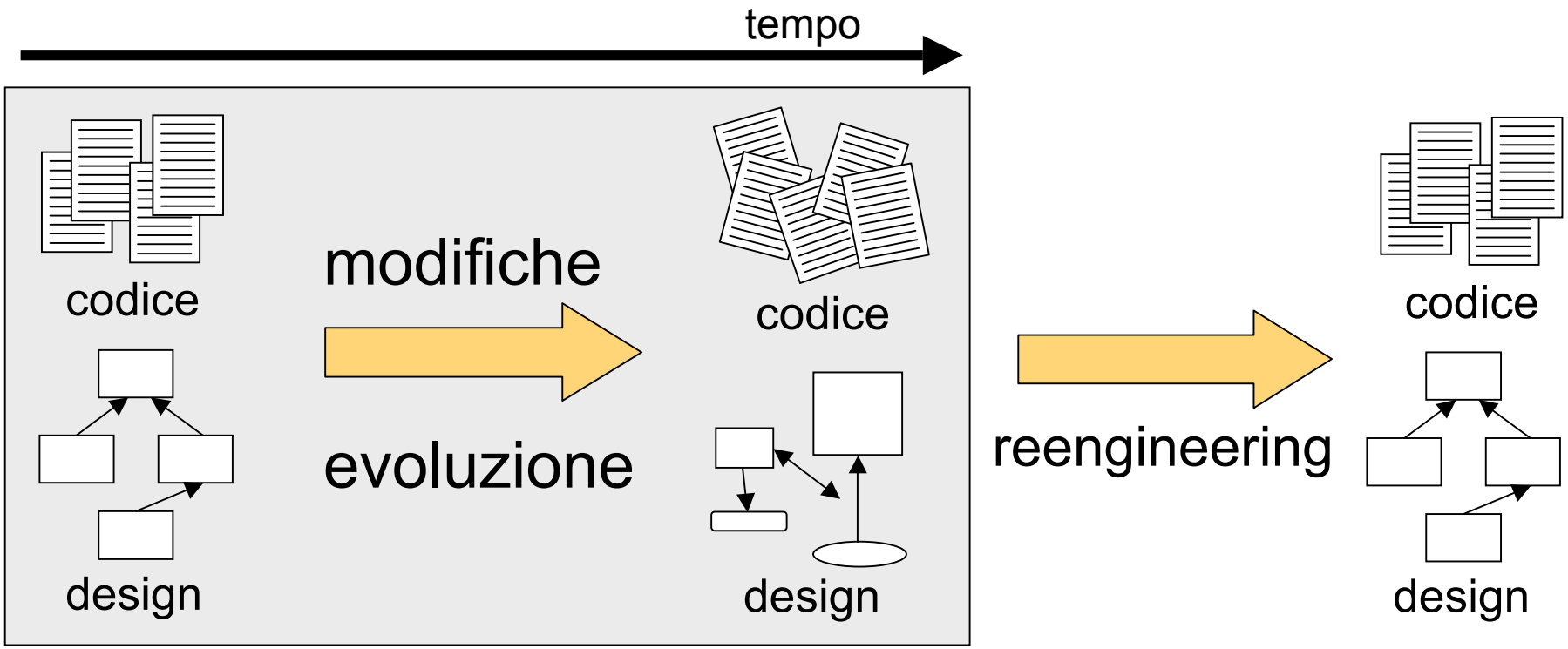


Mozilla: 3'000'000 di linee di codice C e C++, quasi un milione di modifiche (commits) fatte da centinaia di sviluppatori in più di sei anni

Riassumendo…

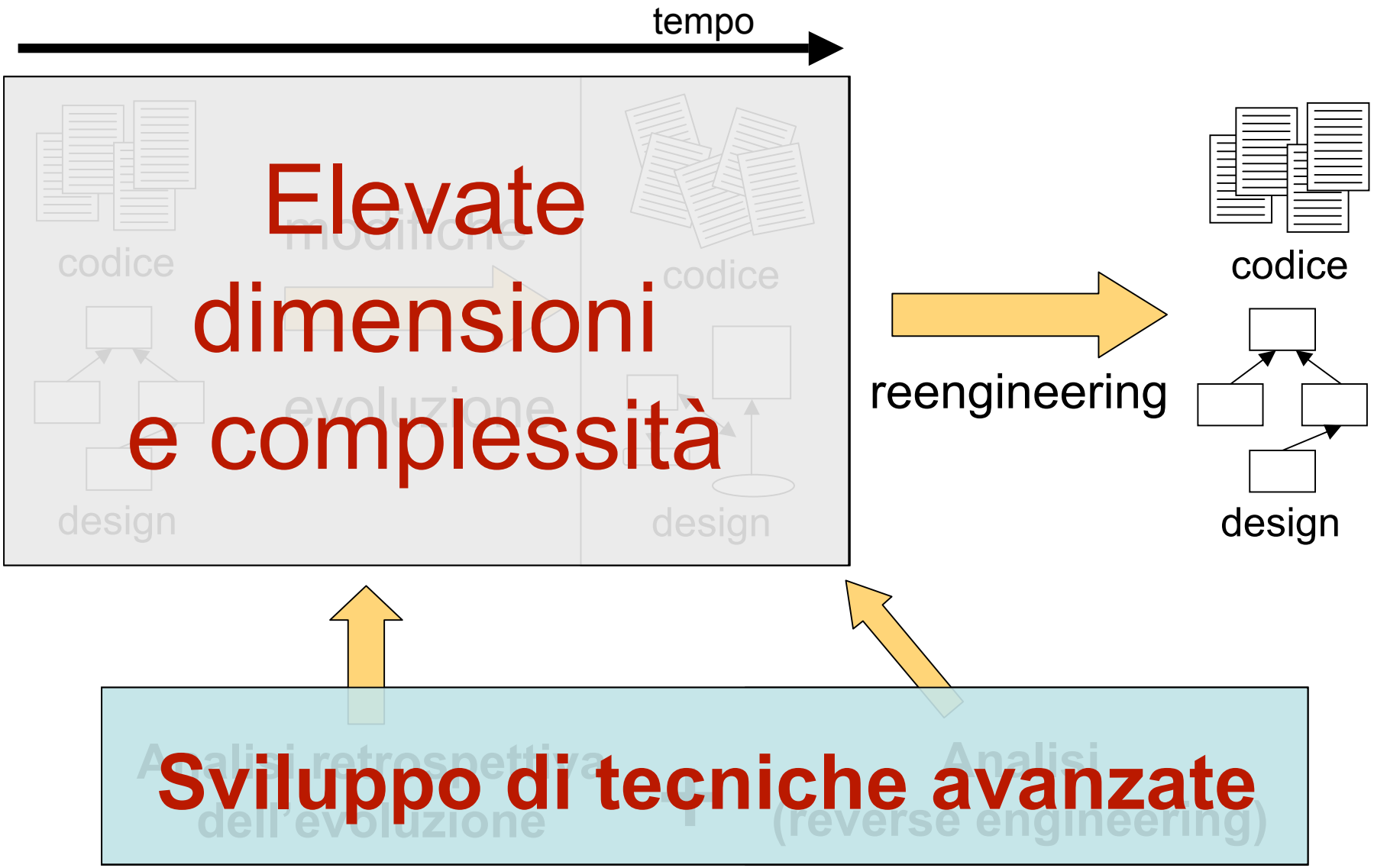


Riassumendo…



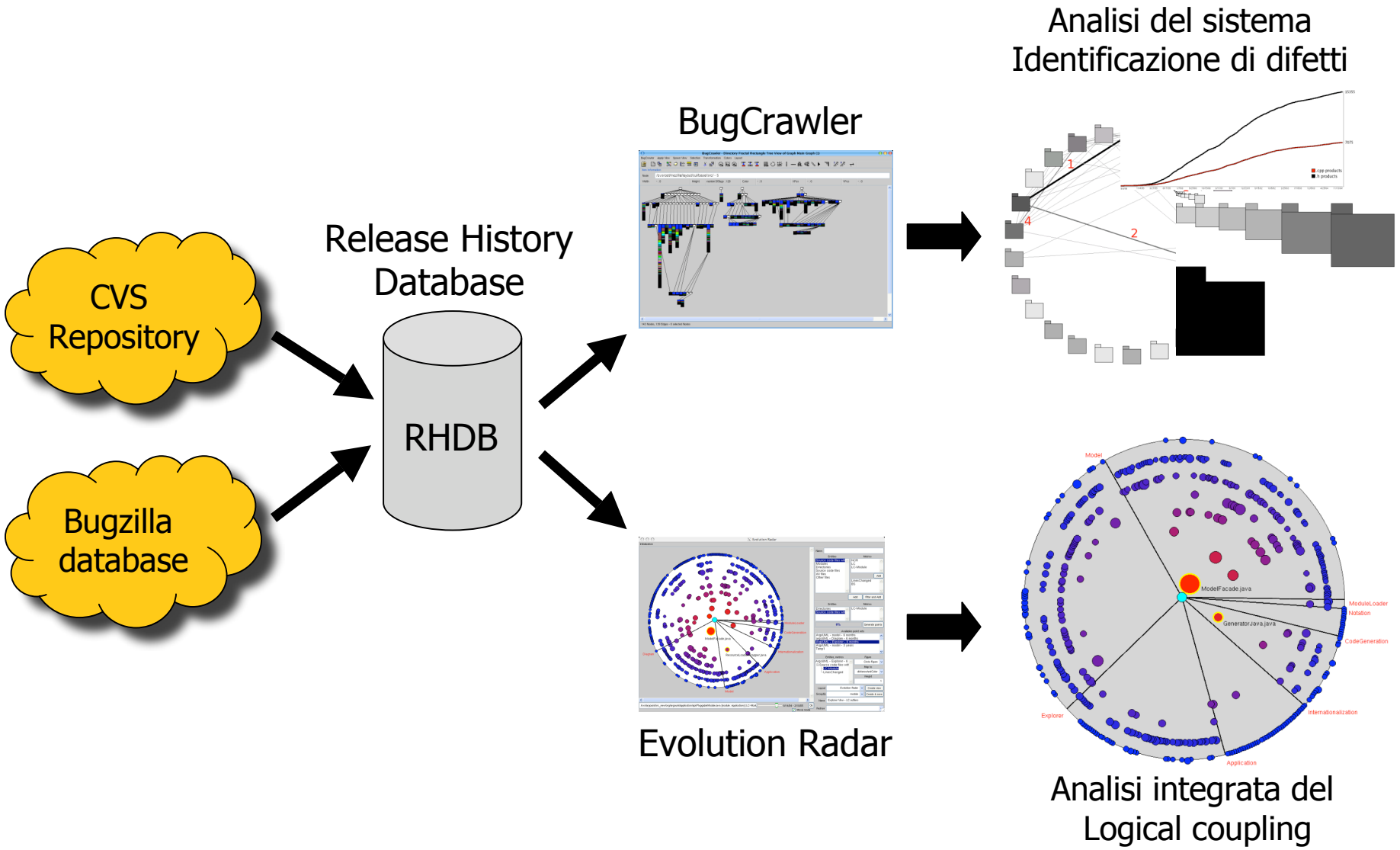
Analisi retrospettiva dell'evoluzione + **Analisi (reverse engineering)**

Riassumendo…



- Introduzione
  - Definizioni, motivazioni, problemi
- **Il nostro approccio all'evoluzione del software**
  - **Recupero dati e visualizzazione**
  - BugCrawler e i diversi aspetti dell'evoluzione
    - Demo
  - Evolution radar e dipendenze logiche
    - Demo
- Conclusioni e direzioni da esplorare

Il nostro approccio



CVS & Bugzilla

cvs log file

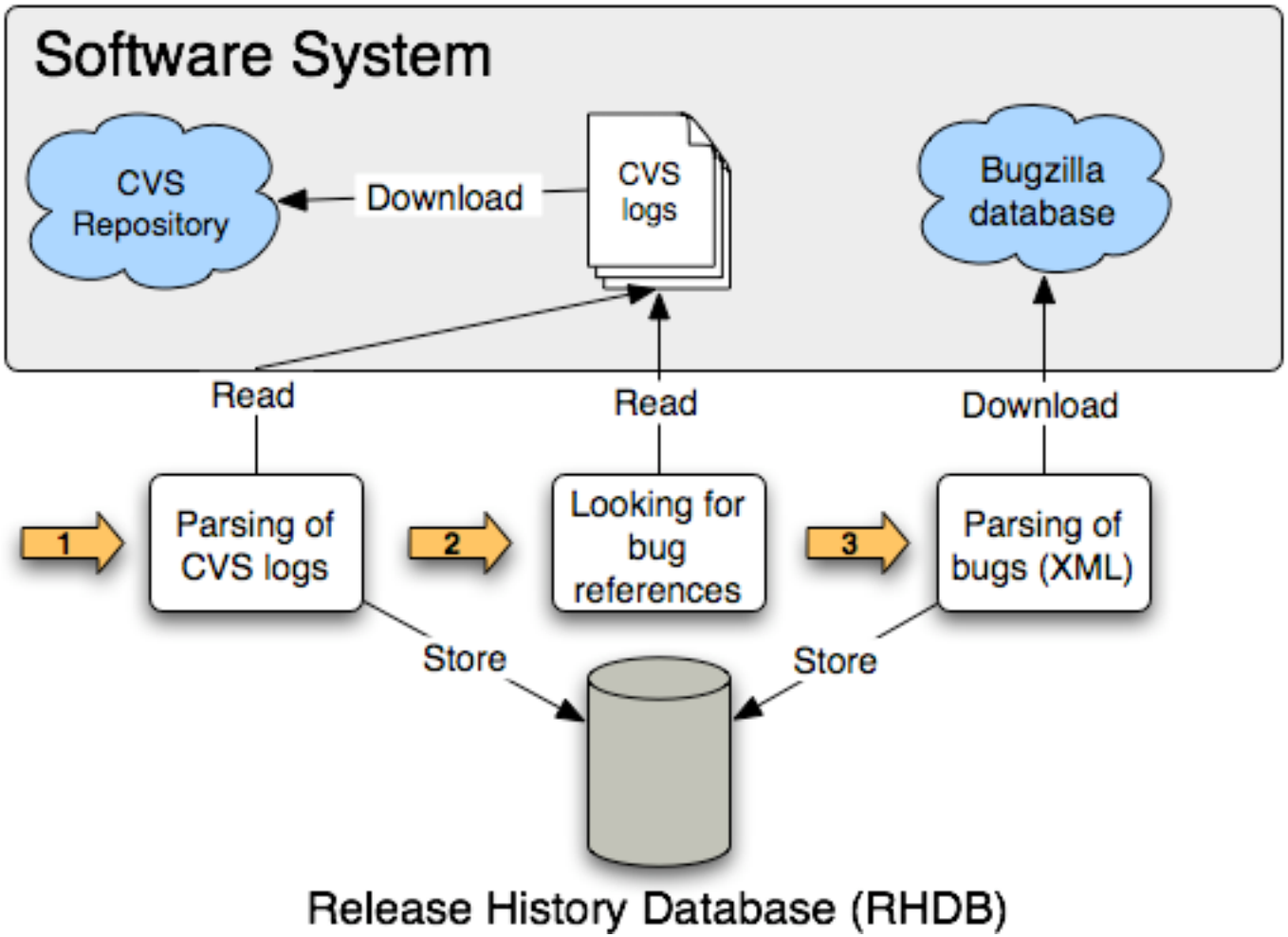
RCS file: layout/html/style/src/nsCSSFrameConstructor.cpp,v
Working file: nsCSSFrameConstructor.cpp
head: 1.804
branch:
symbolic names:
  MOZILLA_1_3a_RELEASE: 1.800
  ...
keyword substitution: kv
total revisions: 976; selected revisions: 976
description:
----
revision 1.804
date: 2002/12/13 20:13:16; author: doe@netscape.com;
state: Exp; lines: +15 -47
Don't set NS_BLOCK_SPACE_MGR and
NS_BLOCK_WRAP_SIZE on ...

...
----
revision 1.638
date: 2001/09/29 02:20:52; author: doe@netscape.com;
state: Exp; lines: +14 -4; branches: 1.638.4;
bug 94341 keep a separate pseudo frame list for a new pseudo
block or inline frame ...

Bugzilla bug report

- **Bug id**: 94341
- **Bug status**: assigned
- **Product:** Firefox
- **Component:** Layout
- **Depends_on:** 34859
- **Blocks:** 56540, 36042, 41245
- **Bug severity**: critical
- **Bug priority**: P3
- **Target milestone:** 2.0

Popolare il RHDB



Esplosione della quantità di dati

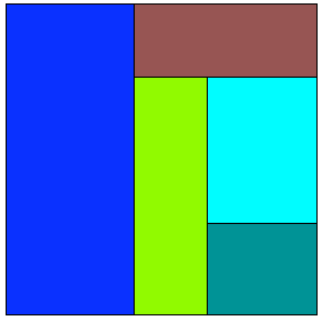
Software	#Linee di codice	#file sorgenti	#commits	#bug reports	#bug references	#anni di storia
Mozilla	3.000.000	30.000	818.000	25.000	164.000	6
Gimp	500.000	4.300	170.000	1.700	15.000	6

- Le tecniche di supporto all'analisi devono
  - Gestire grandi quantità di dati (scalabili)
  - Facilitare l'interpretazione dei dati
    - Aggregare informazioni alzando il livello di astrazione
    - Fornire i dettagli "on-demand"
    - Supportare un'analisi graduale

Visualizzazione

- Strumento di **supporto** all'analisi
  - Facilità l'interpretazione di grandi quantità di dati
  - Per analizzare dati "sconosciuti" (non possile con query)
  - Permette di identificare *outliers* nel contesto
- Deve essere **interattiva**
  - Le figure rappresentano entità e relazioni che vogliamo poter ispezionare e analizzare in ogni momento
- Un mito da sfatare
  - Il nostro obiettivo non è fare visualizzazioni "attraenti" ma **efficienti** che facilitino l'interpretazione dei dati

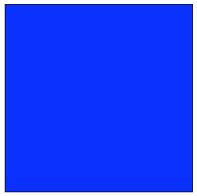
Esempio: Fractal Figure [3]



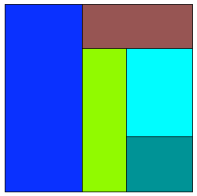
warren
dcone
gerv
… …

Area di ogni rettangolo proporzionale al numero di commit fatto dall'autore corrispondente

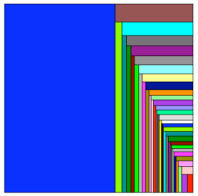
- Sviluppo di entità dal punto di vista degli autori
  - Quanti hanno lavorato su un artefatto?
  - Come lo sviluppo è distribuito fra gli autori?
- Permette di definire un "*pattern language*"



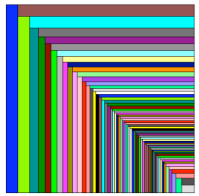
Singolo autore



Pochi autori bilanciati



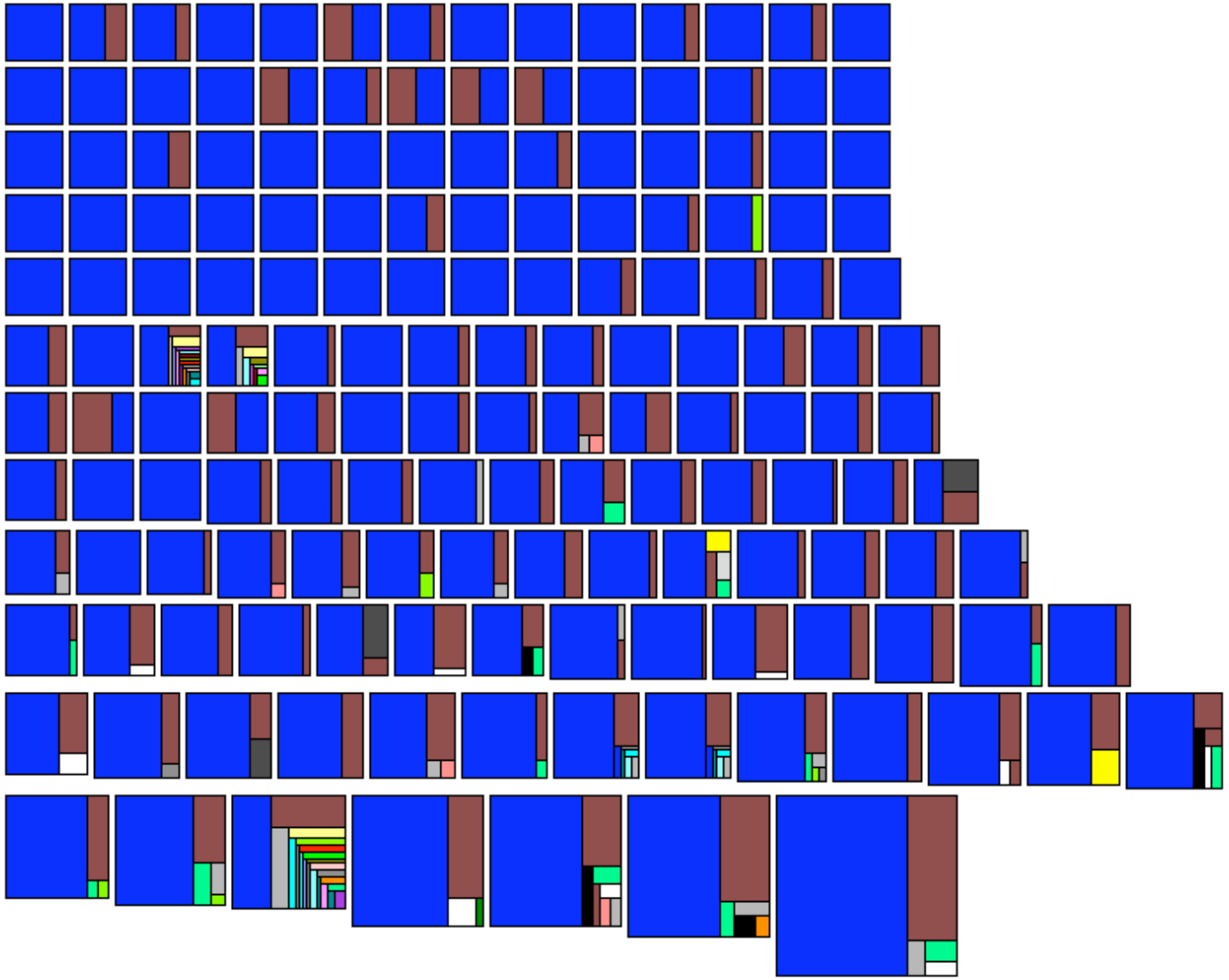
Autore principale



Molti autori bilanciati

[3] D'Ambros, Lanza, Gall. *Fractal Figure: Visualizing Development Effort for cvs Entities.* Proceedings of VISSOFT 2005. IEEE CS Press, 2005

Fractal Figure applicata (Gimp)

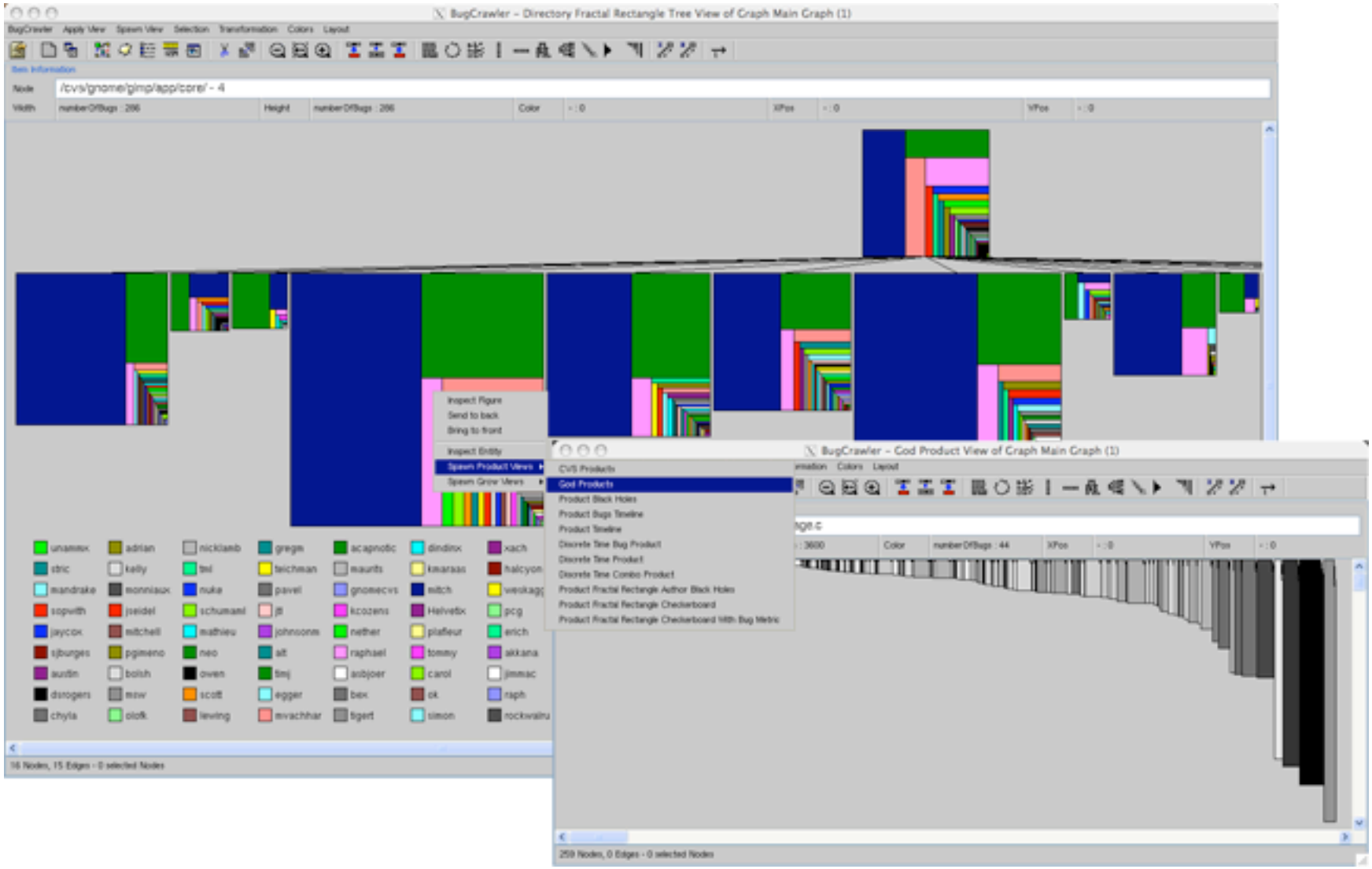


hans
neo
adrian
mitch
scott
yosh
pcg
amundson
akkana
raph
sopwith
...

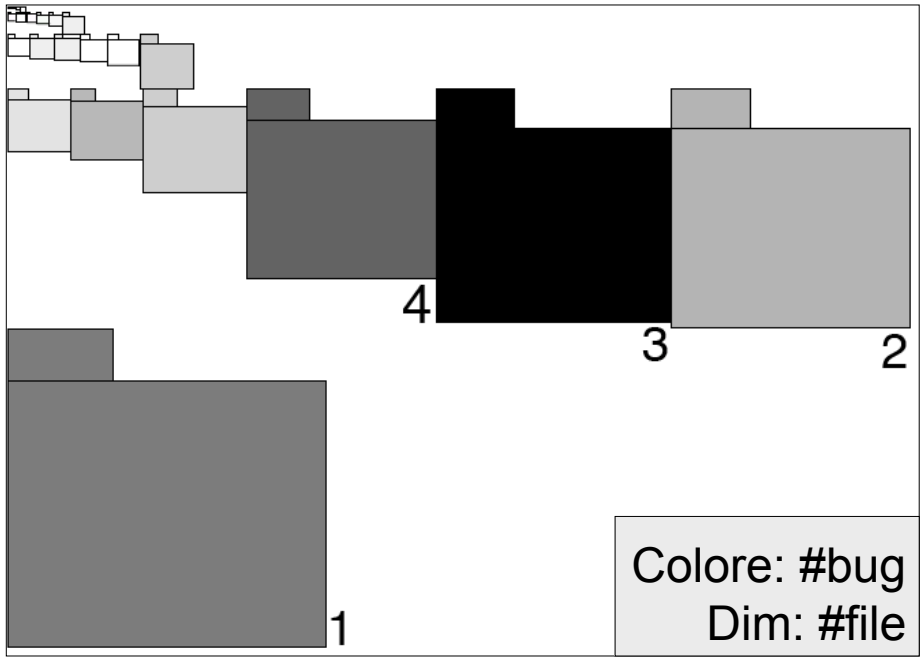
Dimensione: File
#bugs

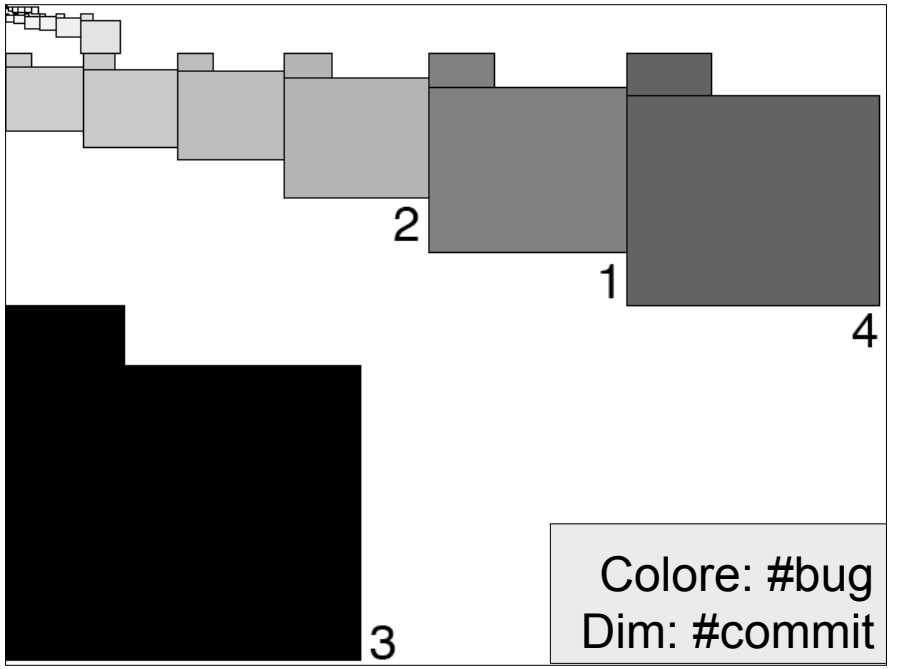
- Introduzione
  - Definizioni, motivazioni, problemi
- Il nostro approccio all'evoluzione del software
  - Recupero dati e visualizzazione
  - **BugCrawler e i diversi aspetti dell'evoluzione**
    - **Demo**
  - Evolution radar e dipendenze logiche
    - Demo
- Conclusioni e direzioni da esplorare

Demo di BugCrawler



Ottenere una visione d'insieme

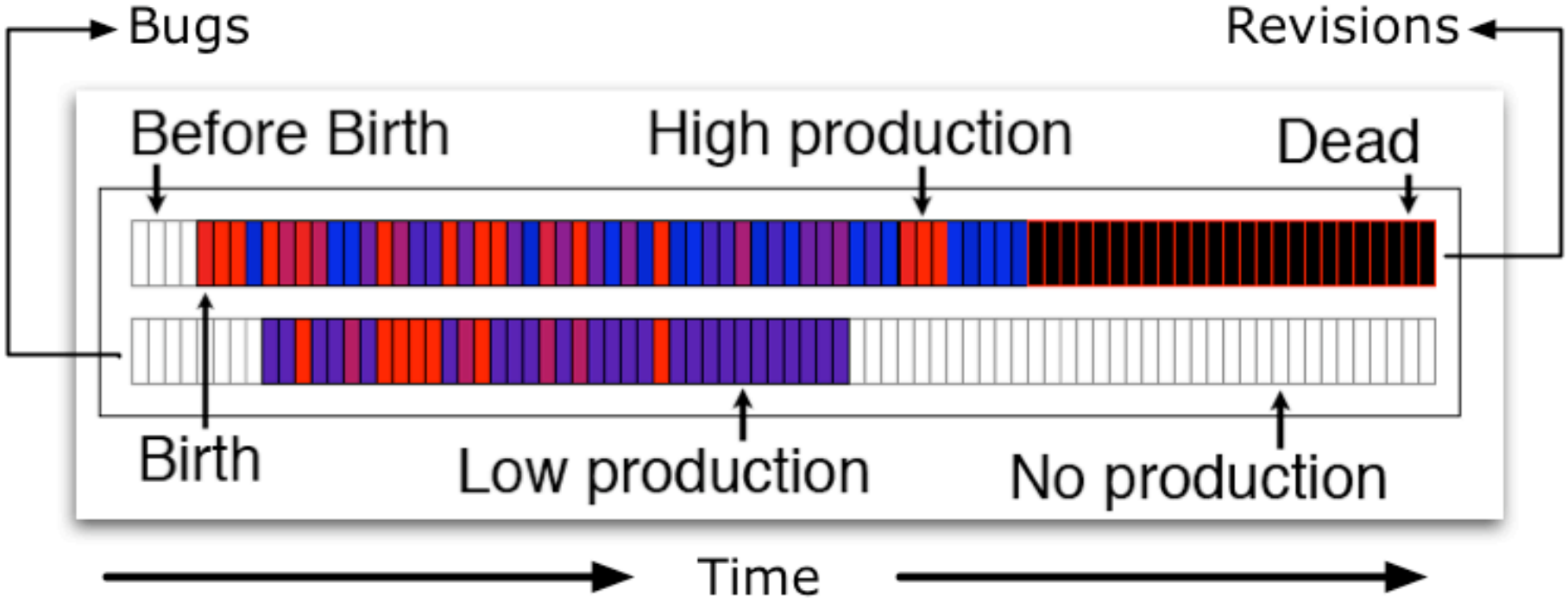




- Di supporto per capire:
  - Moduli chiave, dimensioni relative, dove sono concentrati i bug
  - Dove focalizzare l'analisi dettagliata (in piccolo)

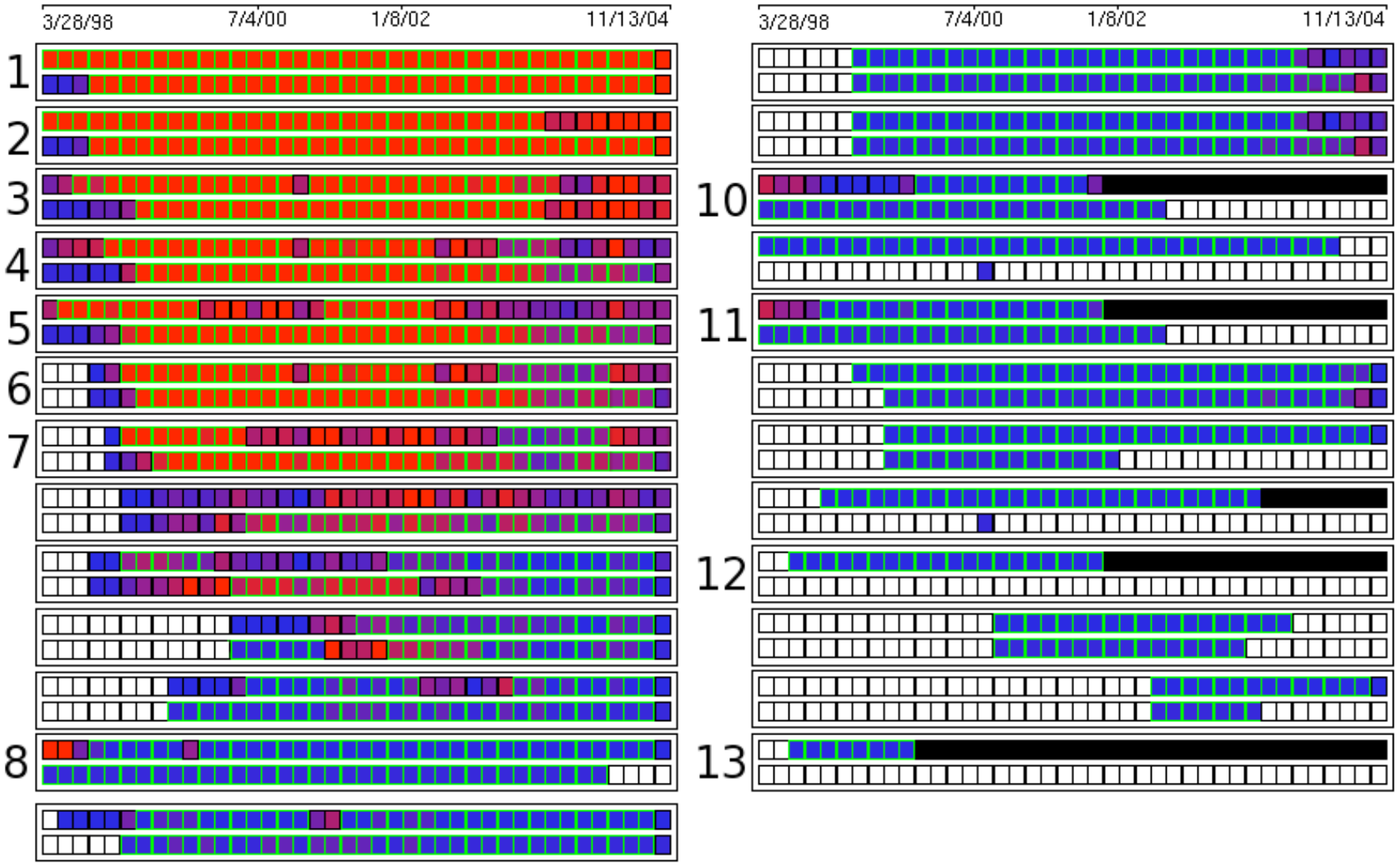
Considerare il tempo

- Figura ad-hoc: **Discrete Time Figure** [4]



[4] M. D'Ambros, M. Lanza. *Software Bugs and Evolution: A Visual Approach to Understand their Relationship.* Proceedings of CSMR 2006. IEEE CS Press, 2006.

Evoluzione temporale moduli di mozilla

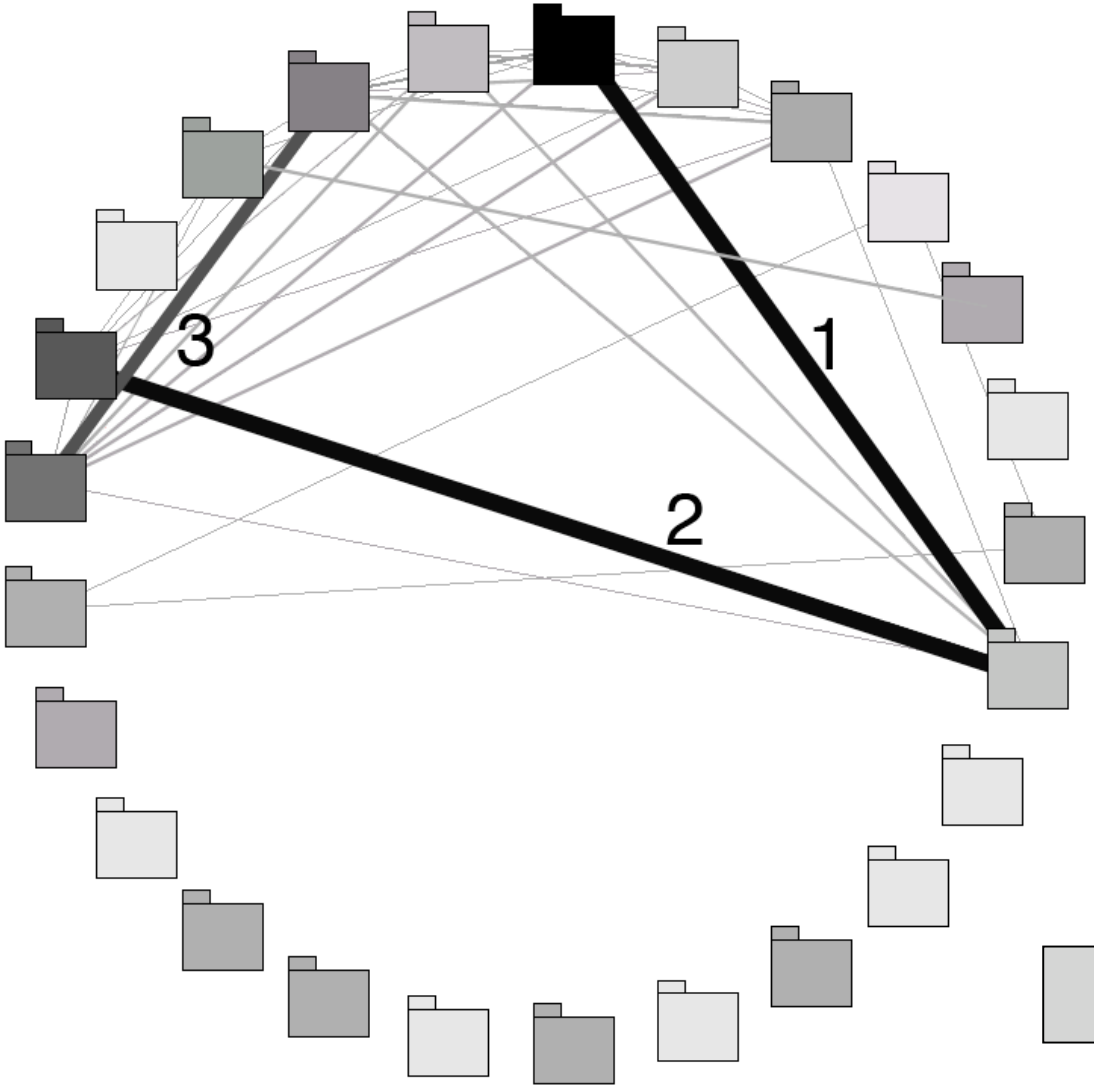


Dipendenze tra moduli

- Un bug condiviso tra due moduli rappresenta una dipendenza fra di essi
  - Può rappresentare un difetto architetturale (de-modularizzazione)
  - Non visibile nella struttura del sistema ma solo nell'evoluzione

  ➡ Più problematica di dipendenze statiche e strutturali
  - Tanto più forte quanti più sono i bug condivisi

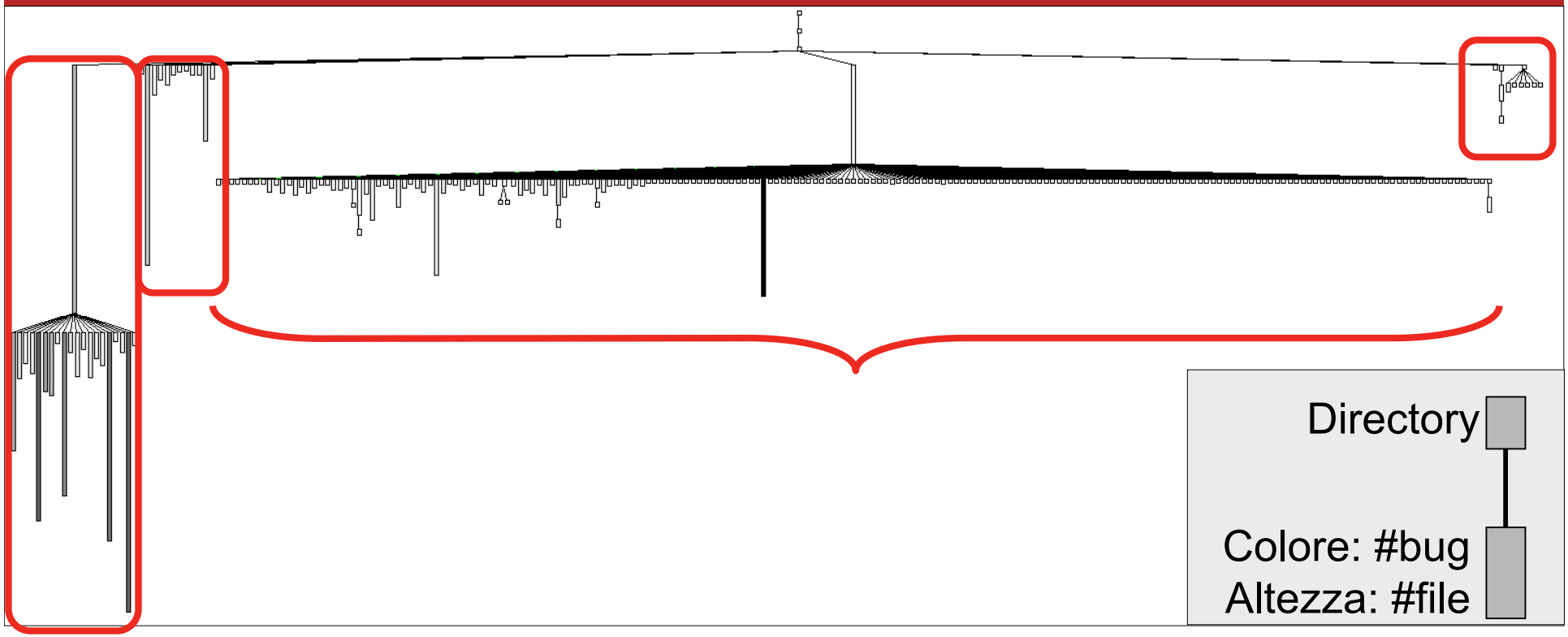
Dipendenze fra i moduli di mozilla



- Di supporto per identificare:
  - Coppie di moduli con le dipendenze più forti
  - Moduli con molte dipendenze

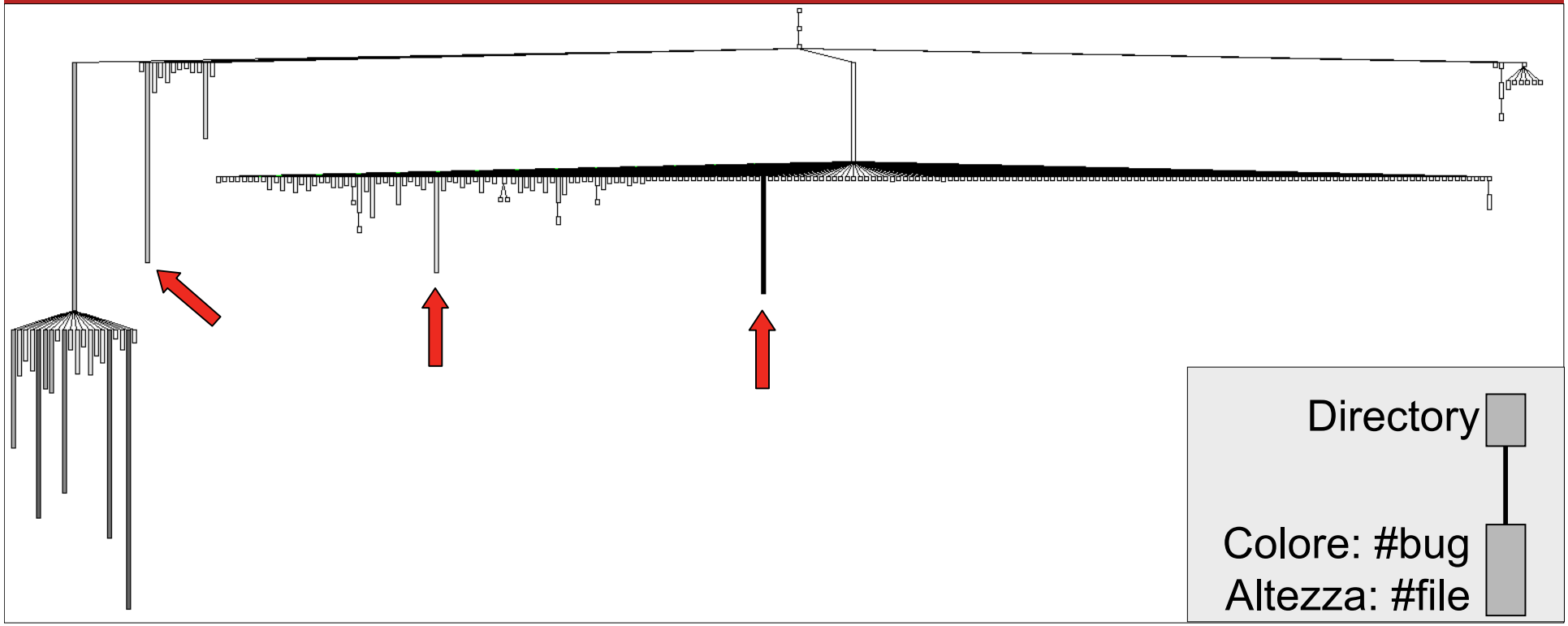
Colore: #bug

Studiare le gerarchie di directory



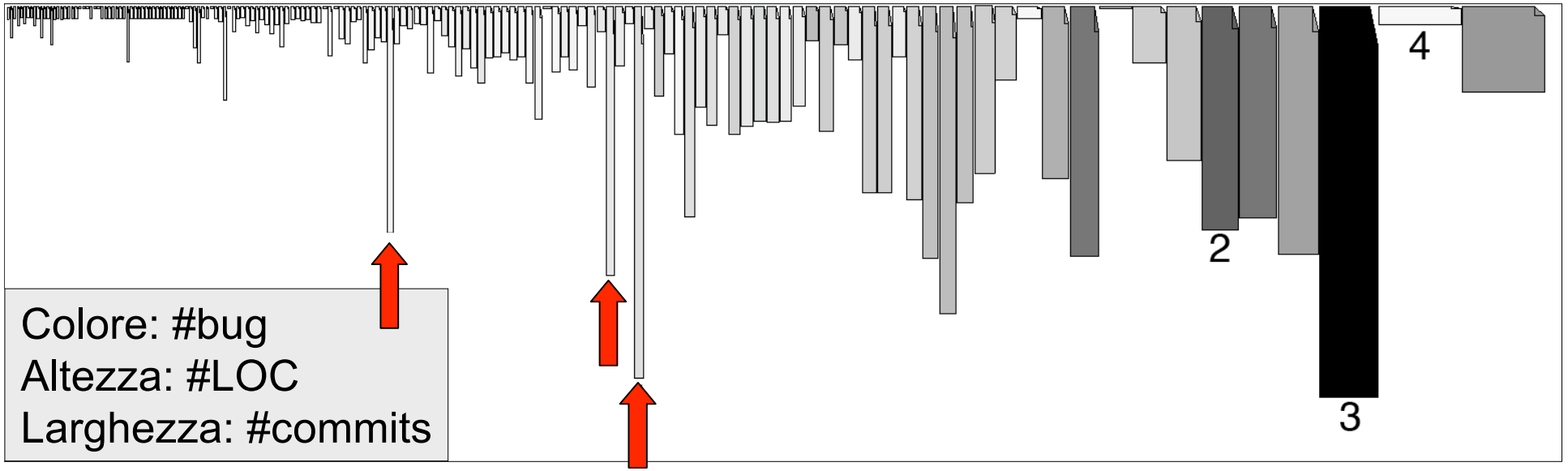
- Di supporto per capire:
  - La struttura del sistema o di un suo modulo
  - Quali sono le gerarchie di directory principali

Studiare le gerarchie di directory



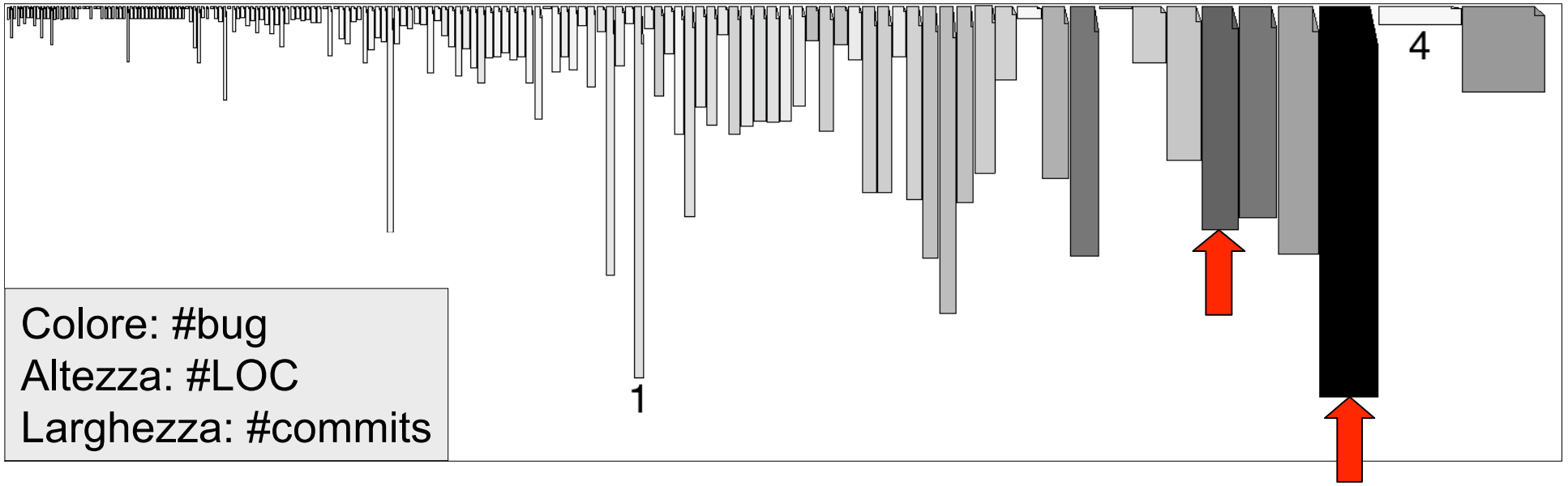
- Di supporto per capire:
  - La struttura del sistema o di un suo modulo
  - Quali sono le gerarchie di directory principali
  - Dove si trovano la maggioranza dei bugs
  - Dove si trovano gli outliers (nel contesto della gerarchia)

Studiare il contenuto delle directory



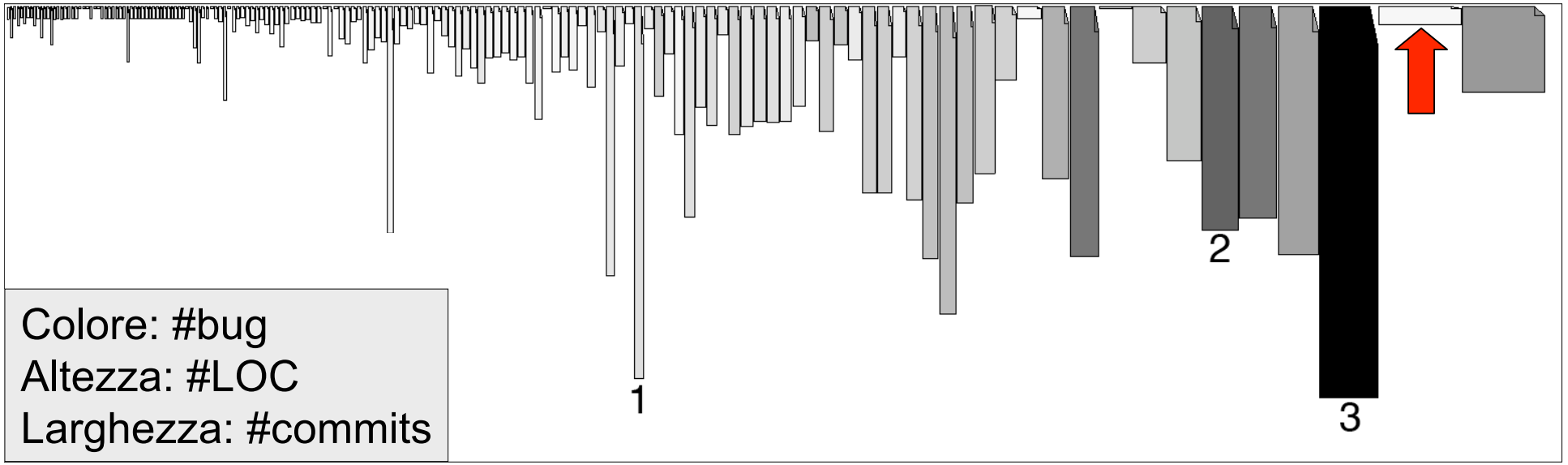
- Di supporto per identificare outliers:
  - Alti e stretti: Molte LOC e pochi commits (copy-paste, inseriti dopo nel repository)

Studiare il contenuto delle directory



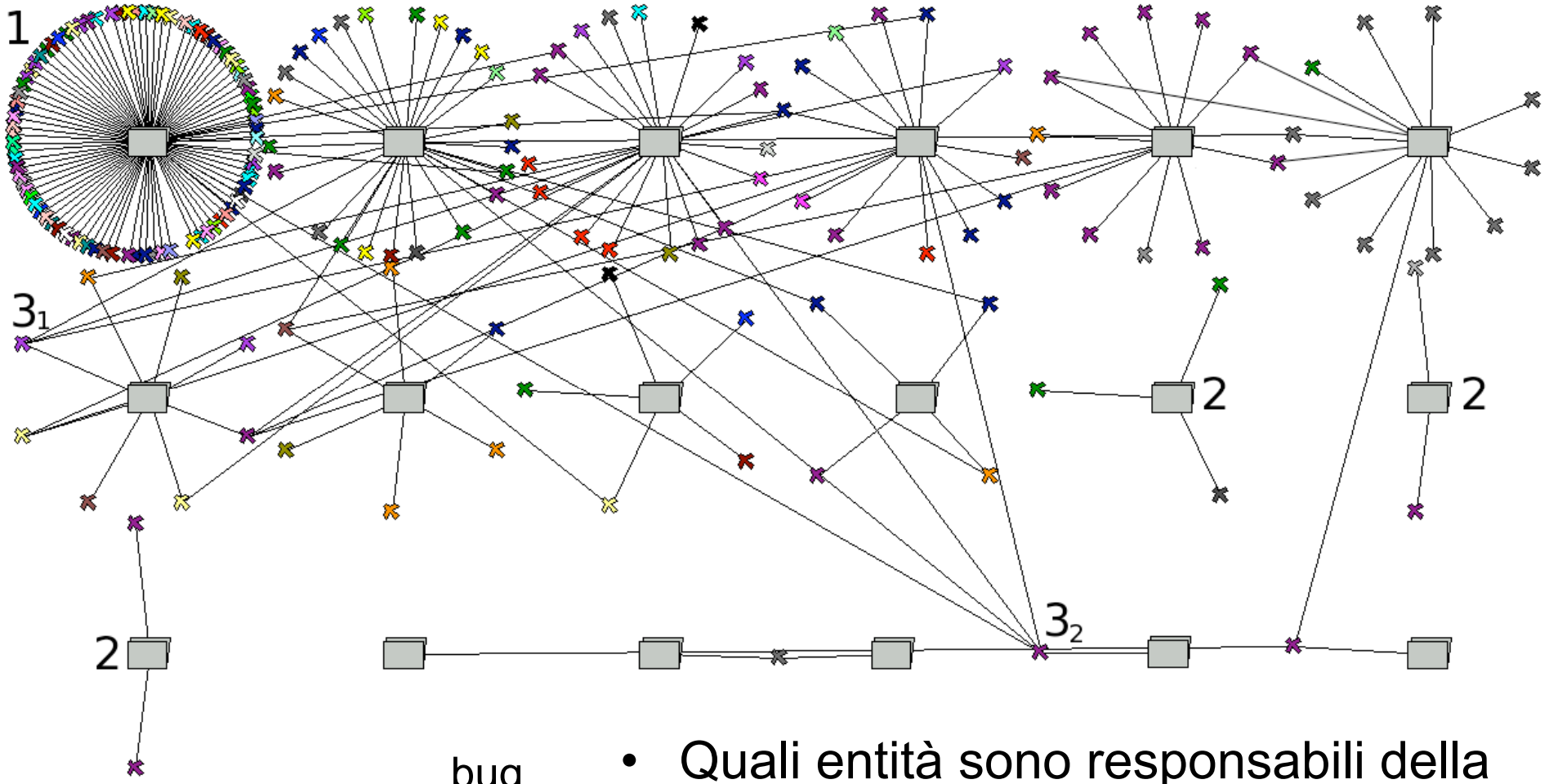
- Di supporto per identificare outliers:
  - Alti e stretti: Molte LOC e pochi commits (copy-paste, inseriti dopo nel repository)
  - Alti e scuri: Molte LOC e bug (serve refactoring)

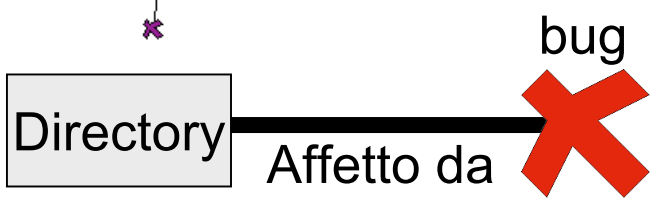
Studiare il contenuto delle directory



- Di supporto per identificare outliers:
  - Alti e stretti: Molte LOC e pochi commits (copy-paste, inseriti dopo nel repository)
  - Alti e scuri: Molte LOC e bug (serve refactoring)
  - Bassi e larghi: decrescono in dimensione (perdono responsabilità)

Dettagli delle dipendenze fra moduli





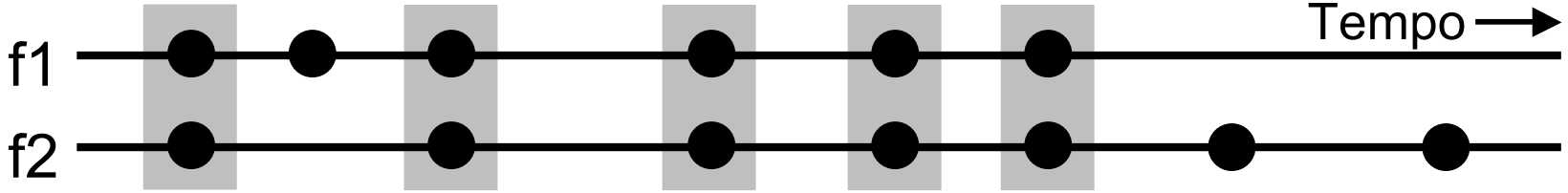
- Quali entità sono responsabili della dipendenza fra moduli in termini di bug condivisi?

BugCrawler: riassumendo…

- Fornisce un'insieme di viste a diversi livelli di dettaglio per supportare l'analisi dell'evoluzione del software in maniera **metodologica**
  - "*in grande*" per capire la struttura del sistema in termini di moduli
  - "*in piccolo*" per capire la struttura interna dei moduli
- Le viste mostrano differenti aspetti evolutivi sulle medesime entità
- Il tool supporta l'analisi consentendo di "saltare" da un aspetto all'altro e di aumentare e diminuire il livello di dettaglio (dal "grande" al "piccolo" e viceversa)

- Introduzione
  - Definizioni, motivazioni, problemi
- Il nostro approccio all'evoluzione del software
  - Recupero dati e visualizzazione
  - BugCrawler e i diversi aspetti dell'evoluzione
    - Demo
  - **Evolution radar e dipendenze logiche**
    - **Demo**
- Conclusioni e direzioni da esplorare

Dipendenze logiche

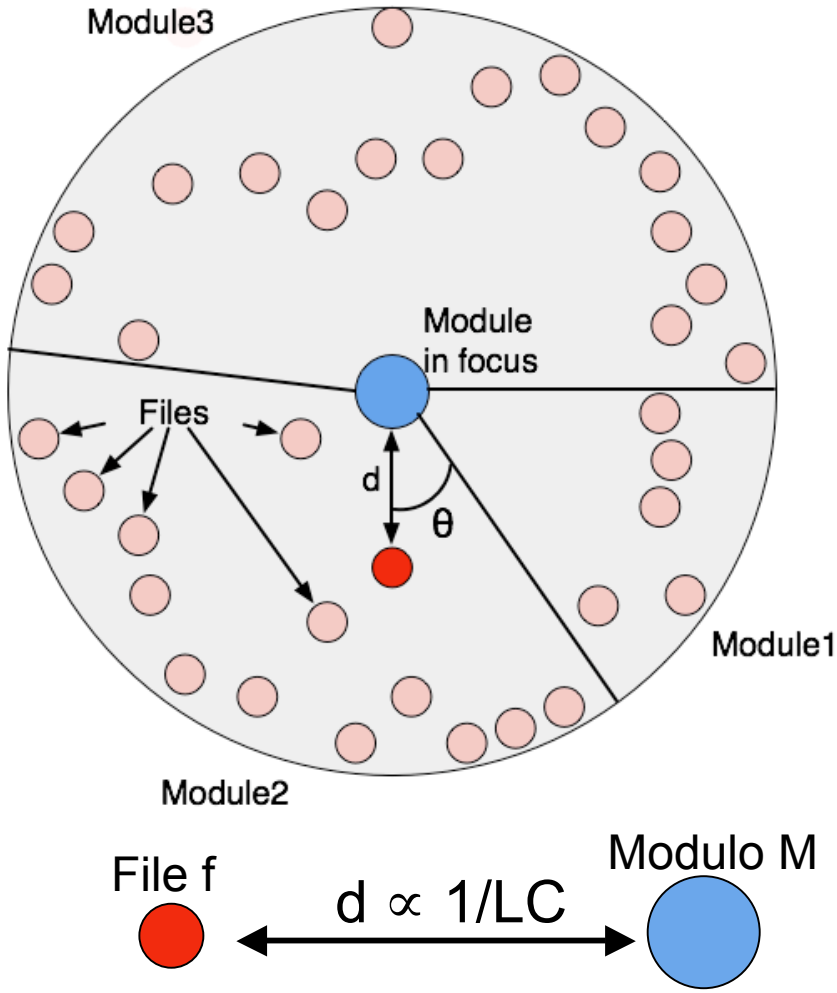


- **Logical coupling (LC):** dipendenza implicita tra due entità che vengono modificate *insieme* [5]
  - Visibile solo nell'evoluzione
  - Ortogonale a dipendenze statiche
- Approcci in letteratura:
  - Dipendenze fra moduli
    - Perdita di informazioni dettagliate (chi genera le dipendenze)
  - Analisi di change impact
    - Manca una vista d'insieme del sistema

[5] Gall *et. al. Detection of Logical Coupling Based on Product Release History.* Proceedings of ICSM 1998, IEEE CS Press, 1998.

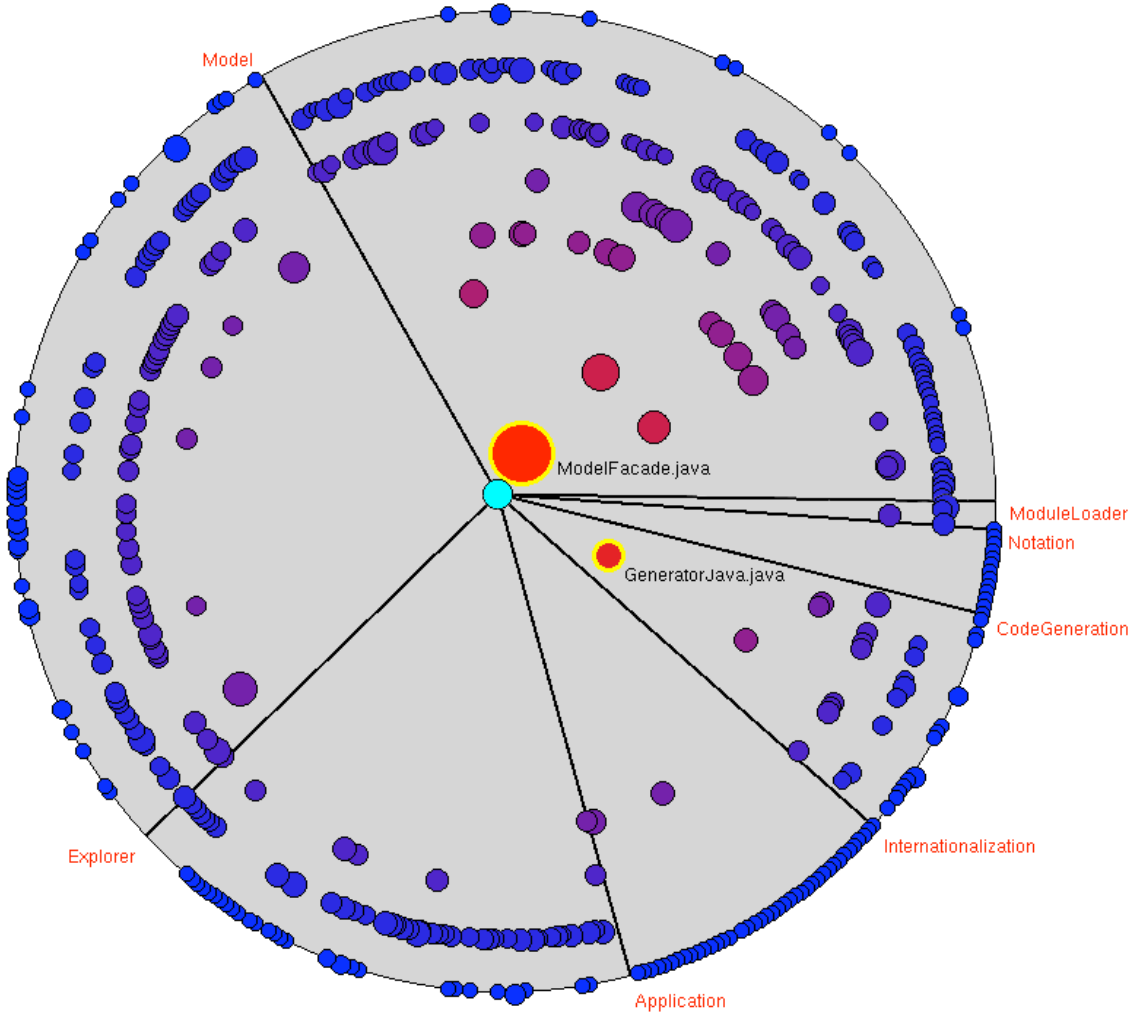
Studiare LC integrati: Evolution Radar [6]

- Modulo in analisi posizionato al centro
- Gli altri moduli rappresentati come settori
- Per ogni modulo, tutti i suoi file sono rappresentati come circonferenze e posizionati con coordinate polari:
  - **d**: inversamente proporzionale a LC
  - θ: distribuzione uniforme con ordine alfabetico
- Si possono mappare metriche su dimensione e colore delle circonferenze

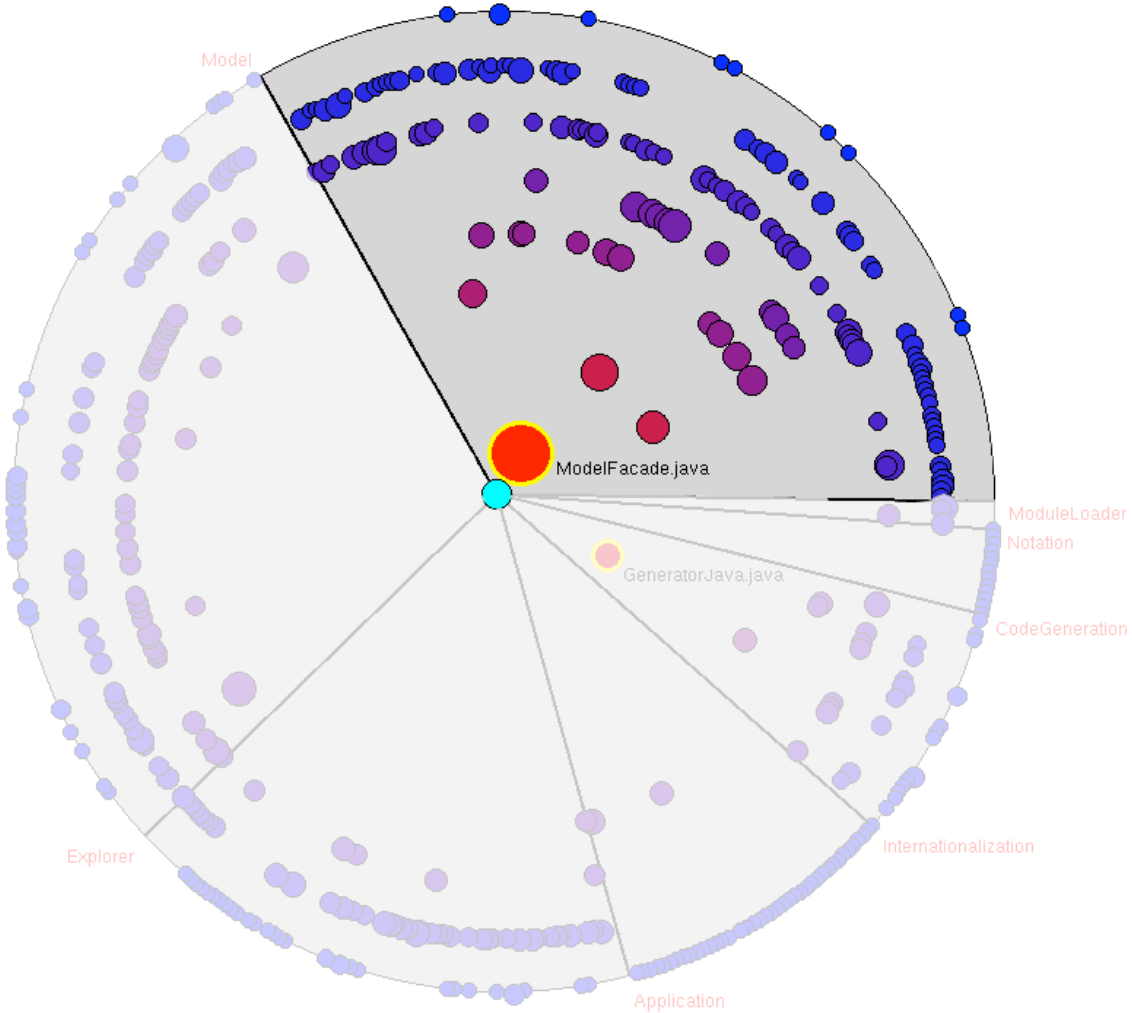


[6] M. D'Ambros, M. Lanza. *Reverse Engineering with Logical Coupling.* Proceedings of WCRE 2006. IEEE CS Press 2006.

Evolution Radar esemplificato

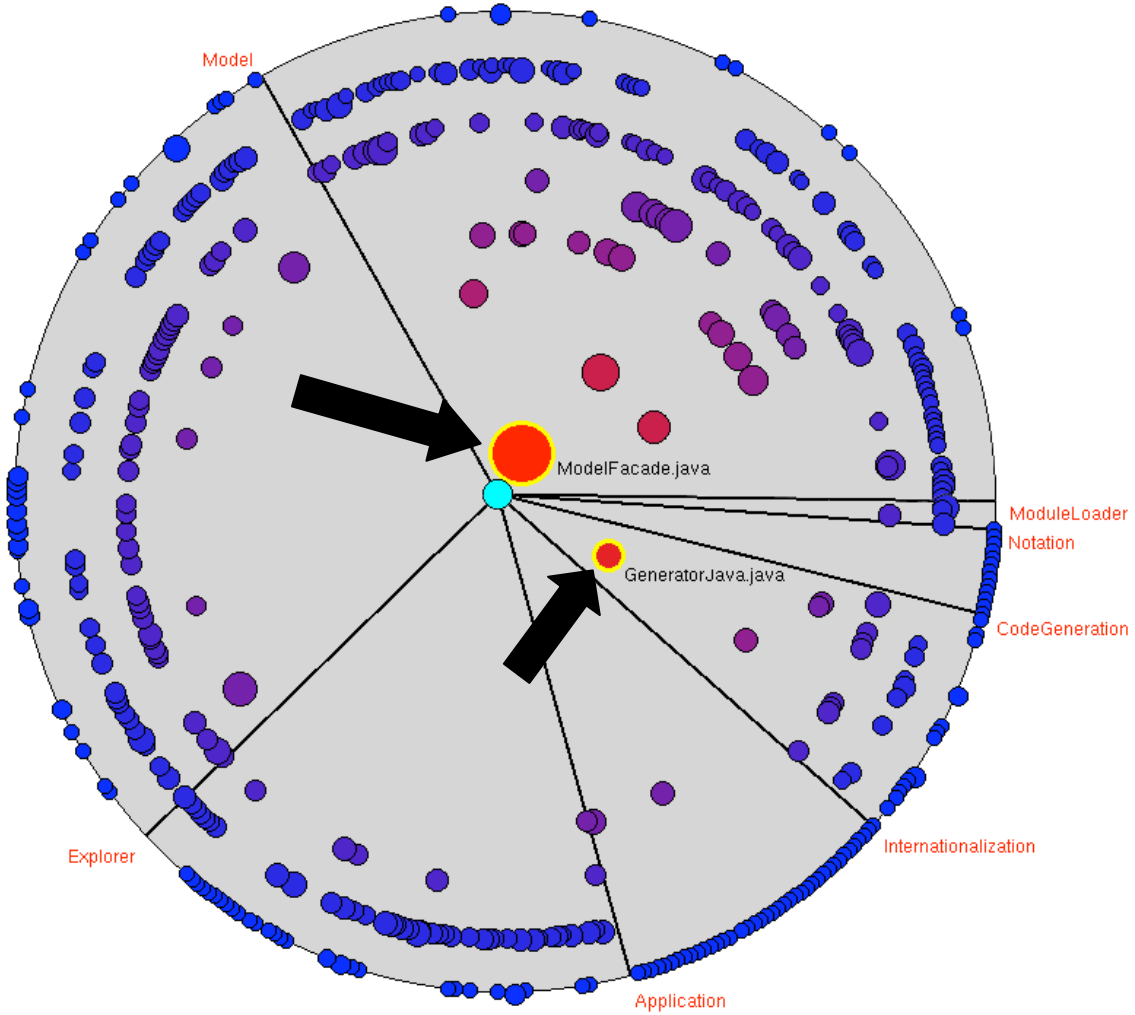


Evolution Radar esemplificato



- LC tra il modulo al centro e tutti gli altri
- Come è strutturato il coupling in termini dei singoli file

Evolution Radar esemplificato



- LC tra il modulo al centro e tutti gli altri
- Come è strutturato il coupling in termini dei singoli file
- Quali sono i file con il coupling più forte

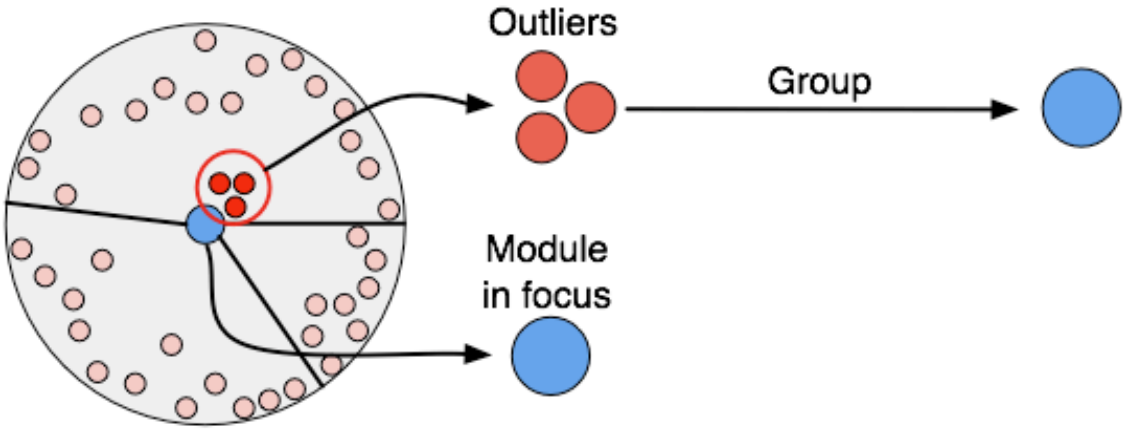
Una visualizzazione statica non è sufficiente per l'analisi…

Interazione con il radar

- Interazione semplice
  - Tutte le entità nella visualizzazione (file e moduli) possono essere analizzate
    - Lettura del codice sorgente
    - Informazioni relative ai commit (data, commenti …)
    - Contenuto dei moduli
- Interazione avanzate
  1. Spawning
  2. Muoversi “nel tempo”
  3. Tracking

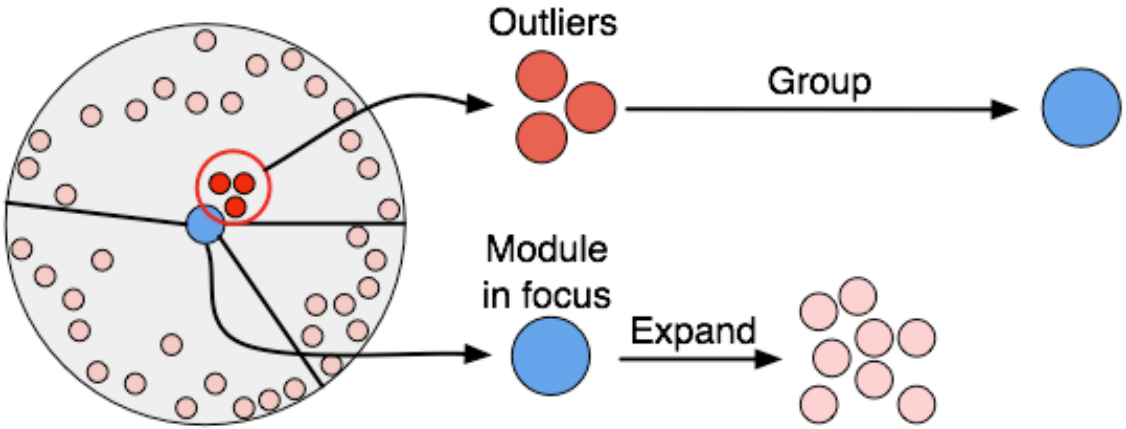
Spawning

Per capire quali file all'interno del modulo analizzato sono responsabili del coupling



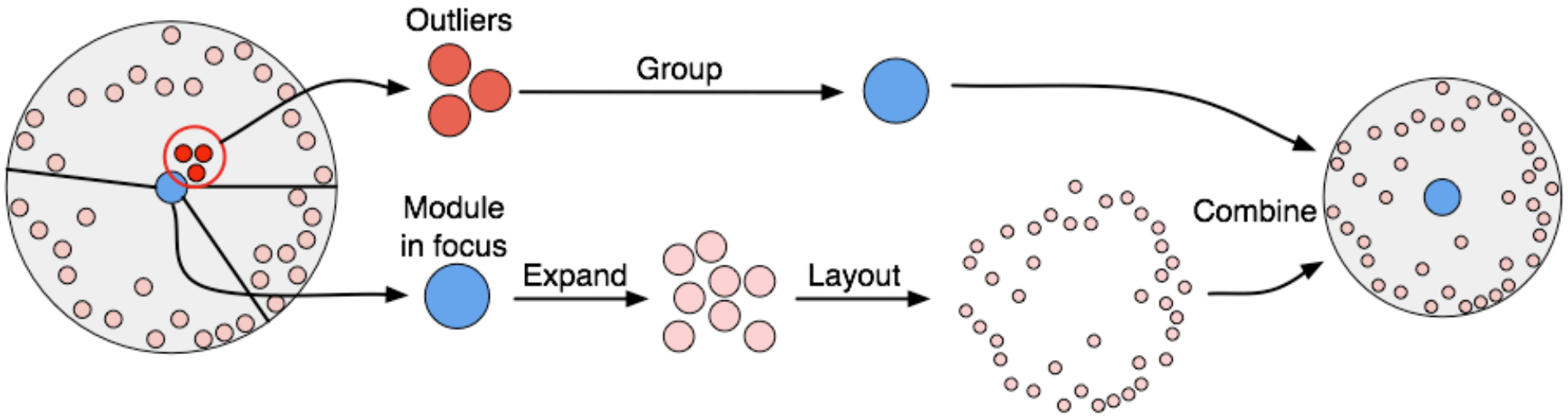
Spawning

Per capire quali file all'interno del modulo analizzato sono responsabili del coupling



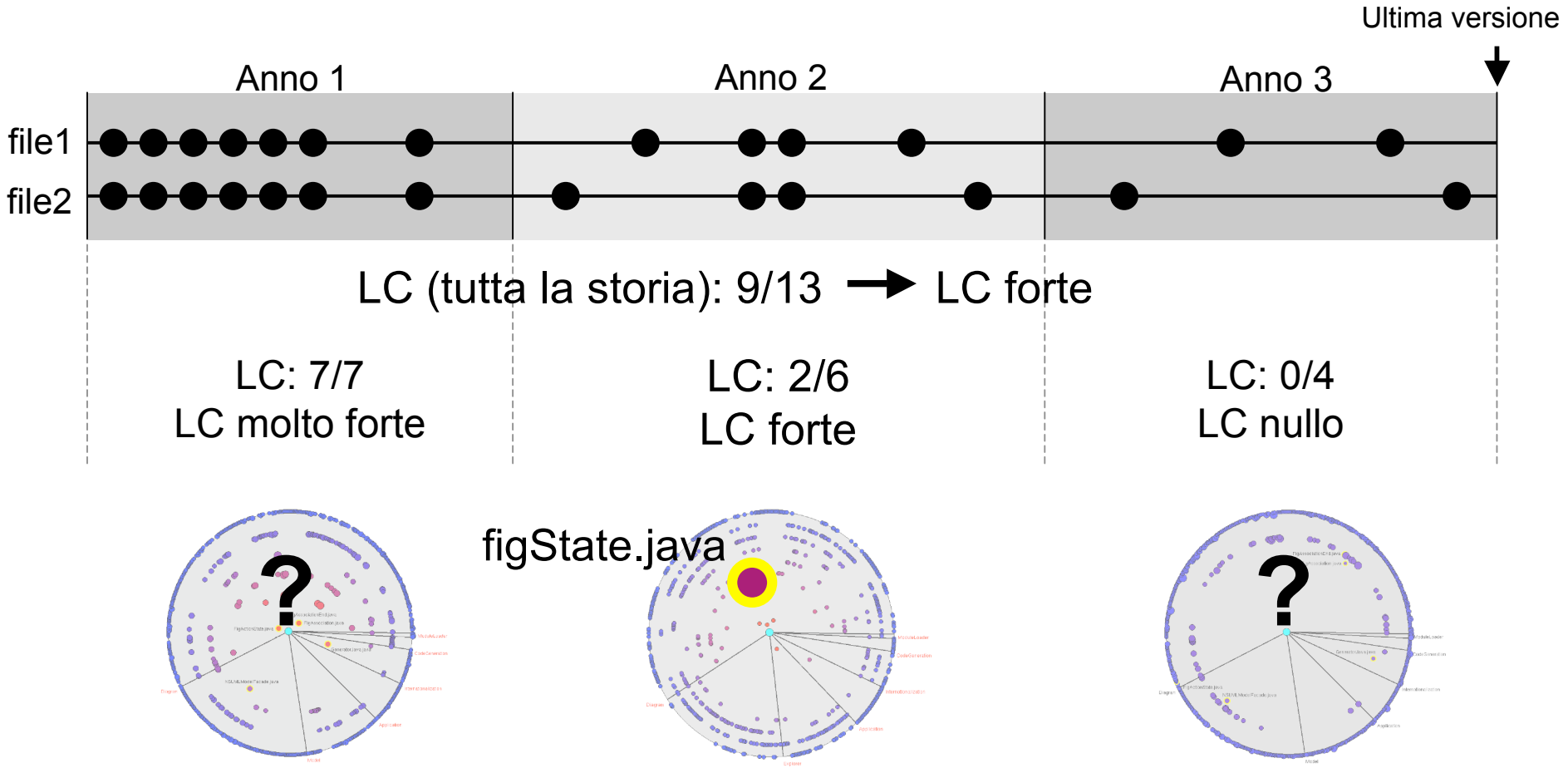
Spawning

Per capire quali file all'interno del modulo analizzato sono responsabili del coupling



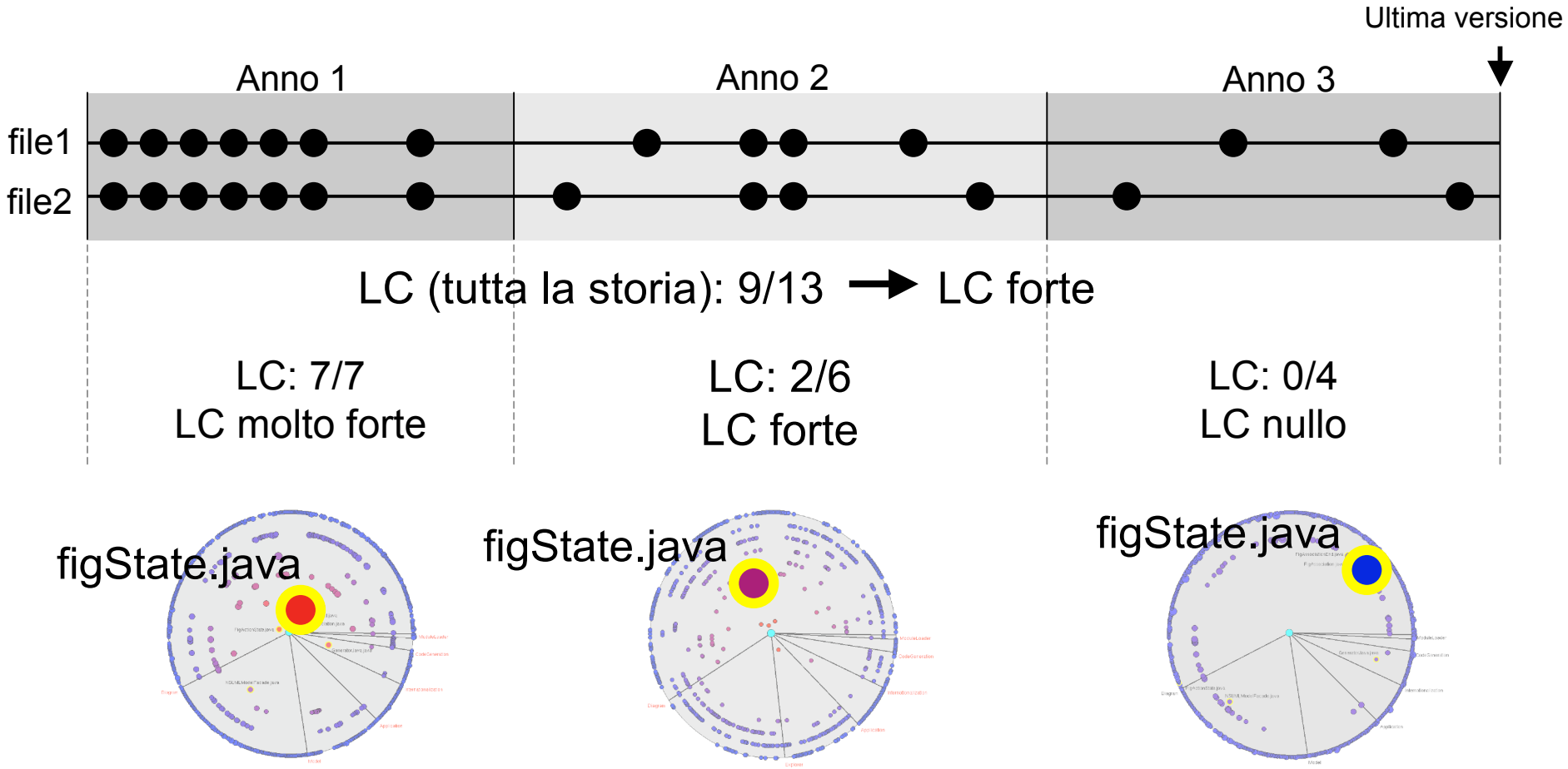
Muoversi nel tempo & Tracking

- **Problema**:
  - Il valore del LC dipende dall'intervallo di tempo considerato

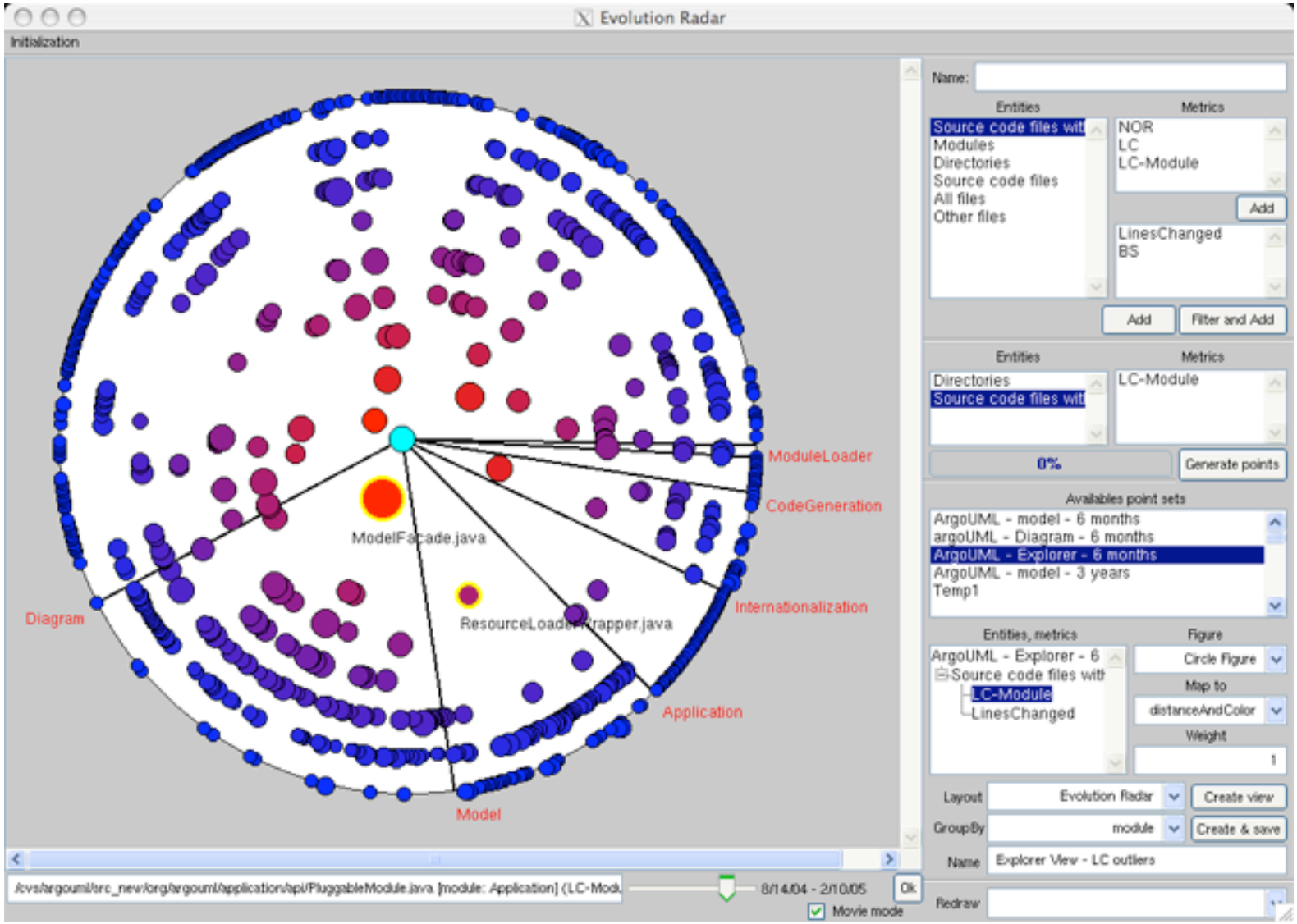


Muoversi nel tempo & Tracking

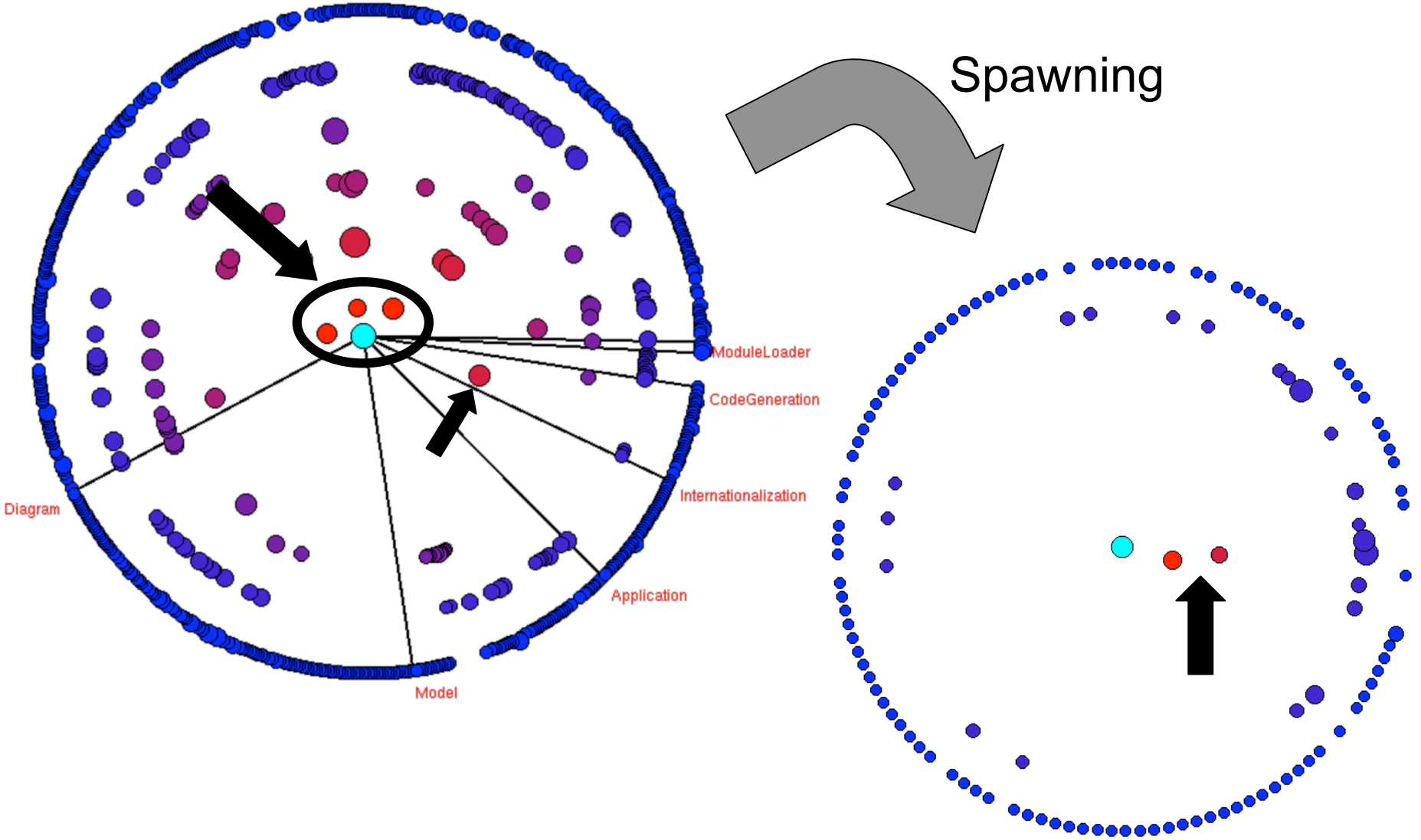
- **Problema**:
  - Il valore del LC dipende dall'intervallo di tempo considerato



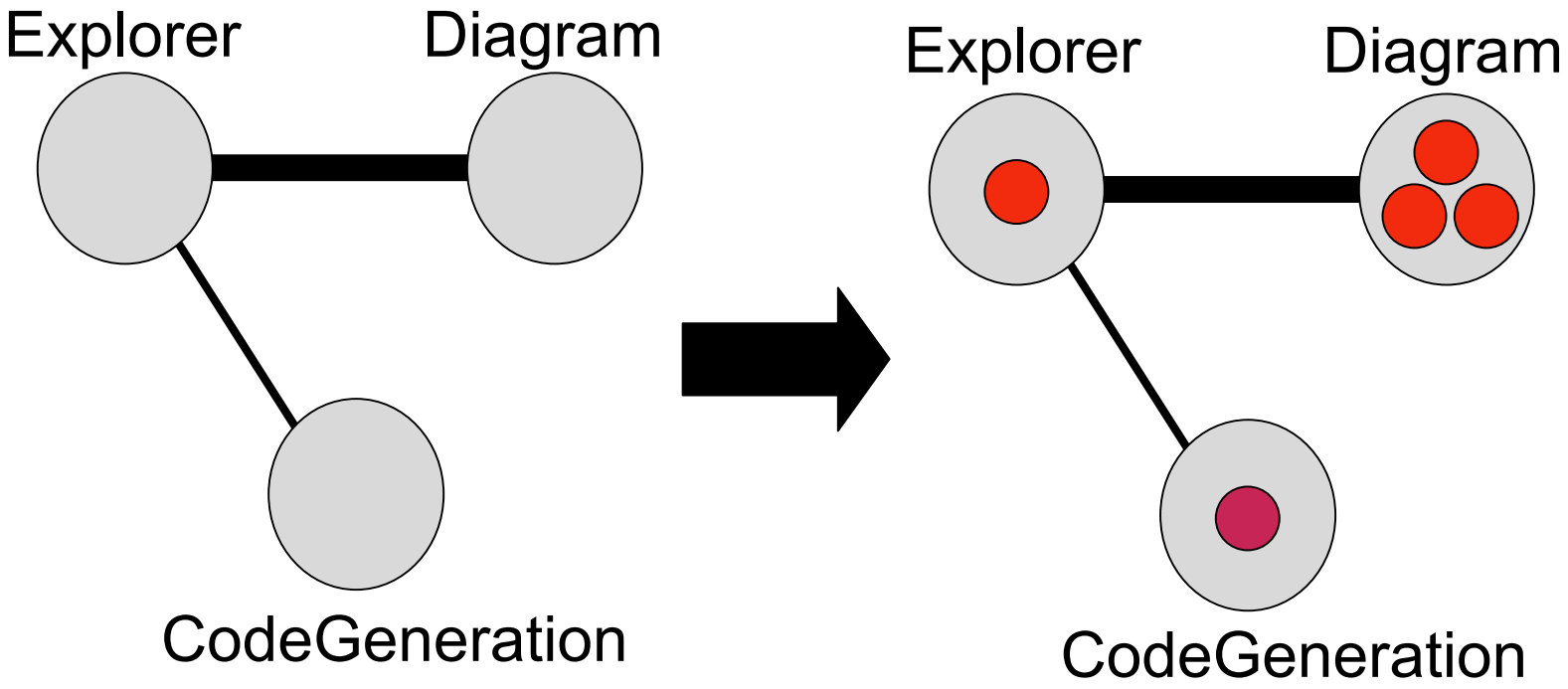
Demo dell'Evolution Radar



Explorer: Agosto-Dicembre 2005

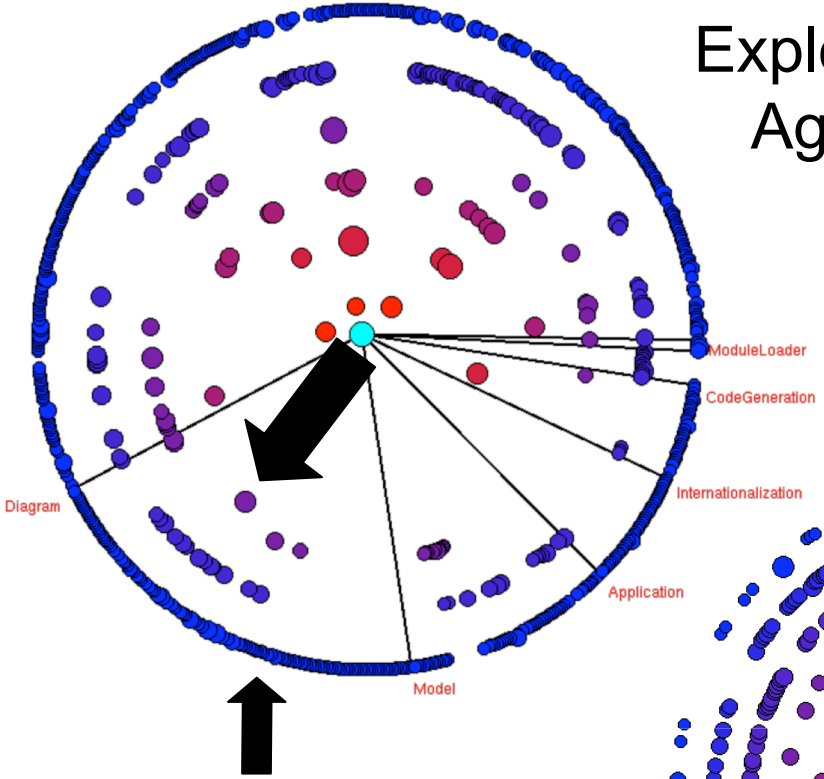


Notevole riduzione di informazioni



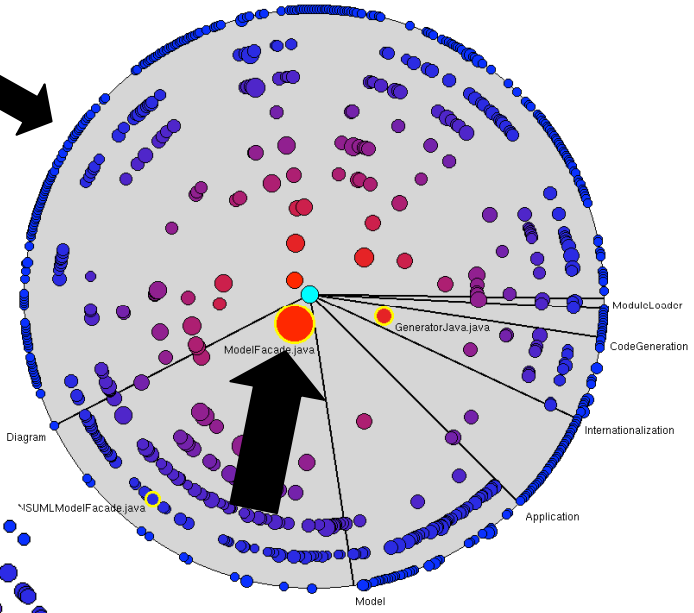
- Le dipendenze fra moduli sono ridotte a dipendenze fra piccoli gruppi di file
- Questi file sono indice di degrado architetturale, vanno spostati o re-ingegnerizzati

L'evoluzione di ModelFacade



Explorer al centro
Ago-Dic 2005

Explorer al centro
Ago-Dic 2004



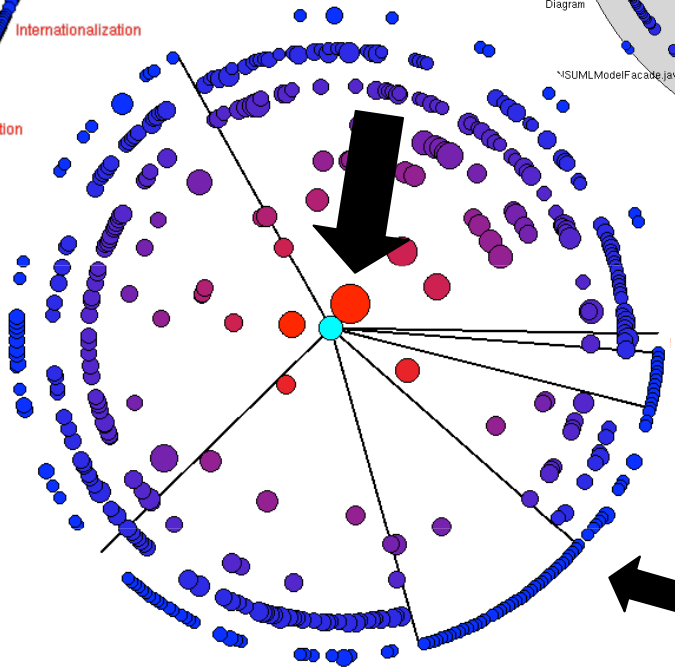
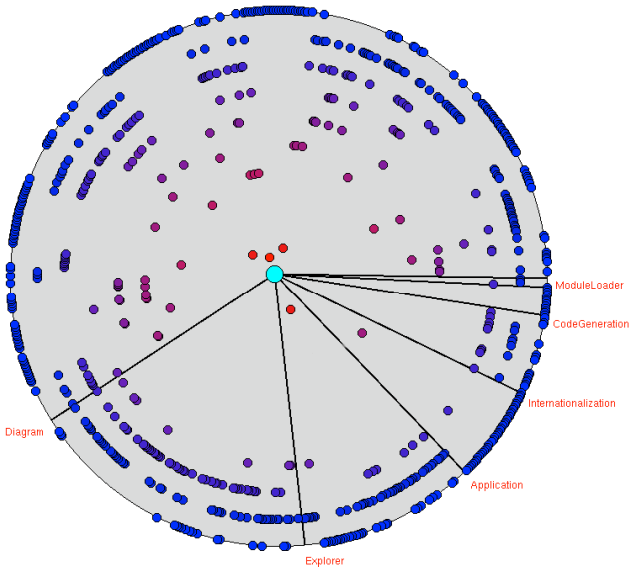


Diagram al centro
Ago-Dic 2004

Evolution Radar: riassumendo…

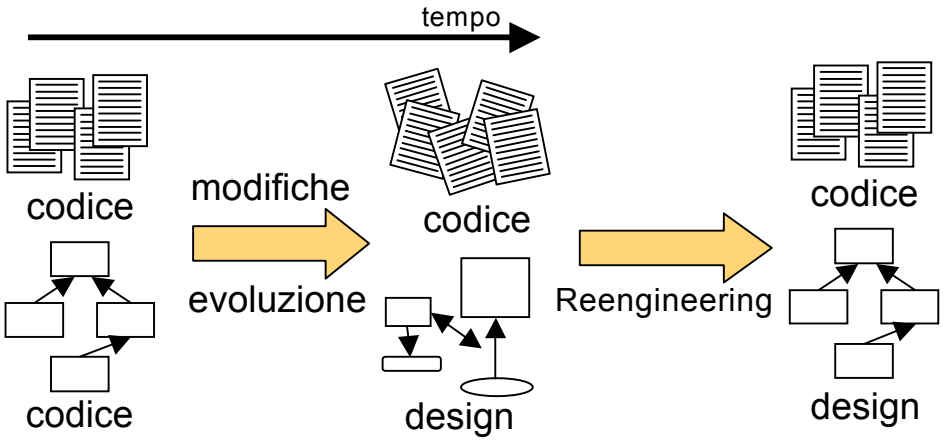
- Le dipendenze logiche (LC)
  - Sono “nascoste”, visibili solo nell’evoluzione
  - Rappresentano un degrado architetturale

- L’evolution radar supporta
  - Lo studio delle dipendenze fra moduli
  - L’analisi della struttura di tali dipendenze in termini dei singoli file



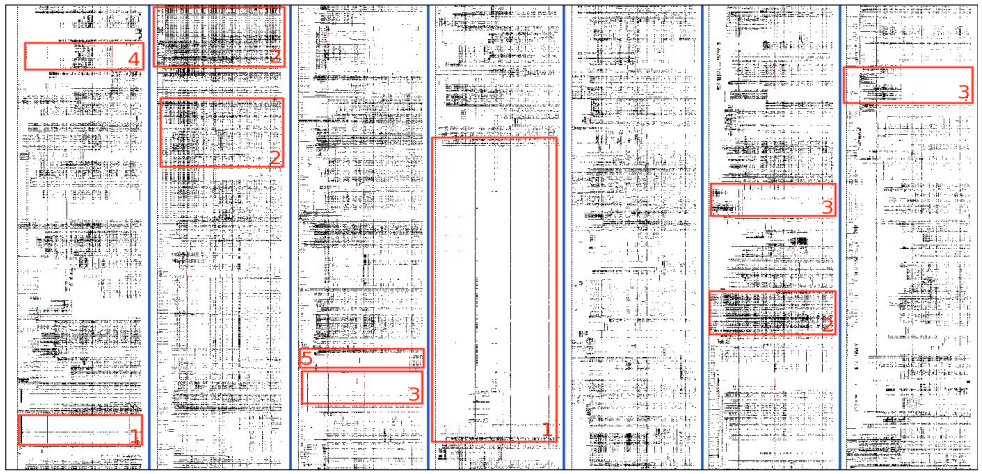
- Introduzione
  - Definizioni, motivazioni, problemi
- Il nostro approccio all'evoluzione del software
  - Recupero dati e visualizzazione
  - BugCrawler e i diversi aspetti dell'evoluzione
    - Demo
  - Evolution radar e dipendenze logiche
    - Demo
- **Conclusioni e direzioni da esplorare**

Conclusioni



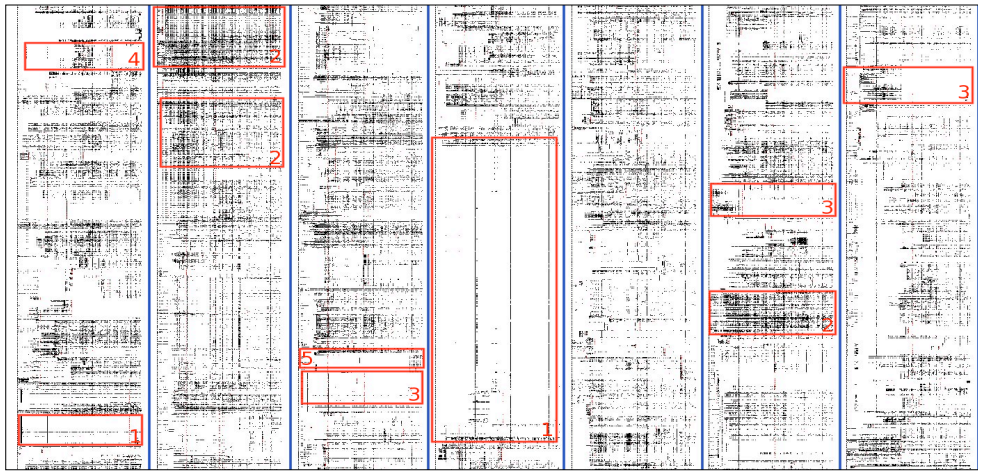
- Il software è soggetto a continui cambiamenti, evolve
  - Peggiora l'architettura e il design
  - Necessario reengineering
- Reengineering preceduto da analisi di struttura ed evoluzione

Conclusioni



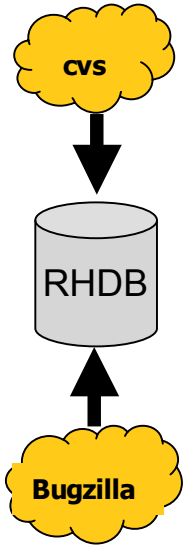
- Il software è soggetto a continui cambiamenti, evolve
  - Peggiora l'architettura e il design
  - Necessario reengineering
- Reengineering preceduto da analisi di struttura ed evoluzione
- Analisi difficile per complessità del sw e della sua evoluzione

Conclusioni

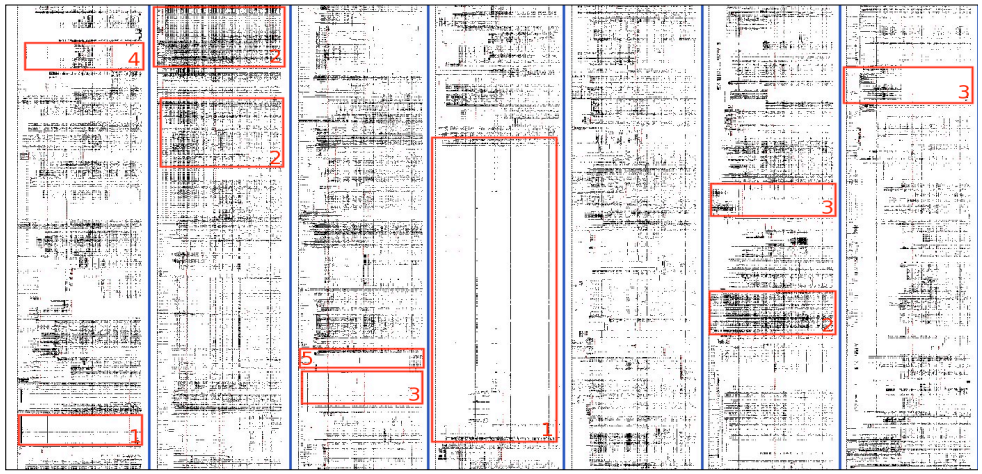


- Informazioni strutturali ed evolutive raccolte e processate

- Il software è soggetto a continui cambiamenti, evolve
  - Peggiora l'architettura e il design
  - Necessario reengineering
- Reengineering preceduto da analisi di struttura ed evoluzione
- Analisi difficile per complessità del sw e della sua evoluzione

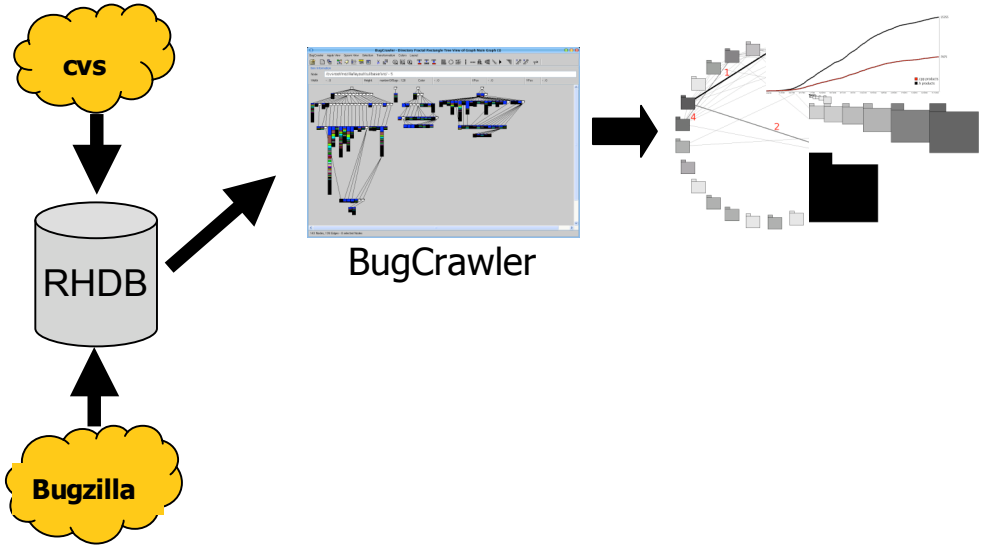


Conclusioni

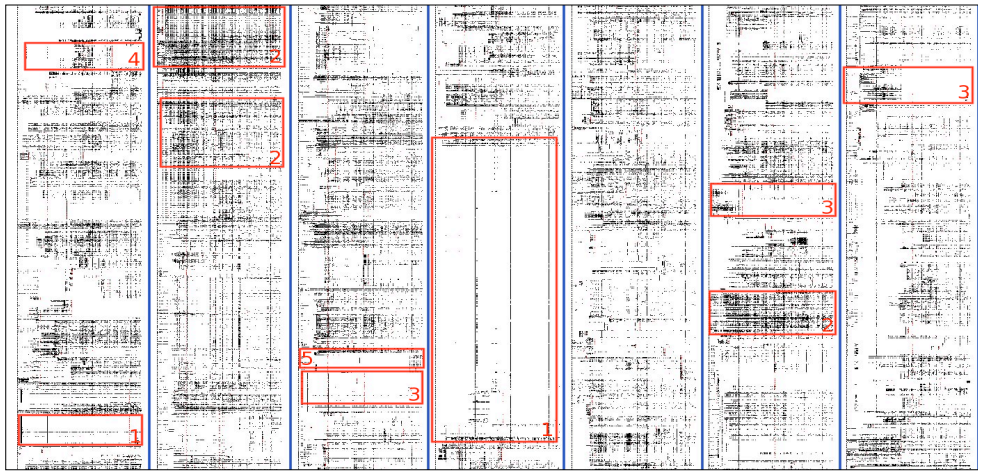


- Il software è soggetto a continui cambiamenti, evolve
  - Peggiora l'architettura e il design
  - Necessario reengineering
- Reengineering preceduto da analisi di struttura ed evoluzione
- Analisi difficile per complessità del sw e della sua evoluzione

- Informazioni strutturali ed evolutive raccolte e processate
- Applicazione di tecniche visuali per interpretare i dati
- BugCrawler
  - Analisi di diversi aspetti evolutivi
  - Identificazione di difetti di design

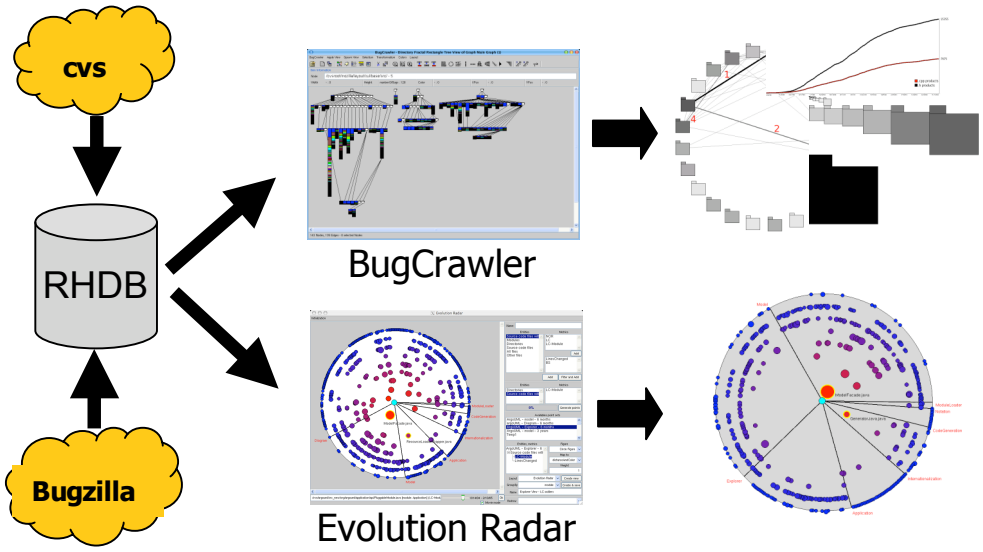


Conclusioni



- Il software è soggetto a continui cambiamenti, evolve
  - Peggiora l'architettura e il design
  - Necessario reengineering
- Reengineering preceduto da analisi di struttura ed evoluzione
- Analisi difficile per complessità del sw e della sua evoluzione

- Informazioni strutturali ed evolutive raccolte e processate
- Applicazione di tecniche visuali per interpretare i dati
- BugCrawler
  - Analisi di diversi aspetti evolutivi
  - Identificazione di difetti di design
- Evolution radar
  - Analisi dipendenze logiche
  - Identificazione difetti architetturali



Direzioni da esplorare

- Nuove tecniche per collegare bug con entità cvs. (e.g. basarsi sui commenti dei bug)
- Migliorare le metriche evolutive (eg. considerare il ciclo di vita dei bug)
- Studiare la relazione fra bug e dipendenze logiche (integrazione di BugCrawler e Evolution Radar)
- Analisi statistica delle dipendenze logiche (e.g. modelli predittivi)
- …

Contatti

Marco D'Ambros

marco.dambros@lu.unisi.ch

http://www.inf.unisi.ch/phd/dambros/

Per approfondire (1/3)

- [Clements03] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, J. Stafford, *Documenting Software Architecture: Views and Beyond*, Addison-Wesley, 2003.
- [Demeyer02] S. Demeyer, S. Ducasse, O. Nierstrasz, *Object-oriented Reengineering Patterns*, Morgan Kaufman, 2002.
- [Dromey95] R.G. Dromey, *A Model for Software Product Quality*, IEEE Transactions on Software Engineering, 21(2):146-162, Feb, 1995.
- [Dunsmore03] A. Dunsmore, M. Roper, M. Wood, The Development and Evaluation of Three Diverse Techniques for Object-Oriented Code Inspection, IEEE Transactions on Software Engineering, 29(8):677-686, 2003.
- [Fowler99] M. Fowler, *Refactoring: Improving the Design of Existing Code*, Addison-Wesley Longman, 1999.
- [Garland&Anthony] J. Garland, R. Anthony, *Large-Scale Software Architecture - A practical guide using UML*, John Wiley, 2003.
- [Hoffman01] D.M. Hoffman, D.M. Weiss (eds.), *Software Fundamentals: Collected Papers by D.L. Parnas*, Addison-Wesley 2001.
- [Kerievsky05] J. Kerievsky, *Refactoring to Patterns*, Pearson Education, 2005
- [Mens99a] K. Mens, R. Wuyts, T. D'Hondt, *Declaratively Codifying Software Architectures using Virtual Software Classifications*, TOOLS-Europe 99, 1999.
- [Mili02] H. Mili, A. Mili, S. Yacoub, E. Addy, *Reuse-Based Software Engineering: Techniques, Organization, and Controls*, John Wiley & Sons, 2002.
- [Murp95a] G. Murphy, D. Notkin, K. Sullivan, *Software Reflexion Models: Bridging the gap between Source and High-Level Models*, Proceedings of SIGSOFT'95, 1995.
- [Riel96] A.J. Riel, *Object-Oriented Design Heuristics*, Addison-Wesley, 1996.
- [Seacord03] R.C. Seacord, D. Plakosh, G.A. Lewis, *Modernizing Legacy Systems*, Addison-Wesley, 2003.
- [Shaw&Garlan] M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1996.
- [swebok] Software Engineering Body of Knowledge, www.swebok.org
- [SPICE] Software Process Improvement and Capability dEtermination, http://www.sqi.gu.edu.au/spice/
- [Vermeulen00] A. Vermeulen, S.W. Ambler, G. Bumgardner, E. Metz, T. Misfeldt, J. Shur, P. Thompson, *The Elements of Java Style*, Cambridge University Press, 2000.

Per approfondire (2/3)

- [BR00] K.H. Bennett, V. T. Rajlich, *The staged model of the software life-cycle: A new perspective on software evolution,* IEEE Computer, vol. 33, no. 7, pp. 66-71, July 2000
- [Capiluppi04] Capiluppi A, Ramil JF, *Multi-level Empirical Studies: an Approach focused on Open Source Software*, 10th International Software Metrics Symposium, IEEE CS Press, 2004
- [Gall+97] H. Gall, M. Jazayeri, R. Klösch, G. Trausmuth, *Software Evolution Observations Based On Product Release History*, Proceedings of the International Conference on Software Maintenance, 1997
- [Lanza03] M. Lanza. Object-Oriented Reverse Engineering - Coarse-grained, Fine-grained, and Evolutionary Software Visualization. PhD thesis, University of Berne, May 2003
- [Lehman80] M.M. Lehman, *Programs, Life Cycles and Laws of Software Evolution*, Proc. IEEE Special Issue on Software Engineering, v. 68, n. 9, Sept. 1980, pp. 1060-1076
- [LR00] M.M. Lehman, J. F. Ramil, *Software Evolution in the Age of Component-Based Software Engineering*, IEE Proc.-Softw., Vol. 147, No.6, December 2000
- [LB85] M.M. Lehman, L.A. Belady, *Program Evolution - Processes of Software Change*, Academic Press, London, 1985, 538 pps.
- [Lehman] M.M. Lehman, *Rules and Tools for Software Evolution Planning and Management*, www-dse.doc.ic.ac.uk/~mml/feast
- [Lehman+97] M.M. Lehman, J.F. Ramil, P.D. Wernick, D.E. Perry, W.M. Turski. *Metrics and Laws of Software Evolution - The Nineties View*, Metrics, IEEE CS Press, 1997.
- [Lehman98] M.M. Lehman, *The Future of Software - Managing Evolution*, IEEE Software, 15(1):40-44, Jan-Feb 1998
- [Parnas94] D.L. Parnas, *Software Aging*, Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy, pp. 279 - 287, ACM Press, 1994.
- [Seacord03] Robert C. Seacord, Daniel Plakosh, Grace A. Lewis, *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*, SEI Series in Software Engineering, Addison-Wesley, 2003
- [swebok] Software Engineering Body of Knowledge, http://www.swebok.org/ [22.08.2005]
- [Turski96] W.M. Turski, *Reference Model for Smooth Growth of Software Systems*, IEEE Transactions on Software Engineering, 22(8):599-600, August 1996
- [Yang03] Hongji Yang, Martin Ward, *Successful Evolution of Software Systems*, Artech House, 2003

Per approfondire (3/3)

- [Zelkowitz79] M.V. Zelkowitz, A.C. Shaw, J.D. Gannon, *Principles of Software Engineering and Design*. Prentice Hall,1979
- [Wu+04] J. Wu, R.C. Holt, A.E. Hassan. *Exploring Software Evolution Using Spectrographs*, Proceedings Working Conference on Reverse Engineering, pp. 80-89, IEEE CS Press 2004
- [Damb06c] M. D'Ambros, M. Lanza. *Reverse Engineering with Logical Coupling*. Proceedings of 13th Working Conference on Reverse Engineering, pp. 189 - 198, IEEE Computer Society, 2006
- [Fisch06b] M. Fischer, H. C. Gall. *EvoGraph: A Lightweight Approach to Evolutionary and Structural Analysis of Large Software Systems*. Proceedings of 13th Working Conference on Reverse Engineering, pp. 179 - 188, IEEE Computer Society, 2006
- [Damb06b] M. D'Ambros, M. Lanza, M. Lungu. *The Evolution Radar: Visualizing Integrated Logical Coupling Information.* Proceedings of 3rd International Workshop on Mining Software Repositories, pp. 26-32, 2006.
- [Damb06a] M. D'Ambros, M. Lanza. *Software Bugs and Evolution: A Visual Approach to Uncover Their Relationships.* Proceedings of 10th European Conference on Software Maintenance and Reengineering, pp. 227-236 , IEEE Computer Society, 2006.
- [Damb05b] M. D'Ambros, M. Lanza, H. C. Gall. *Fractal Figures: Visualizing Development Effort for CVS Entities.* Proceedings of 3rd IEEE International Workshop on Visualizing Software For Understanding and Analysis, pp. 46 - 51, IEEE CS Press, 2005.
- [Damb05a] M. D'Ambros. *Software Archaeology - Reconstructing the Evolution of Software Systems*. Master Thesis, Politecnico di Milano, Italy, 2005.