

# Engenharia de Software II

## Compreensão de Programas

Prof. André Hora  
DCC/UFMG  
2019.1

# Práticas de Manutenção

- Compreensão de programas: atividades para entender o software existente
- Refatoração de código
- Solicitação de mudanças: equipe para estimar priorização, custo (ex: tamanho, complexidade)
- Análise de impacto: áreas impactadas pela mudança em potencial

# Agenda

- **Visão geral**
- Objetivos da compreensão
- Modelo de compreensão
- Fatores que afetam compreensão

# Compreensão de Programas

- Mudanças em software:
  - Manutenção corretiva (defeitos)
  - Manutenção preventiva (refatoração)
  - Manutenção adaptativa (ambiente externo)
  - Manutenção perfectiva (novos requisitos)
- Antes de implementar qualquer mudança, é necessário:
  - Conhecer o software como um todo
  - Conhecer os detalhes relacionados à mudança

# Etapas

- Conhecimento geral sobre **o que** o software faz
- Identificar **onde** as alterações devem ser realizadas
- Conhecimento mais profundo sobre **como** as partes a serem alteradas funcionam

# Custo

- Compreensão de programas consome uma proporção significativa do esforço de manutenção
- Exemplos da indústria:
  - HP: ler código custa **\$200** milhões por ano (dados de 1993)
  - Indústria em geral: **50%** do tempo gasto com alteração é com entendimento de código
- Dados podem variar de acordo com qualidade do código, documentação, nível de experiência do desenvolvedor, tipo da correção, etc

# Compreensão de Programas em Java

- Estudos recentes apontam para dados mais críticos: **56% e 94%** do tempo é gasto com entendimento (dados de 2014)

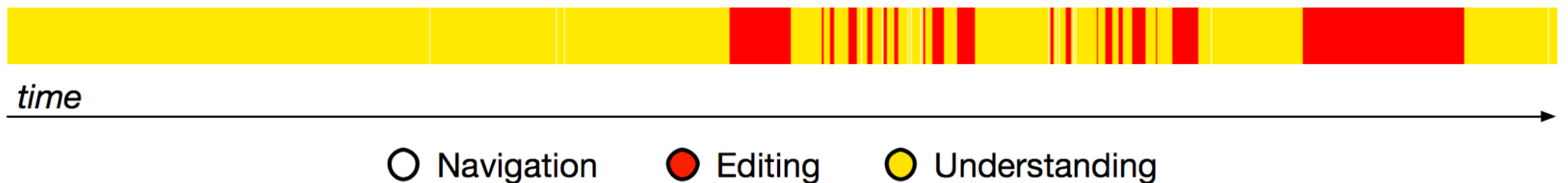


Fig. 2: Visualizing Java Development Activities.

# Agenda

- Visão geral
- **Objetivos da compreensão**
- Modelo de compreensão
- Fatores que afetam compreensão



# Objetivos da Compreensão de Código

- **Objetivo final:** ser capaz de implementar uma solicitação de mudança com sucesso
- Isso envolve lidar com certos aspectos:
  - Domínio do problema (para estimar esforço)
  - Efeito de execução (verificar se mudança teve efeito desejado)
  - Relação causa-efeito (análise de impacto)
  - Ferramentas de suporte (métricas de complexidade, manutenibilidade, engenharia reversa)

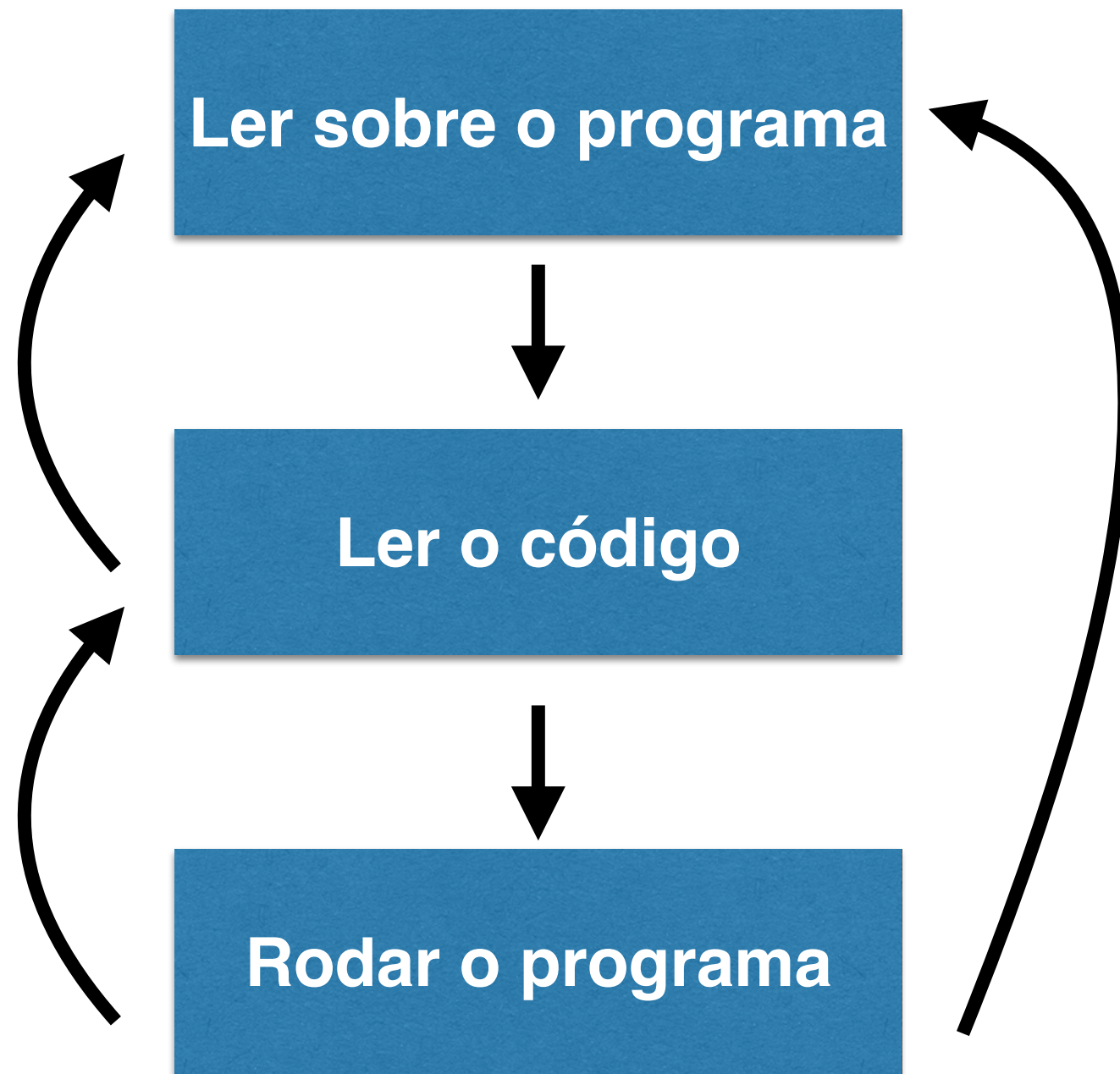
# Agenda

- Visão geral
- Objetivos da compreensão
- **Modelo de compreensão**
- Fatores que afetam compreensão

# Modelo de Compreensão

- Desenvolvedores podem variar suas formas de pensar, revolver problemas e selecionar técnicas
- De modo geral, três ações envolvem o entendimento de programas:
  1. **Ler sobre o programa:** acessar documentação (especificação, projeto, etc), se disponível
  2. **Ler o código:** obter visão global e local do programa; usar ferramentas de engenharia reversa
  3. **Rodar o programa:** revelar características do sistema difíceis de obter apenas lendo código

# Modelo de Compreensão



# Exercício

- Escreva 5 fatores que podem afetar a compreensão de programas
- Exemplo: formatação do código

```
<?php
class Calculator {
public function add($a, $b) { return $a + $b; }
public function multiply($a, $b) { return $a * $b;
} public function divide($a, $b) { if($b == null) {
throw new Exception("Division by zero"); } return $a / $b; }
public function subtract($a, $b) { return $a - $b; } }
?>
```



```
<?php
class Calculator {
    public function add($a, $b) {
        return $a + $b;
    }
    public function multiply($a, $b) {
        return $a * $b;
    }
    public function divide($a, $b) {
        if ($b == null) {
            throw new Exception ( "Division by zero" );
        }
        return $a / $b;
    }
    public function subtract($a, $b) {
        return $a - $b;
    }
}
?>
```

# Agenda

- Visão geral
- Objetivos da compreensão
- Modelo de compreensão
- **Fatores que afetam compreensão**

# Fatores que Afetam Compreensão

- Conhecimento sobre o domínio
- Suporte de ferramentas engenharia reversa
- Qualidade do código (legibilidade, complexidade, uso de padrões de projeto, etc)
- Qualidade da documentação (externa e interna)
- Nível de experiência do desenvolvedor

# Fatores que Afetam Compreensão

- **Conhecimento sobre o domínio**
- Suporte de ferramentas engenharia reversa
- Qualidade do código (legibilidade, complexidade, uso de padrões de projeto, etc)
- Qualidade da documentação (externa e interna)
- Nível de experiência do desenvolvedor

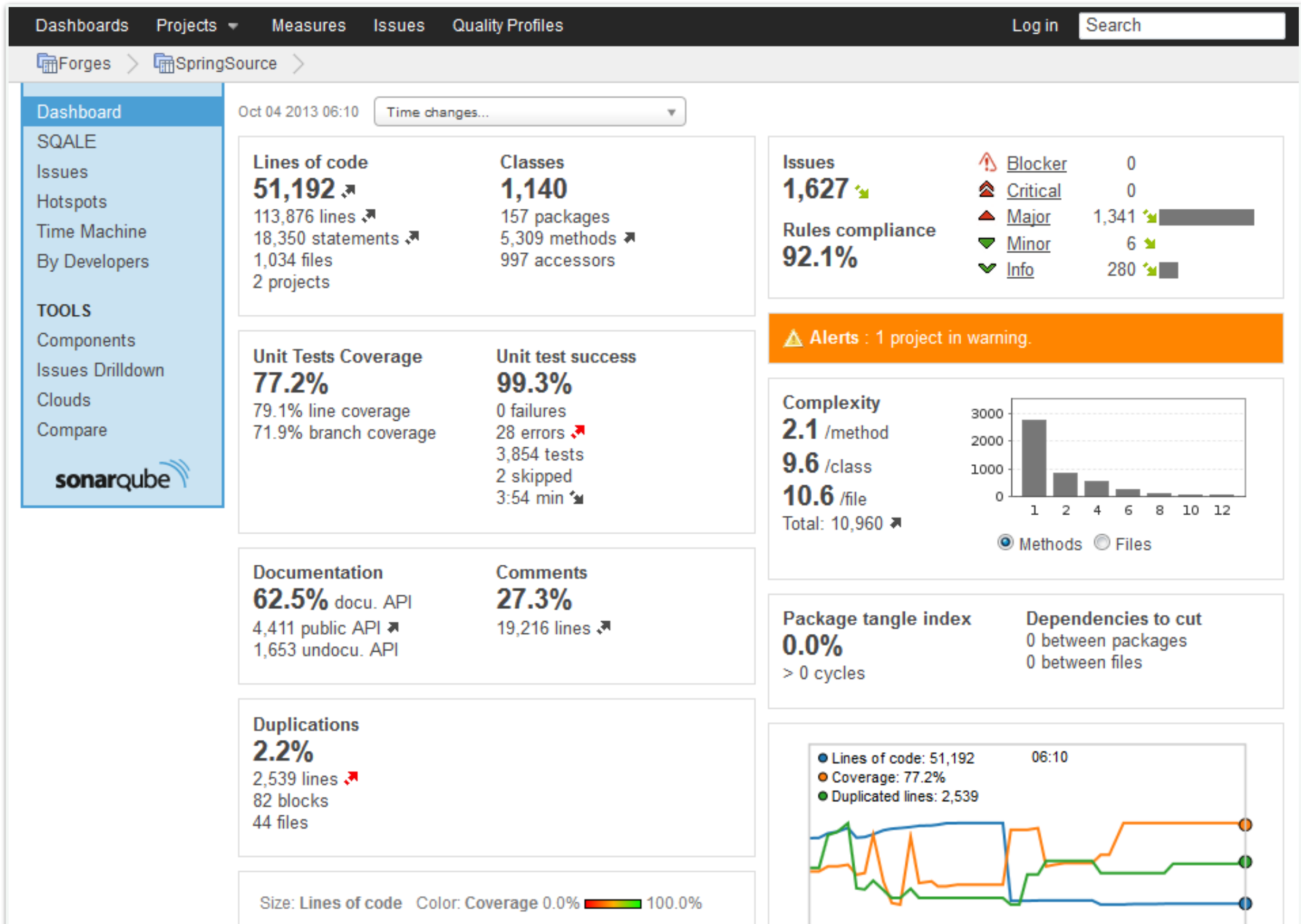


# Conhecimento Sobre o Domínio

- Ao longo do tempo, desenvolvedores tornam-se especialistas em um determinado domínio (linguagem de programação, biblioteca, BD, etc)
- Logo, o nível de expertise tem um papel significativa na compreensão de programas
- Estudos mostram que o nível de expertise influencia positivamente no entendimento

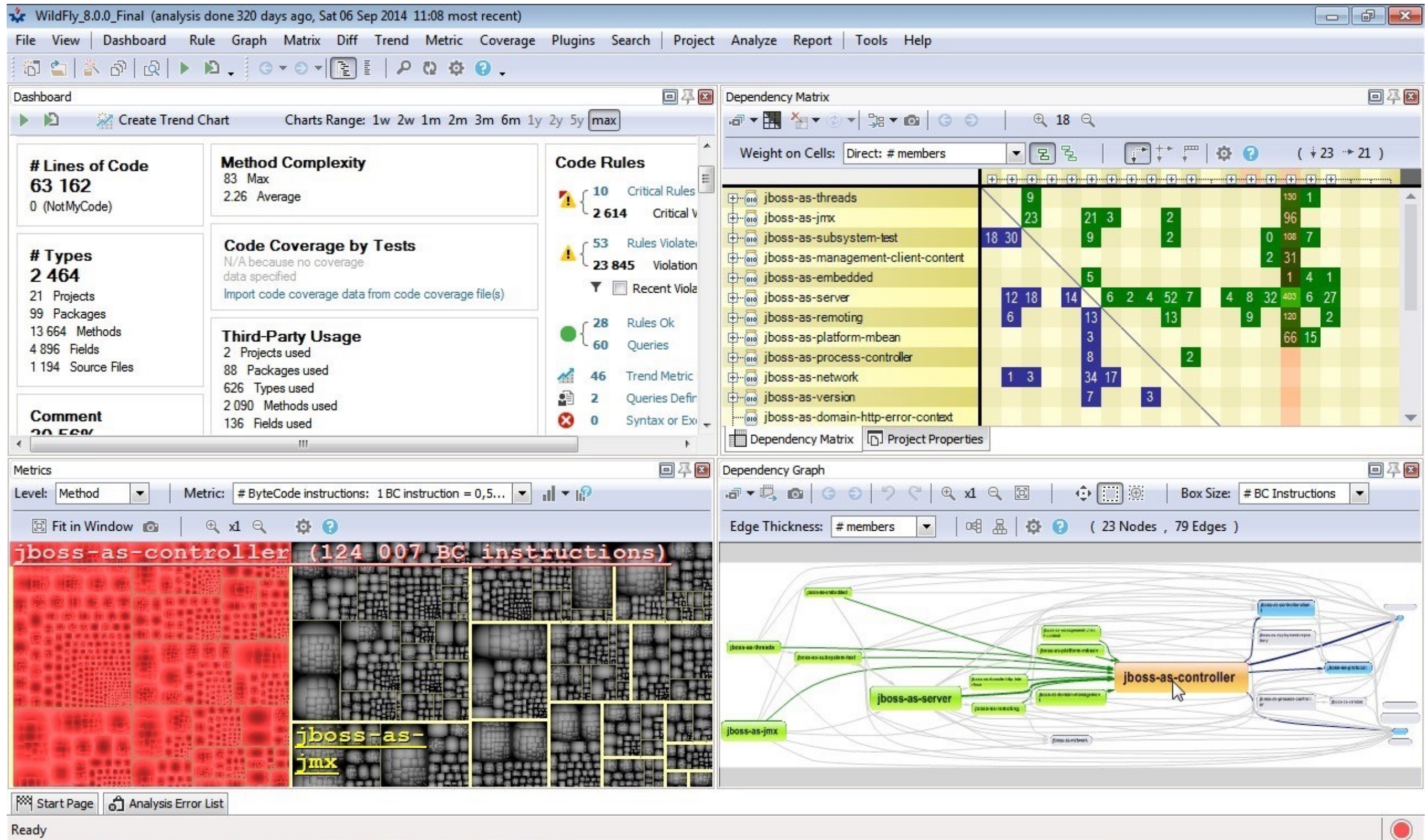
# Fatores que Afetam Compreensão

- Conhecimento sobre o domínio
- **Suporte de ferramentas engenharia reversa**
- Qualidade do código (legibilidade, complexidade, uso de padrões de projeto, etc)
- Qualidade da documentação (externa e interna)
- Nível de experiência do desenvolvedor





# JArchitect





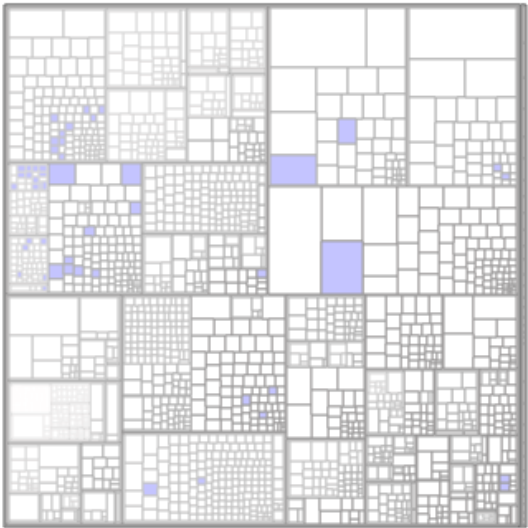
**Moose Finder**

All model classes (2387) (FAMIXClassGroup)      Group (50) (FAMIXClassGroup)

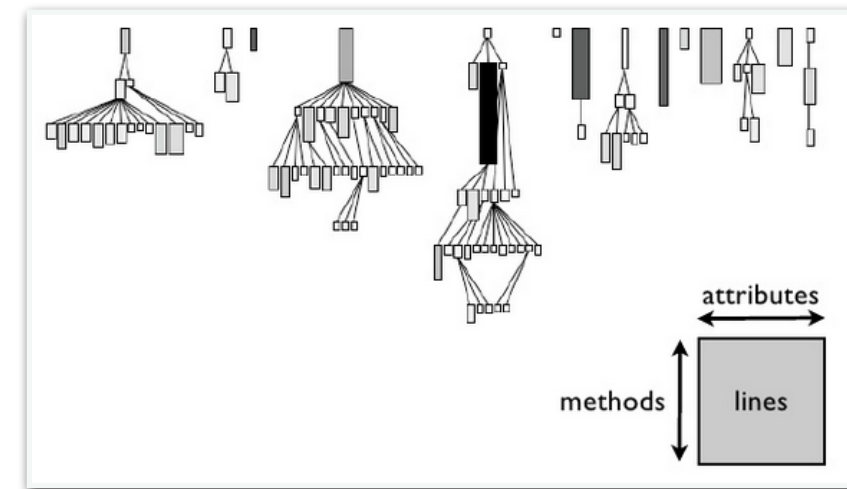
- org::argouml::persistence::OldModelMemberFilePersister
- org::argouml::model::mdr::UndoCoreHelperDecorator.setPo
- org::argouml::kernel::ProjectImpl
- org::argouml::core::propertypanels::ui::UMLTimeExpressionM
- org::argouml::uml::ui::foundation::core::UMLClassActiveChe
- org::argouml::uml::ui::ActionCut
- org::argouml::ui::ProjectBrowser.loadProject2(File,boolean,f
- org::argouml::persistence::XmiFormatException
- org::argouml::language::csharp::importer::csparser::nodes::e
- org::argouml::kernel::DefaultUndoManager::InteractionStack
- org::argouml::core::propertypanels::ui::UMLIncludeListMode
- org::argouml::uml::ui::behavior::state\_machines::UMLSynch
- org::argouml::core::propertypanels::ui::UMLTransitionTrigge
- org::argouml::uml::ui::PropPanelFactoryManager
- org::argouml::uml::ui::ActionBooleanTaggedValue
- org::argouml::cognitive::ui::WizStepTextField
- org::argouml::uml::ui::UMLStimulusActionTextField
- org::argouml::uml::ui::foundation::core::ActionAddEnumerat
- org::argouml::language::cpp::generator::GeneratorCpp
- org::argouml::state2::diagram::FigVertex::FigRegionCompart
- org::argouml::core::propertypanels::ui::UMLModelElementNa
- org::argouml::ui::cmd::ShortcutField.ShortcutField(String,int
- org::argouml::core::propertypanels::ui::RowSelector::MoveB
- org::argouml::core::propertypanels::model::GetterSetterMan
- org::argouml::uml::reveng::ImporterManager
- org::argouml::uml::diagram::activity::ui::InitActivityDiagram

200 / 2387

each isAnnotated



# Moose



**Moose Meta Browser**

Entities      24 Relations      48 Properties

**Entities**

- FAMIX.NamedEntity (FAMIX)
  - FAMIX.ContainerEntity (FAMIX)
    - FAMIX.BehaviouralEntity (FAMIX)
    - FAMIX.ScopingEntity (FAMIX)
    - FAMIX.Type (FAMIX)
      - FAMIX.AnnotationType (FAMIX)
      - FAMIX.Class (FAMIX)
      - FAMIX.Enum (FAMIX)
      - FAMIX.ParameterType (FAMIX)
      - FAMIX.ParameterizedType (FAMIX)
      - FAMIX.PrimitiveType (FAMIX)
      - FAMIX.TypeAlias (FAMIX)
    - FAMIX.LeafEntity (FAMIX)
  - FILE.AbstractFile (FILE)

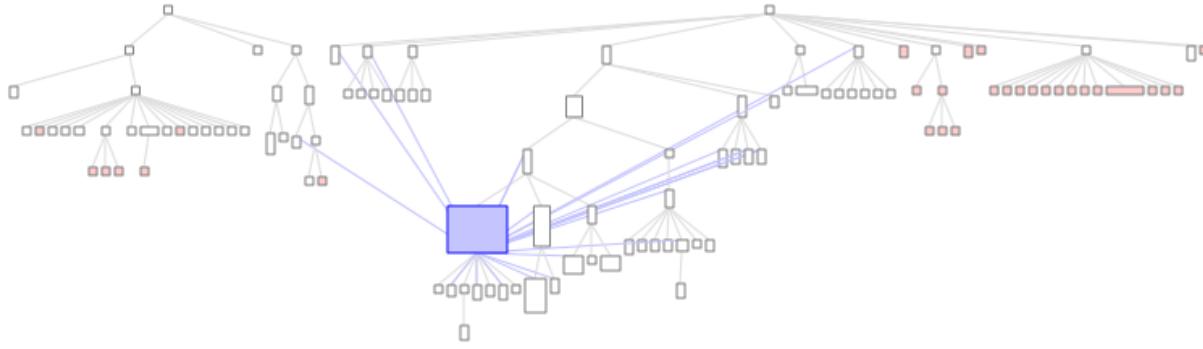
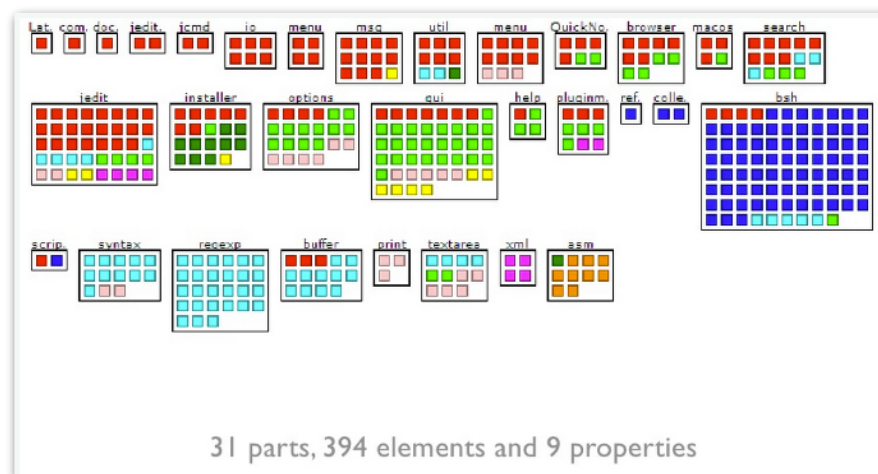
**24 Relations**

- annotationInstances (NamedEntity)
- argumentsInParameterizedTypes (ParameterizedType \*)
- attributes (Attribute \*)
- behavioursWithDeclaredType (BehaviouralEntity \*)
- belongsTo (NamedEntity)
- clientTypes (Type \*)
- comments (SourcedEntity)
- container (ContainerEntity)
- declaredSourceLanguage (SourcedEntity)
- definedAnnotationTypes (ContainerEntity)
- functions (ContainerEntity)
- incomingReferences (ContainerEntity)
- instances (Instance \*)
- methods (Method \*)
- outgoingReferences (ContainerEntity)

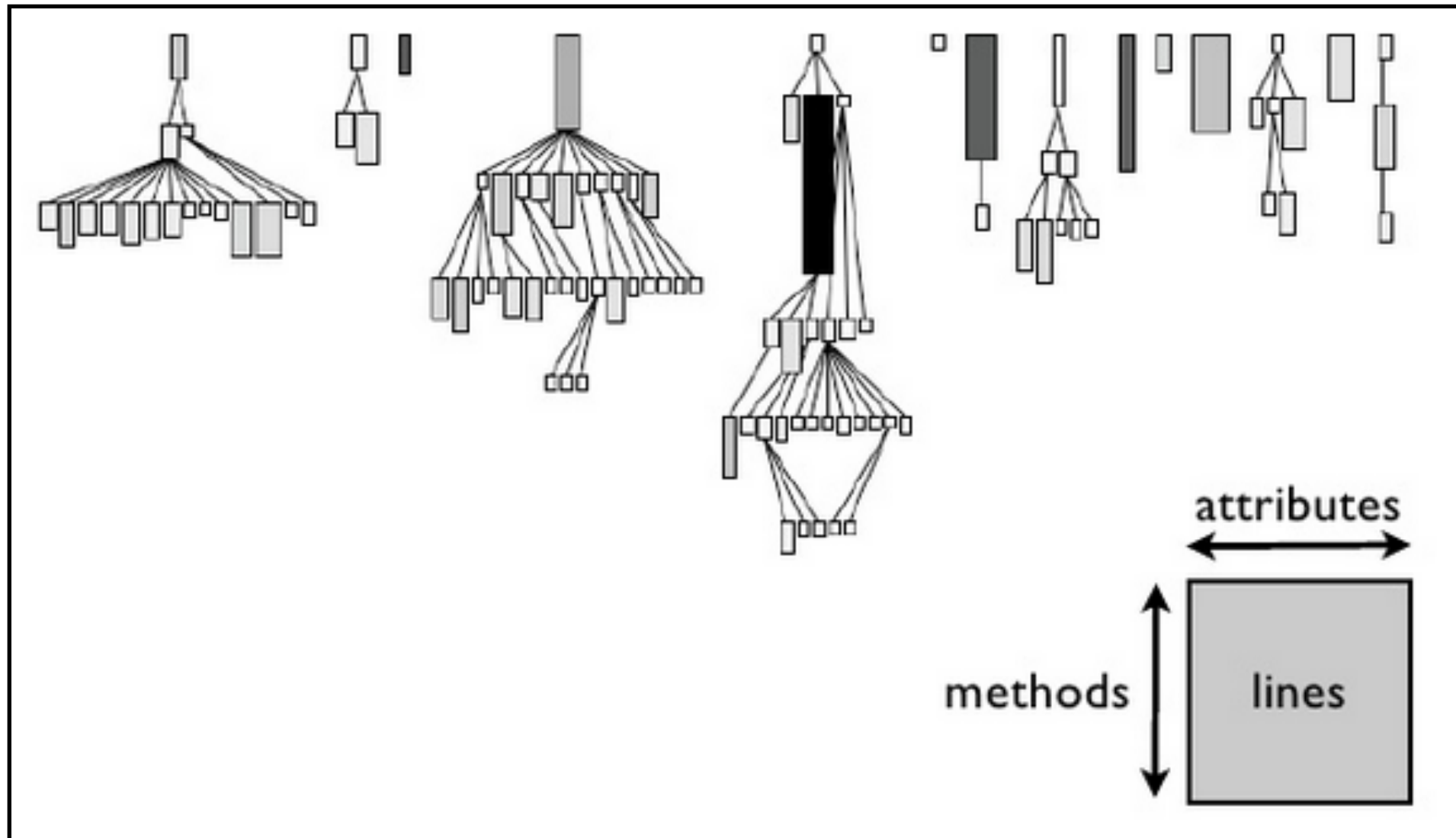
**48 Properties**

name	type
annotationInstances	NamedEntity
argumentsInParameterizedTypes	ParameterizedType *
attributes	Attribute *
behavioursWithDeclaredType	BehaviouralEntity *
belongsTo	NamedEntity
clientTypes	Type *
comments	SourcedEntity
container	ContainerEntity
declaredSourceLanguage	SourcedEntity
definedAnnotationTypes	ContainerEntity
functions	ContainerEntity
incomingReferences	ContainerEntity
instances	Instance *
methods	Method *
outgoingReferences	ContainerEntity

**Map**

# System Complexity



# Fatores que Afetam Compreensão

- Conhecimento sobre o domínio
- Suporte de ferramentas engenharia reversa
- **Qualidade do código (legibilidade, complexidade, uso de padrões de projeto, etc)**
- Qualidade da documentação (externa e interna)
- Nível de experiência do desenvolvedor

# Qualidade de Código

- Complexidade
- Legibilidade
- Comentários em código
- Formatação



# Qualidade de Código

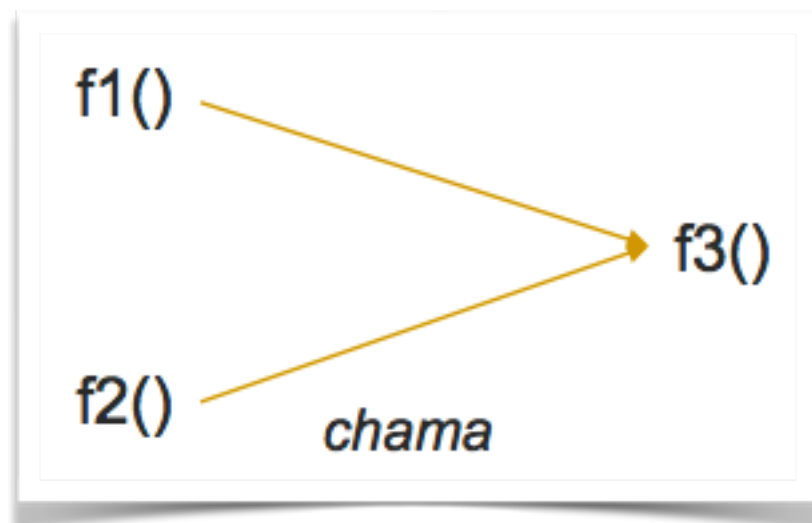
- **Complexidade**
- Legibilidade
- Comentários em código
- Formatação

## Exemplos

- Fan-in
- Tamanho do Código
- Complexidade Ciclomática
- Profundidade de Aninhamento
- CK e várias outras

# Fan-in

- Número de funções que chamam uma dada função
- Valor alto significa grande impacto em mudanças (propagação)



$$\text{fan-in}(f3) = 2$$

# Tamanho do Código (LOC)

- Mede o tamanho do sistema
- Relacionado com a complexidade de manutenção, podendo indicar possíveis pontos de refatoração
- Em geral, quanto maior, mais complexo e propenso a erros será o componente
- LOC: número de linhas de código
  - Com ou sem documentação?

```
import lejos.nxt.*;

public class Hello
{
    /**
     * The main method is where your program starts
     */
    public static void main(String[] args) throws Exception
    {
        // makes a buzzing sound
        Sound.buzz();
        // shows text on column 3, row 4 of the LCD
        LCD.drawString("I am alive !!", 3, 4);
        // pauses 2000 ms (= 2sec)
        Thread.sleep(2000);
        // makes another buzzing sound
        Sound.buzz();
        // end of program
    }
}
```

# Complexidade Ciclomática

- Indica a complexidade do código através da quantidade de caminhos de execução
- Mede a complexidade de controle do programa
  - if, while, for, etc
- Relacionada à facilidade de compreensão

# Profundidade de Aninhamento

- Número de estruturas internas, como for, while e if aninhadas
- Dificulta compreensão

```
1. public int method(int n) {  
2.     for(int i = 0; i < n; i++) {  
3.         for(int j = 0; j < n; j++) {  
4.             for(int k = 0; k < n; k++) {  
5.                 ...  
6.             }  
7.         }  
8.     }  
9. }
```

# Métricas OO (CK)

- Métricas de Chidamber-Kemerer (CK):  
específicas para sistemas orientado a objetos
  - Profundidade da Herança (DIT)
  - Número de Filhos (NOC)
  - Acoplamento entre Objetos (CBO)
  - Falta de Coesão em Métodos (LCOM)
  - Métodos Ponderados por Classes (WMC)
  - Resposta para Classe (RFC)