# The Future of Programming Environments: Integration, Synergy and Assistance

Andreas Zeller
FOSE / ICSE 2007

## Writing a program in the past...

Editor (write)

Compiler (compile)

Runtime Environment (execute)

## Writing a program today...

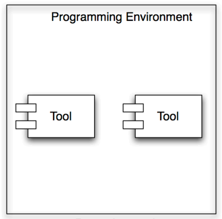
Cross-referencers, code navigation (understand)

Code checkers (control quality)

Refactoring browsers (improve quality)

CSCW - Comp. Supported Collaborative Work (collaborate)

## 1 Integration



“*The quality of a programming environment is not only the quality of its programming tools, but also the integration of these tools.*”

## Different Viewpoints

Behavioral (how does it work)

Semantical (what does it mean)

Syntactical (how is it programmed)

Architectural (how is it designed)

## Different Artifacts: Program + Process

Code

Design Documents (e.g. UML diagrams)

Change Histories (e.g. CVS/SVN repositories)

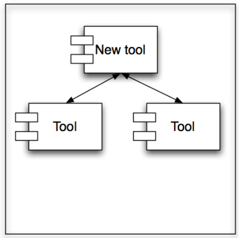
Test Logs

Bug databases

Programmer activity

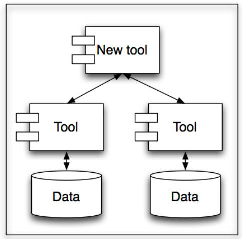
...

## 2 Synergy

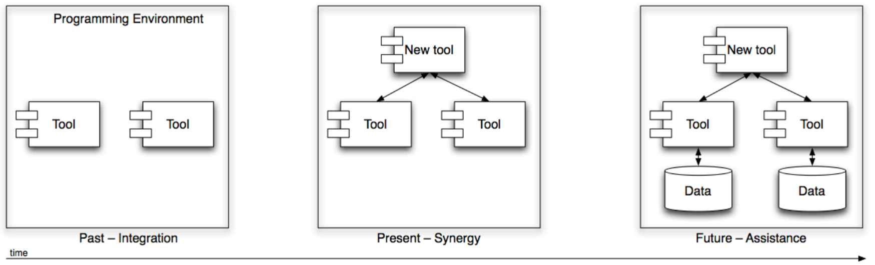


“ *Managing programs integrates with managing processes and creates synergies.* ”

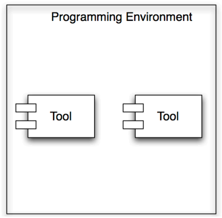
## 3 Assistance



“ *As our environments evolve to collect more and more data [...] one can expect rules and recommendations to emerge from this data, effectively assisting the programmer in daily tasks and decisions like an expert could do.* ”



## 1 Integration



“*To make the environment more than a mere aggregation of tools, it is necessary that the tools not only present their results to the user, but also provide support for automation.*”

## Uniform Interfaces

(User-friendly ways to invoke a tool)

UNIX tool (user friendly to humans and other tools)

EMACS editor (“*a Lisp interpreter with a screen*”)

SMALLTALK (environment ≡ language)

Equally friendly for human users and programs

## Application Interfaces

(Interfaces dedicated to automation)

Separation of functionality and presentation

Counter-examples: standalone compilers ; debuggers

Internal vs. external interfaces (like in Eclipse)

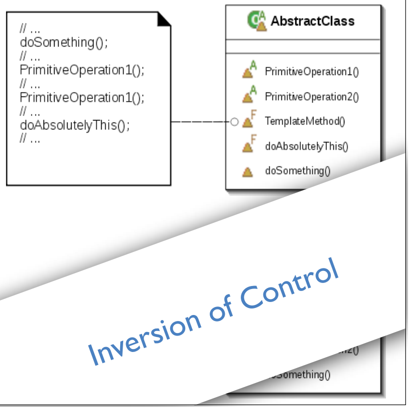
Additional Investment

## Extensible Frameworks

(Explicitly encourage tool integration)

Controlled extensibility (provide “hooks” to extend features)

Internal vs. external interfaces (like in Eclipse)



```
// ...
doSomething();
// ...
PrimitiveOperation1();
// ...
PrimitiveOperation1();
// ...
doAbsolutelyThis();
// ...
```

Inversion of Control

## Extensibility: theEclipse Plugin System

Plugin

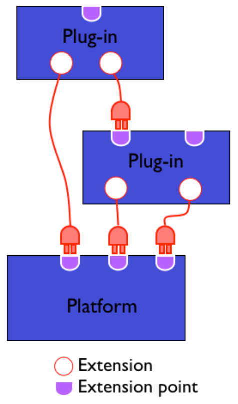
provides functionality to users and other plugins

Extension point

named entity for collecting contributions

Extension

a contribution



## Extensibility: theEclipse Plugin System (2)

```
<plugin
      id = "com.example.tool"
      name = "Example Plug-in Tool"
      class = "com.example.tool.ToolPlugin">
  <requires>
      <import plugin = "org.eclipse.core.resources"/>
      <import plugin = "org.eclipse.ui"/>
  </requires>
  <runtime>
      <library name = "tool.jar"/>
  </runtime>
  <extension
      point = "org.eclipse.ui.preferencepages">
     <page id = "com.example.tool.preferences"
        icon = "icons/knob.gif"
        title = "Tool Knobs"
        class = "com.example.tool.ToolPreferenceWizard"/>
  </extension>
  <extension-point
      name = "Frob Providers"
      id = "com.example.tool.frobProvider"/>
</plugin>
```

Identification

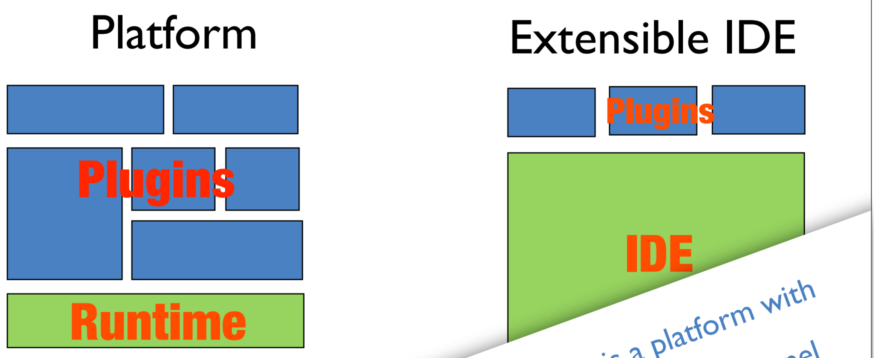
Needed Plugins

Code Location

Declared contribution of this plugin

Declared extension points

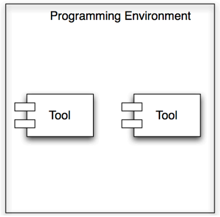
## Platform vs. Extensible IDE



Eclipse is a platform with a small runtime kernel

## 1 Integration

### Lessons



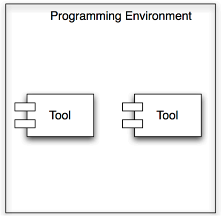
1. *Support automation interfaces*

1b. *...by separating functionality from presentation*

2. *Seek extensibility*

## 1 Integration

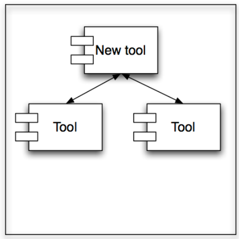
### Trends



*IDEs will...*

1. *increasingly rely on automated, extensible and reusable tools*
2. *serve as universal platforms for new tools*
3. *explicitly foster integration and contribution*

## 2 Synergy



“

syn•er•gy |ˈsinərjē| (also **syn•er•gism** |-ˌjizəm|)
noun
the interaction or cooperation of two or more organizations, substances, or other agents to produce a combined effect greater than the sum of their separate effects : *the synergy between artist and record company.*

”

## Synergy Example: Software Navigation

Connect tasks with its relevant, filtered-out context (Mylyn / Tasktop)

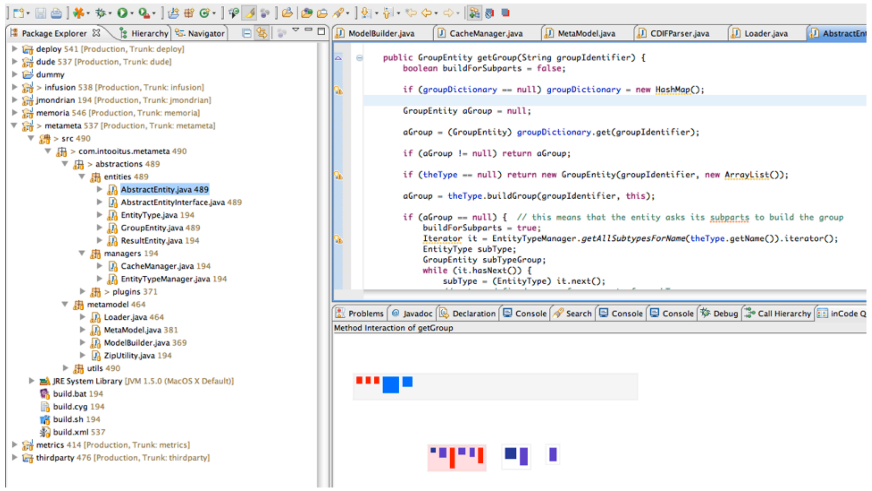
Provide navigation hints based on interaction history

Reveal co-change patterns (eRose - obsolete)

Corelate information about an entity from all project artifacts (Hipikat - obsolete)

## Synergy Example: Software Navigation

(Suggestion of relevant elements)



```
deploy 541 [Production, Trunk: deploy]
dude 537 [Production, Trunk: dude]
dummy
> infusion 538 [Production, Trunk: infusion]
jmondrian 194 [Production, Trunk: jmondrian]
memoria 546 [Production, Trunk: memoria]
> metameta 537 [Production, Trunk: metameta]
    > src 490
        > com.intooitus.metameta 490
            > abstractions 489
                entities 489
                    AbstractEntity.java 489
                    AbstractEntityInterface.java 489
                    EntityType.java 194
                    GroupEntity.java 489
                    ResultEntity.java 194
                managers 194
                    CacheManager.java 194
                    EntityTypeManager.java 194
                > plugins 371
            metamodel 464
                Loader.java 464
                MetaModel.java 381
                ModelBuilder.java 369
                ZipUtility.java 194
            utils 490
    JRE System Library [JVM 1.5.0 (MacOS X Default)]
    build.bat 194
    build.cyg 194
    build.sh 194
    build.xml 537
metrics 414 [Production, Trunk: metrics]
thirdparty 476 [Production, Trunk: thirdparty]
```

```
public GroupEntity getGroup(String groupIdentifier) {
    boolean buildForSubparts = false;

    if (groupDictionary == null) groupDictionary = new HashMap();

    GroupEntity aGroup = null;

    aGroup = (GroupEntity) groupDictionary.get(groupIdentifier);

    if (aGroup != null) return aGroup;

    if (theType == null) return new GroupEntity(groupIdentifier, new ArrayList());

    aGroup = theType.buildGroup(groupIdentifier, this);

    if (aGroup == null) {  // this means that the entity asks its subparts to build the group
        buildForSubparts = true;
        Iterator it = EntityTypeManager.getAllSubtypesForName(theType.getName()).iterator();
        EntityType subType;
        GroupEntity subTypeGroup;
        while (it.hasNext()) {
            subType = (EntityType) it.next();
```

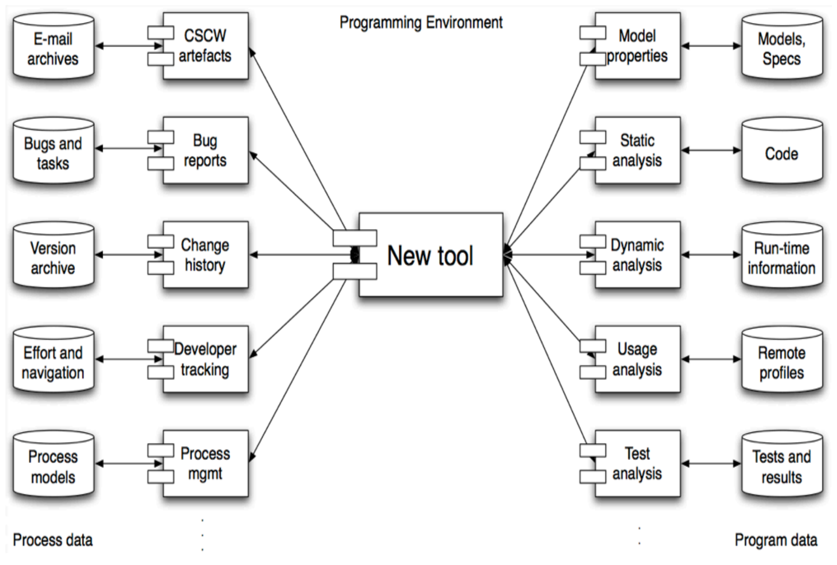
Method Interaction of getGroup

## Delta Debugging on Changes

Isolating code changes that cause a failure

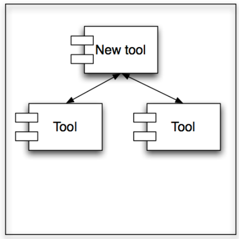
based on:

- automated testing
- change history
- syntactic analysis



## 2 Synergy

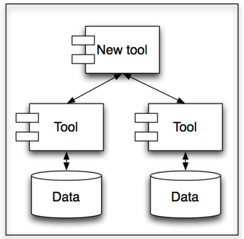
### Trends



*IDEs will...*

1. *collect data from code, runs, and process*
2. *allow tools to combine and leverage such data*
3. *especially support data synergy*

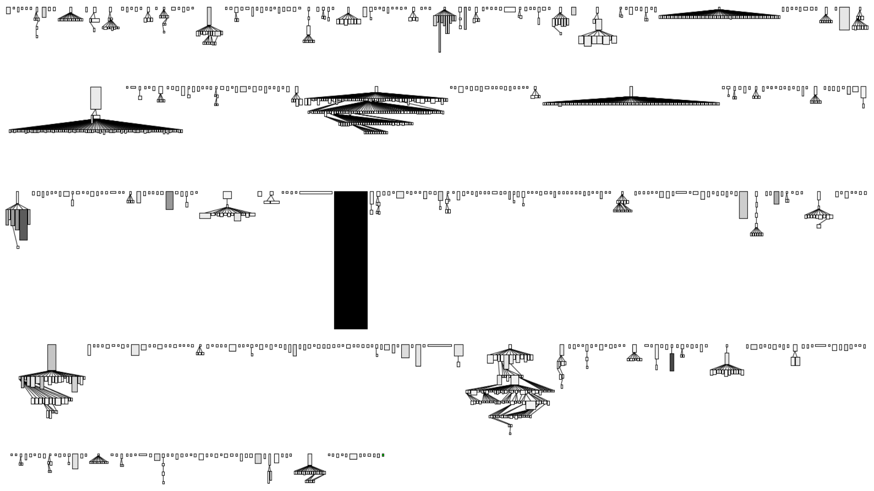
# 3 Assistance



"*Having data [on code and process] available via a programming environment opens the path to all sorts of empirical investigations. [...] The greatest advance, will be the automation of these techniques*"

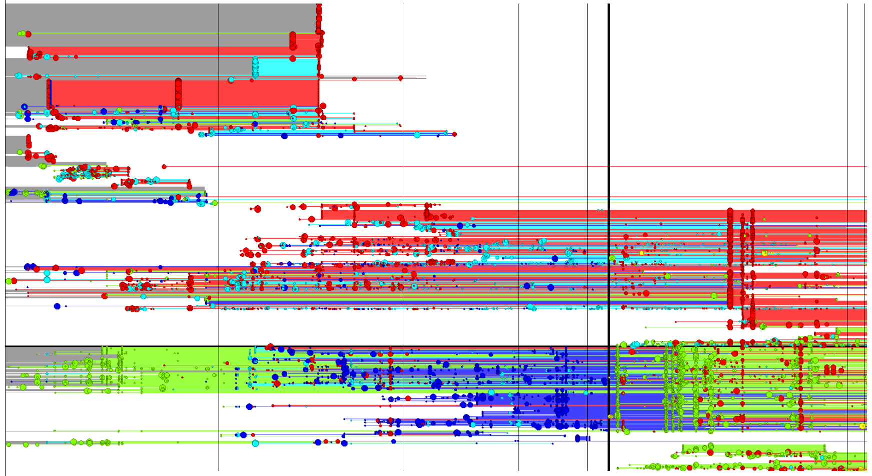
## Automating Data Analysis

(Visualization)



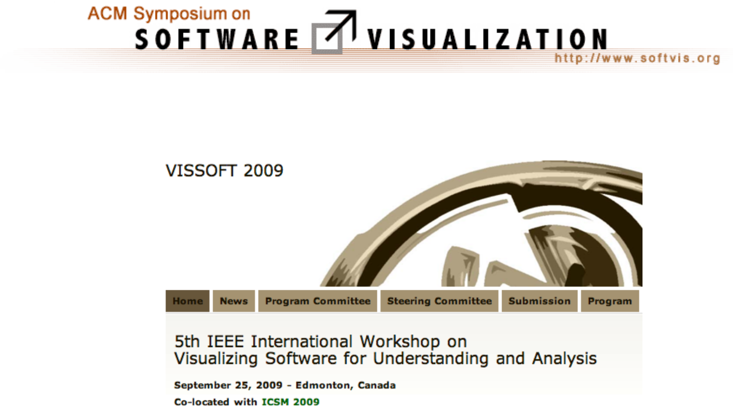
## Automating Data Analysis

(Visualization)



## Automating Data Analysis

(Visualization)



ACM Symposium on SOFTWARE VISUALIZATION
http://www.softvis.org

VISSOFT 2009

Home | News | Program Committee | Steering Committee | Submission | Program

5th IEEE International Workshop on
Visualizing Software for Understanding and Analysis

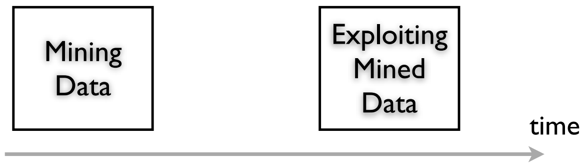
September 25, 2009 - Edmonton, Canada
Co-located with ICSM 2009

## Automating Data Analysis

(Data Mining & Machine Learning)

MSR 2010
Mining Software Repositories

MSR 2010: 7th IEEE Working Conference on Mining Software Repositories
http://www.msrconf.org



## Assisted Decisions

Why in IDEs?

they provide data & implement consequences

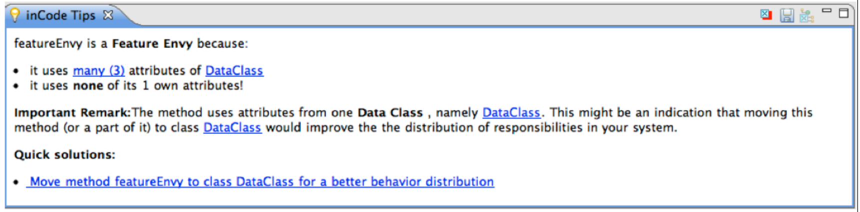
What Assistance?

Better Code & Design

Predict Effort and Risk

Give Rationals

## Assistance Example: inCode Tips



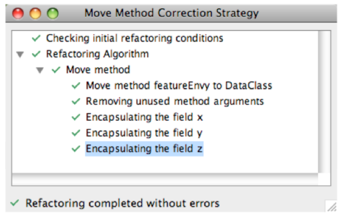
featureEnvy is a **Feature Envy** because:

- it uses many (3) attributes of DataClass
- it uses **none** of its 1 own attributes!

**Important Remark:** The method uses attributes from one **Data Class** , namely DataClass. This might be an indication that moving this method (or a part of it) to class DataClass would improve the the distribution of responsibilities in your system.

**Quick solutions:**

- Move method featureEnvy to class DataClass for a better behavior distribution



```
✓ Checking initial refactoring conditions
▼ ✓ Refactoring Algorithm
    ▼ ✓ Move method
        ✓ Move method featureEnvy to DataClass
        ✓ Removing unused method arguments
        ✓ Encapsulating the field x
        ✓ Encapsulating the field y
        ✓ Encapsulating the field z
✓ Refactoring completed without errors
```

“*As a goal, assistance in a programming environment should be like a good navigator in pair programming: monitoring actions over the driver's shoulder, and knowing what to say, and when to say it*”

## Assisted Decisions

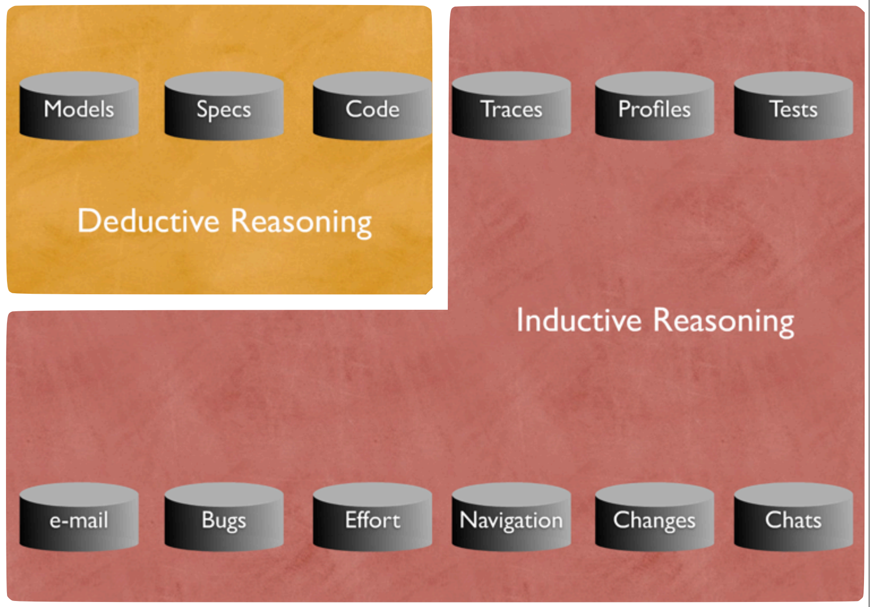
(Issues and Risks)

User Interface (balance between annoying and passive)

Accuracy (balance between false positive and false negatives)

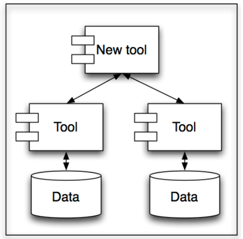
Interpretation (drawing wrong conclusions harming developers)

Research (balance between inductive and deductive processes)



## 3 Assistance

### Trends



*IDEs will…*

1. *mine patterns from program and process data*
2. *apply rules to make predictions*
3. *provide assistance in all development decisions*

28290326
Reverse Engineering