Defect Density Estimation through Verification and Validation

Mark Sherriff and Laurie Williams
North Carolina State University

HCSS ’06
April 19, 2006

Agenda

- Motivation and Hypothesis
- Research Questions
- DevCOP Approach
- Limitations / Open Research Questions
- Questions

Motivation

- Software Reliability
  - Often not estimated until development is complete
  - Actual reliability not known until system is shipped to customers
- Corrective action is more expensive later in the process
- If defect density could be estimated during development…
  - Steps could be taken to address issues early
  - More economical, could improve development effort
- Or, as John Musa says: More Reliable Software Faster and Cheaper

Hypothesis

- Defect density estimation can be based upon the history of verification and validation techniques that have been performed on the project.
- My approach to investigating this hypothesis:
  - DevCOP (Defect Estimation with V&V Certificates on Programming)

Questions that need to be answered:

- What is the best way to record V&V techniques?
- How do I build a model that can predict defect density with V&V information?

Challenges in Recording V&V Efforts

- Evidence and artifacts from V&V techniques come in numerous forms and are stored in different locations
  - Logs from test case runs stored in a versioning system
  - Reports from code inspections stored in different project directories
  - Details on pair programming assignments stored in an agile process management system
  - Assuming this information is even gathered in the first place!
- If the information is actively being stored…
  - Is it being maintained? How much does it cost to maintain it?
  - Is it being used to improve the development process?
  - What is done with this information when the project is "finished"?

Software Certificates

- Software Certificates are…
  - A record of a verification and validation (V&V) practice employed by developers and used to support traceability between code and evidence of the V&V technique used
  - Some examples could be a V&V database or an XML file used to store this information

- Solves the problem of storing V&V information i different formats…
- … but still requires manual intervention to maintain the information and to make use of it.

Recording V&V Techniques

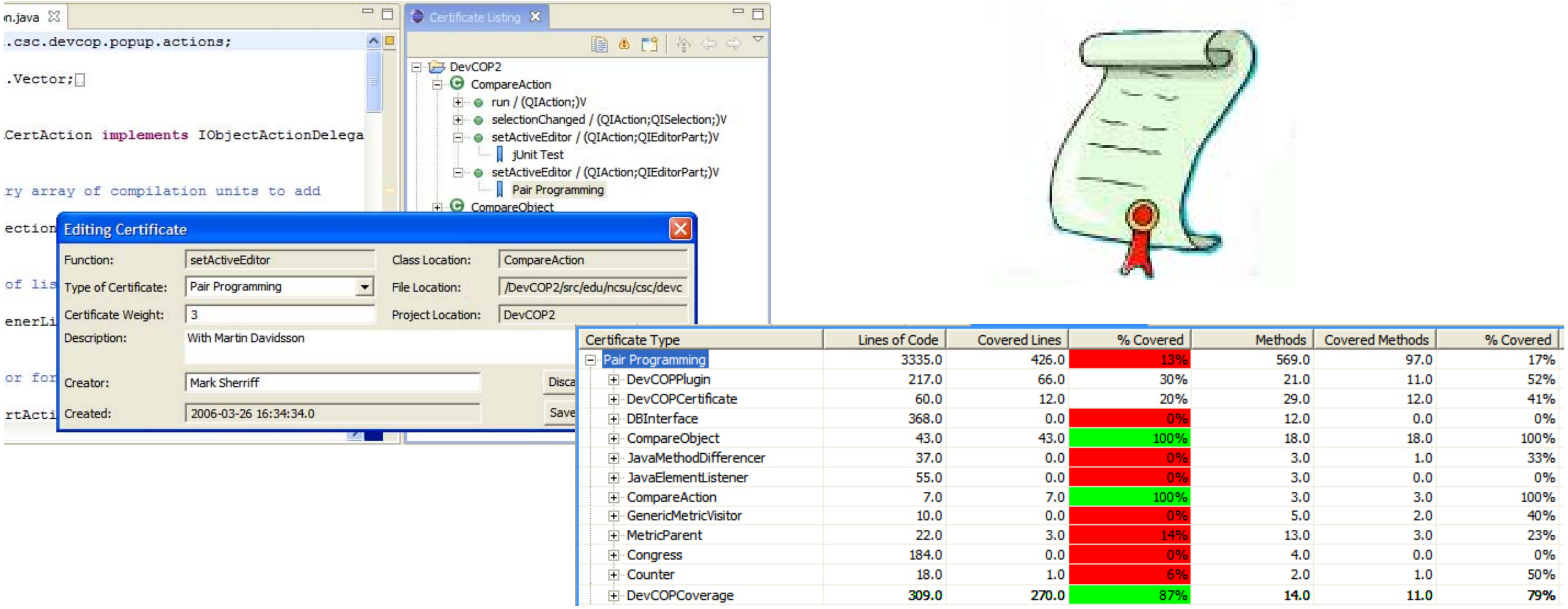
- Software Certificate Management Systems (SCMS)
  - Tool support for software certificates
  - Provides an interface and infrastructure to automatically create, maintain, and analyze software certificates
- Current research:
  - OGI/PSU: Programatica, a SCMS for Haskell

Benefits of an SCMS

- Minimize developer overhead in using certificates
- Encourage developer responsibility
- Software maintenance
  - Maintainers can quickly determine who to talk to and what has already been performed to find a defect
- Analysis of V&V technique effectiveness
  - Once failures are reported from customers, developers can see how different V&V techniques performed
- Reference in future projects
  - Code that reused from previous projects can carry some evidence of what was done to make that component reliable

Our SCMS Implementation

- DevCOP SCMS Eclipse Plug-in
  - DevCOP V&V Certificates



```
.csc.devcop.popup.actions;

.Vector;

CertAction implements IObjectActionDelega

ry array of compilation units to add

ection

of lis

enerLi

or for

rtActi
```

```
DevCOP2
  CompareAction
    run / (QIAction;)V
    selectionChanged / (QIAction;QISelection;)V
    setActiveEditor / (QIAction;QIEditorPart;)V
      jUnit Test
    setActiveEditor / (QIAction;QIEditorPart;)V
      Pair Programming
  CompareObject
```

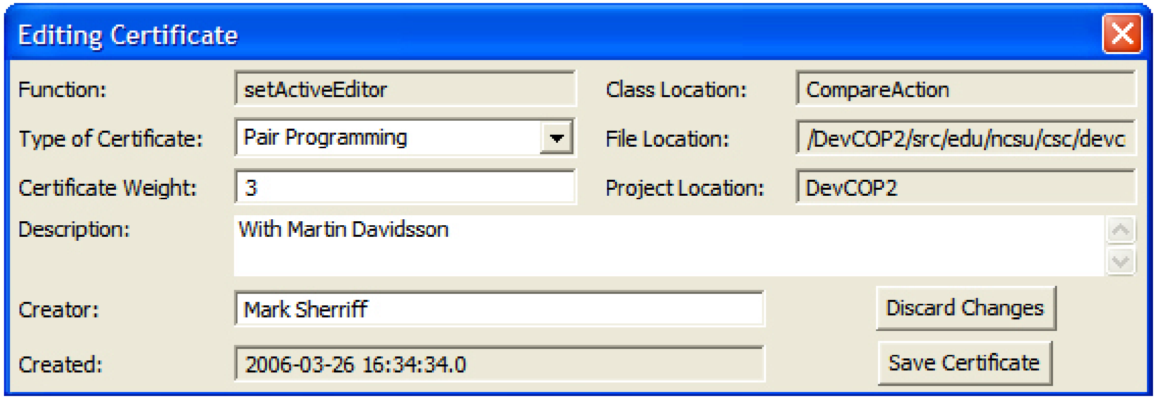
Editing Certificate

Function:	setActiveEditor	Class Location:	CompareAction
Type of Certificate:	Pair Programming	File Location:	/DevCOP2/src/edu/ncsu/csc/devc
Certificate Weight:	3	Project Location:	DevCOP2
Description:	With Martin Davidsson		
Creator:	Mark Sherriff		
Created:	2006-03-26 16:34:34.0		

Certificate Type	Lines of Code	Covered Lines	% Covered	Methods	Covered Methods	% Covered
Pair Programming	3335.0	426.0	13%	569.0	97.0	17%
DevCOPPlugin	217.0	66.0	30%	21.0	11.0	52%
DevCOPCertificate	60.0	12.0	20%	29.0	12.0	41%
DBInterface	368.0	0.0	0%	12.0	0.0	0%
CompareObject	43.0	43.0	100%	18.0	18.0	100%
JavaMethodDifferencer	37.0	0.0	0%	3.0	1.0	33%
JavaElementListener	55.0	0.0	0%	3.0	0.0	0%
CompareAction	7.0	7.0	100%	3.0	3.0	100%
GenericMetricVisitor	10.0	0.0	0%	5.0	2.0	40%
MetricParent	22.0	3.0	14%	13.0	3.0	23%
Congress	184.0	0.0	0%	4.0	0.0	0%
Counter	18.0	1.0	6%	2.0	1.0	50%
DevCOPCoverage	309.0	270.0	87%	14.0	11.0	79%

DevCOP Certificates

- A DevCOP certificate contains:
  - identifying information for the function;
  - identifying information for the developer that created it;
  - the type of V&V technique used;
  - Descriptive information about the technique used; and
  - a hash of the function’s abstract syntax tree (AST).



Editing Certificate

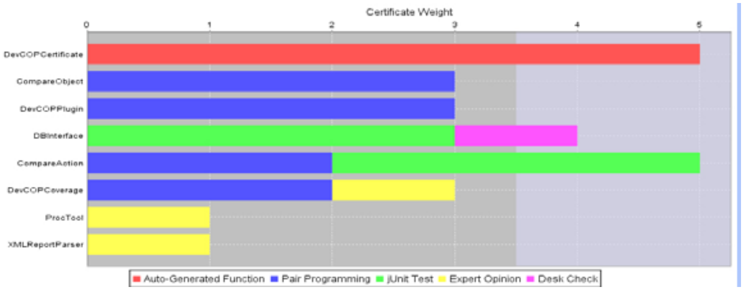
Function:	setActiveEditor	Class Location:	CompareAction
Type of Certificate:	Pair Programming	File Location:	/DevCOP2/src/edu/ncsu/csc/devc
Certificate Weight:	3	Project Location:	DevCOP2
Description:	With Martin Davidsson		
Creator:	Mark Sherriff		
Created:	2006-03-26 16:34:34.0		

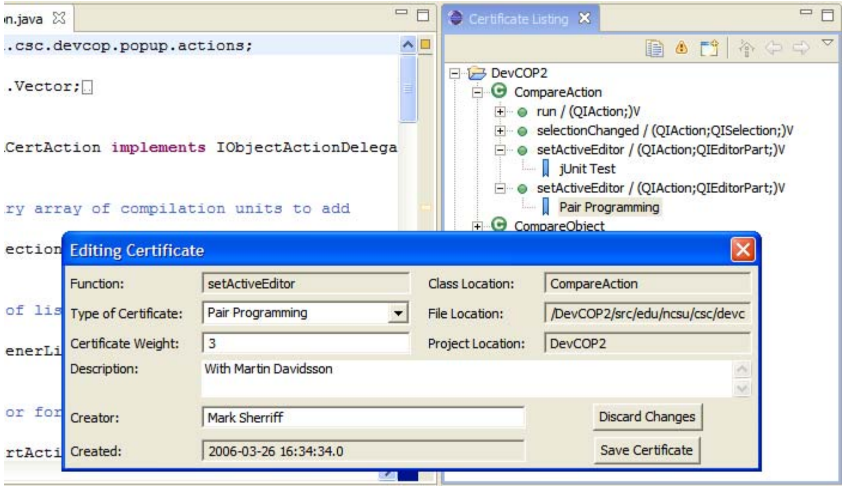
DevCOP Certificates

Type	Description	Examples
Manual	All manual checking performed by people	pair programming, code inspections, desk check, audits
Automated Static	Performed automatically on uncompiled code	static analysis tools, continuous compiling techniques
Dynamic	Includes all techniques performed at run time	black box testing, white box testing, system testing
Formal	All formal and mathematical techniques	lambda calculus, formal proofs

DevCOP SCMS Eclipse Plug-in

- Currently supports manual V&V techniques and jcoverage certificates
- Provides different methods for examining and managing certificate data
- Demo





```
.csc.devcop.popup.actions;

.Vector;

CertAction implements IObjectActionDelega

ry array of compilation units to add

ection

of lis

enerLi

or for

rtActi
```

DevCOP2
CompareAction
run / (QIAction;)V
selectionChanged / (QIAction;QISelection;)V
setActiveEditor / (QIAction;QIEditorPart;)V
jUnit Test
setActiveEditor / (QIAction;QIEditorPart;)V
Pair Programming
CompareObject

Editing Certificate

Function: setActiveEditor

Class Location: CompareAction

Type of Certificate: Pair Programming

File Location: /DevCOP2/src/edu/ncsu/csc/devc

Certificate Weight: 3

Project Location: DevCOP2

Description: With Martin Davidsson

Creator: Mark Sherriff

Created: 2006-03-26 16:34:34.0

Building a Model

- We now have a means for collecting V&V information (with some added benefits)
- Still need to answer the second question:
  - How do I build a model that can predict defect density with this V&V information?
- Several different modeling techniques available

Parametric Modeling

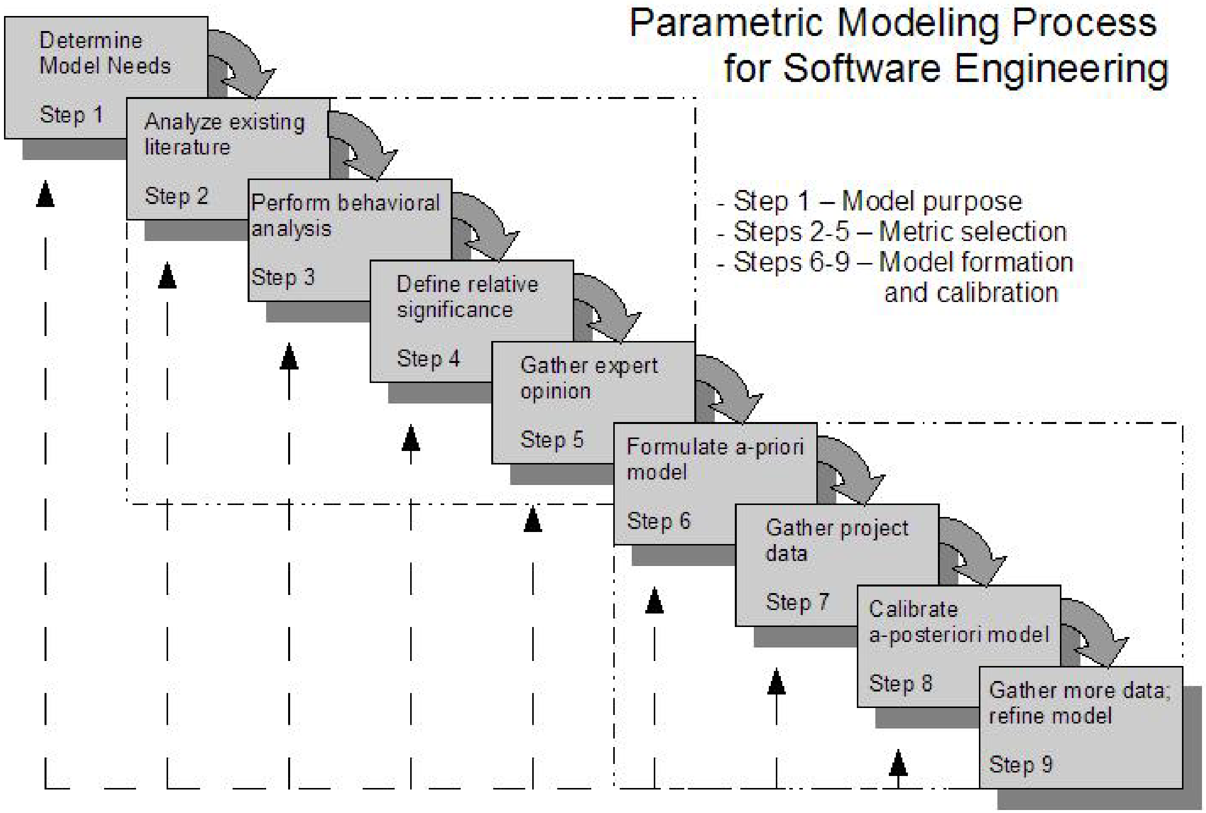
- Method by which dependant variables are related to one or more independent variables with regard to previous data
- In Software Engineering…
  - Purpose is to provide an estimated answer to a software development question earlier in the development lifecycle
- Famous SE parametric models: COCOMO 81 and COCOMO II

Parametric Modeling

- **S**oftware **T**esting **R**eliability **E**arly **W**arning
  - Java and Haskell versions
  - Uses a suite of metrics gathered on static code to provide a reliability estimate (includes code complexity, testing effort, and code size metrics)
  - The model is calibrated to a particular organization using a regression equation
- If a reliability estimate can be created from testing and static metrics, could it be improved if we added other verification and validation information to the model?

Our Parametric Model Implementation

- DevCOP Parametric Model



Parametric Modeling Process
for Software Engineering

Determine Model Needs
Step 1

Analyze existing literature
Step 2

Perform behavioral analysis
Step 3

Define relative significance
Step 4

Gather expert opinion
Step 5

Formulate a-priori model
Step 6

Gather project data
Step 7

Calibrate a-posteriori model
Step 8

Gather more data; refine model
Step 9

- Step 1 – Model purpose
- Steps 2-5 – Metric selection
- Steps 6-9 – Model formation and calibration

Building the Model

- Defect density prediction created from historical information from similar projects
  - V&V coverage information by technique
  - Past defect density / “trouble reports”
  - Generates weighting coefficient for each V&V type

$$DefectDensity = a + \sum_{j=1}^{certificate\_type} (c_j * Size_j)$$

Limitations / Open Questions

- Granularity of Certificates
  - Method level, not class or line of code
- Composition of Certificates
  - The effect of composing V&V techniques is still an open research question
- Defect Severity
  - All defects are treated equally
- Manual Intervention
  - Since users can interact directly with certificates, there is an issue of trust

Where the Research Stands

- Background research
- Plug-in ready for use
- Need in-process data to start building models to test

Future Work

- Other Languages
- Packaging Certificates

Thank you!

Questions? Queries? Quandaries?

Contact Information:

Mark Sherriff

mark.sherriff@ncsu.edu

DevCOP Project: http://agile.csc.ncsu.edu/devcop/

Download the plug-in! Let me know what you think!