Safe NAnt

a simple NAnt IDE

Daniel Quinta Leonel Braga Rui Gonçalo

University of Minho
Department of Informatics
Formal Methods in Software Engineering

February 23, 2011

Index

NAnt

- free and open source software tool for automating software build processes
- similar to Apache *Ant*
- reads a xml file
- OS execute the specified tasks

Short example

```
<project name="proj.name" default="first">
  <property name="a" value="prop.a" readonly="true" />

  <target name="first">
    <mkdir dir="folder" />
    <call target="second" />
  </target>
  <target name="second" depends="third">
    <delete dir="folder" />
  </target>
  <target name="third">
    <property name="b" value="prop.b" />
  </target>
</project>
```

Goals

- Implement a simple *Integrated Development Environment* in C#
- Alloy model as background
- Define contracts
- Measurement of software quality
- Unit testing

Application entities

- Project
  - Target
  - *abstract* Task
    - Property
    - Call
    - Delete
    - MkDir
- User Interface
- XML Parser
- XML Writer

Application requirements

- read and write *build* files
- implement the desired operations
- based on Alloy model invariants
  - no targets cyclic dependencies
  - no double delete of the same directory in a row
  - no double mkdir of the same directory in a row
  - preserve the readonly condition of property task

Operations

Alloy

C#

addTarget[p,p': Project, t: Target]

public void addTarget(string name)

removeTarget[p,p': Project, t: Target]

public bool removeTarget(string name)

addDependency[p,p': Project, tdep,t: Target]

public void addDependency(string src, string dst)

removeDependency[p,p': Project, tdep,t: Target]

public void removeDependency(string src, string dst)

addTask[p,p':Project, t: Target, ta: Task, i: SeqIdx, aux: Seq]

public void addTask2Target(string trg, Task tsk, int index)

Code Contracts

- allow the specification of preconditions, postconditions and object invariants
- well embedded on Visual Studio
- provide runtime analysis

Alloy and Contracts Preconditions

```
pred addDependency[p,p': Project, tdep,t: Target]
  tdep != t
  (t+tdep)  in p.targets
  tdep not in t.^(p.depends + CallGraph[p])
```

Contracts Preconditions of addDependency method

```
Contract.Requires<Exception>
(!String.IsNullOrWhiteSpace(src),"");
Contract.Requires<Exception>
(!String.IsNullOrWhiteSpace(dst),"");
Contract.Requires<Exception>
(String.Compare(src, dst) != 0,"");
Contract.Requires<Exception>
(this.TargetList.ContainsKey(src),"");
Contract.Requires<Exception>
(this.TargetList.ContainsKey(dst),"");
Contract.Requires<Exception>
(!this.getAllDependencies(this.getTargetByKey(src)).Contains
    (this.getTargetByKey(dst)),"");
```

Alloy and Contracts Postconditions

```
pred removeDependency[p,p': Project, tdep,t : Target]
  p'.default = p.default
  p'.targets = p.targets
  p'.depends = p.depends - (tdep->t)
  p'.gtasks = p.gtasks
  p'.ttasks = p.ttasks
```

Contracts Postconditions of removeDependency method

```
Contract.EnsuresOnThrow<Exception>(!this.getTargetByKey(src)
    .Dependencies.Contains(this.getTargetByKey(dst)), "");
```

Alloy and Contracts Invariants

```
pred inv[ p : Project]
  DefaultIn[p]
  NoEqualTargetNames[p]
  NoCyclicDependencies[p]
  DependsTargetInProject[p]
  CallTargetsInProject[p]
  ReadOnlyProperty[p]
  NoDoubleDelete[p]
```

Contracts Invariant of project class

```
[ContractInvariantMethod]
void objectInvariant() {
Contract.Invariant(Contract.ForAll(TargetList.Values, x => !
    getAllDependencies(x).Contains(x)));
Contract.Invariant(this.insertPropertyOk());
Contract.Invariant(DefaultTarget==null || (TargetList.
    ContainsKey(DefaultTarget.TargetName)));
Contract.Invariant(!hasRepeatedElem());
Contract.Invariant(Contract.ForAll(TargetList.Values, x =>
    Contract.ForAll(x.Dependencies, dep => TargetList.
    ContainsKey(dep.TargetName))));
```

Safe NAnt is **safe**!

NAnt tool doesn't check the dependencies of Call tasks:

```
<project name="proj.name" default="first">
  <target name="first">
    <call target="second" />
  </target>
  <target name="second">
    <call target="first" />
  </target>
</project>
```

But, *Safe NAnt* checks with Contracts:

```
Contract.Requires<Exception>(this.TargetList.ContainsValue(t
    ), "");
Contract.Requires<Exception>(implies(t != null,
!getAllDependencies(t).Contains(c.Target) &&
!t.TargetName.Equals(c.Target.TargetName)), "");
```

Unit testing

incomplete!

- Microsoft Pex
  - generates parameters to parameterized testing
  - explores code and tries to invalidate assertions
  - good to find software bugs and errors
- NUnit
  - is an open source unit testing framework for Microsoft .NET

Unit testing example

```
//Tests add target with name null
[PexMethod(), PexAllowedException(typeof(ArgumentNullException))]
[ExpectedException( typeof(ArgumentNullException))]
public void addTargetStringTestArgumentNullException(string name)
{
  Project p = new Project();
  p.addTarget(name);
}

//Tests add duplicated target with a given string
[PexMethod(), PexAllowedException(typeof(TargetNameExistsException))]
[ExpectedException(typeof(TargetNameExistsException))]
public void addTargetStringTestTargetNameException(string name)
{
  PexAssume.IsTrue(!String.IsNullOrWhiteSpace(name) && !String.IsNullOrEmpty(
       name));
  Project p = new Project();
  p.addTarget(name);
  p.addTarget(name);
}
```

Software Quality

Analysis:

- Static analysis
  - Volume
  - Complexity per unit
  - Duplication
  - Unit size
- Dynamic analysis
  - unit testing (before)

Tools:

- SourceMonitor
  - Volume
  - Complexity per unit
  - Unit size
- Duplo
  - Duplication

Volume

Production Code		Test Code	
Language conversion factor	–	Language conversion factor	–
Man Years	–	Man Years	–

Ranking	Complies	Ranking	Complies
++	–	++	–
+	–	+	–
o	–	o	–
-	–	-	–
- -	–	- -	–

Decision	–	Decision	–

Complexity per unit

Risk Level	% LOC	Relative LOC
Low	95.7	1079 of 1127
Moderate	4.3	48 of 1127
High	0	0 of 1127
Very high	0	0 of 1127

Ranking	Complies
++	yes
+	no
o	no
-	no

Decision	++

Unit size

Risk Level	% LOC	Relative LOC
Low	78.44	884 of 1127
Moderate	21.56	243 of 1127
High	0	0 of 1127
Very high	0	0 of 1127

Ranking	Complies
++	yes
+	no
o	no
-	no
Decision	++

Duplication

Nr of duplicated lines	0.039
Duplication factor %	3.866

Ranking	Complies
++	no
+	yes
o	no
-	no
Decision	+

Overall

	Volume	Complexity per unit	Duplication	Unit size	Unit testing	
	++	++	+	++	o	**Decision**
Analysability	x		x	x	x	
Changeability		x	x	x		
Stability					x	
Testability		x	x	x	x	

Conclusions