3TU.

NIRICT
Netherlands Institute for Research on ICT

Automated Software Testing and Release with Nix Build Farms

Eelco Dolstra
Universiteit Utrecht
Department of Information and Computing Sciences

Eelco Visser
Technische Universiteit Delft
Department of Software Technology

The Goal: Building and Testing Software

- It is good development practice to build and test a software system every time a developer commits a change to the project's version management repository.
- A **build farm** supports this:it's a set of machines that continuously builds and tests software components from a version management system, producing status reports and/or releases.
- It is called a build farm because the software typically must be tested in many hardware / operating system configurations (Windows, Linux, Mac OS X, 32 bits, 64 bits, etc.) - so **a large number of (virtual) machines** is required.
- A build farm allows many quality aspects of a project to be monitored:

Does it build correctly?

```
Making all in nix-store
building help.txt.hh
Making all in nix-hash
building help.txt.hh
Making all in libexpr
building nixexpr-ast.hh
make[3]: Entering directory `/tmp/nix-13939-3/nix-0.10pre6460/src/libexpr'
building all
make all-am
building nixexpr.lo
make[4]: Entering directory `/tmp/nix-13939-3/nix-0.10pre6460/src/libexpr'
if /bin/sh ../../libtool --tag=CXX --mode=compile g++ -DHAVE_CONFIG_H -I. -I. -I../.. -I../.. -I/...-db4-4.4.20/include
-I/...-aterm-2.4.2/include -I./../libutil -I./../libstore -g -O2 -MT nixexpr.lo -MD -MP -MF ".deps/nixexpr.Tpo" -c -o
nixexpr.lo nixexpr.cc; \" then mv -f ".deps/nixexpr.Tpo" ".deps/nixexpr.Plo"; else rm -f ".deps/nixexpr.Tpo"; exit 1; fi
mkdir .libs
g++ -DHAVE_CONFIG_H -I. -I. -I../.. -I./.. -I/...-db4-4.4.20/include -I/...-aterm-2.4.2/include -I./../libutil
-I./../libstore -g -O2 -MT nixexpr.lo -MD -MP -MF .deps/nixexpr.Tpo -c nixexpr.cc -fPIC -DPIC -o .libs/nixexpr.o
nixexpr-ast.hh:6: error: 'AFun' does not name a type
nixexpr-ast.hh: In function '_ATerm* nix::makePos(_ATerm*, int, int)':
nixexpr-ast.hh:10: error: 'symPos' was not declared in this scope
nixexpr-ast.hh:10: error: 'ATmakeInt' was not declared in this scope
nixexpr-ast.hh:10: error: 'ATmakeApl3' was not declared in this scope
nixexpr-ast.hh: In function 'bool nix::matchPos(_ATerm*, _ATerm*&, int&, int&)':
nixexpr-ast.hh:15: error: 'ATgetType' was not declared in this scope
nixexpr-ast.hh:15: error: 'AT_APPL' was not declared in this scope
nixexpr-ast.hh:15: error: 'AFun' was not declared in this scope
```

Does it pass the test suite?

```
List with some elements
strategy failed
List with element of illegal type
List with element of illegal type
Empty list
[ lt-dfta-accept-tests | critical ] No productive start symbols
left in rtg
RTG(Start([]),ProdRules([]))
FAIL: dfta-accept-tests
========================================
1 of 2 tests failed
Please report to stratego-bugs@cs.uu.nl
========================================
make[4]: *** [check-TESTS] Error 1
make[4]: Leaving directory
`/tmp/nix-24398-5/svn-export/stratego-libraries/rtg/tests'
make[3]: *** [check-am] Error 2
```

Is it portable?

```
Making all in nix-setuid-helper
if g++ -DHAVE_CONFIG_H -I. -I. -I../.. -I./.. -I/...-aterm-2.4.2-fixes/include -I./../libutil
-D_FILE_OFFSET_BITS=64 -g -O2 -MT main.o -MD -MP -MF ".deps/main.Tpo" -c -o main.o main.cc; \
then mv -f ".deps/main.Tpo" ".deps/main.Po"; else rm -f ".deps/main.Tpo"; exit 1; fi
main.cc: In function `void secureChown(unsigned int, unsigned int, unsigned
int, unsigned int, const nix::Path&)':
main.cc:49: error: `lchown' undeclared (first use this function)
main.cc:49: error: (Each undeclared identifier is reported only once for each
function it appears in.)
main.cc: In function `void runBuilder(unsigned int, unsigned int, const
nix::StringSet&, const std::string&, std::basic_string<char,
std::char_traits<char>, std::allocator<char> >, int, char**, char**)':
main.cc:101: warning: passing negative value `-1' for argument passing 2 of `
void secureChown(unsigned int, unsigned int, unsigned int, unsigned int,
const nix::Path&)'
main.cc:101: warning: argument of negative value `-1' to `unsigned int'
main.cc: In function `void run(int, char**)':
main.cc:228: warning: passing negative value `-1' for argument passing 1 of `
void secureChown(unsigned int, unsigned int, unsigned int, unsigned int,
const nix::Path&)'
main.cc:228: warning: argument of negative value `-1' to `unsigned int'
make[3]: *** [main.o] Error 1
```

Build Farm Results for Package strategoxt

Note: there is also a overview of the latest build results per package.

Package	Release	Rev	All	Source tarball	i686-linux	i686-darwin	powerpc-darwin	i686-cygwin	Red Hat 9.0	Fedora Core 2	Fedora Core 3	SuSE 9.0	Check	Coverage
strategoxt	0.17M2pre15838	15838	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓✓✓	
strategoxt	0.17M2pre15837	15837	✗	✓	✓	✓	✓		✓	✓	✓	✓	✓✓✗	
strategoxt	0.17M2pre15836	15836	✗	✗	✗	✗	✗		✗	✗	✗	✗	✓✓✗	
strategoxt	0.17M2pre15835	15835	✗	✗	✗	✗	✗		✗	✗	✗	✗	✓✓✗	
strategoxt	0.17M2pre15831	15831	✗	✗	✗	✗	✗		✗	✗	✗	✗	✓✓✗	
strategoxt	0.17M2pre15819	15819	✗	✗	✗	✗	✗		✗	✗	✗	✗	✓✓✗	
strategoxt	0.17M2pre15809	15809	✗	✓	✗	✗	✗		✓	✓	✓	✓	✓✓✗	
strategoxt	0.17M2pre15799	15799	✗	✓	✗	✗	✗		✗	✗	✗	✗	✓✓✗	
strategoxt	15784-bad	15784	✗	✗	✗	✗	✗		✗	✗	✗	✗	✗✗✗	
strategoxt	0.17M2pre15779	15779	✗	✓	✗	✗	✗		✗	✗	✗	✗	✓✓✗	
strategoxt	0.17M2pre15767	15767	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓✓✓	
strategoxt	0.17M2pre15761	15761	✗	✗	✗	✗	✗		✗	✗	✗	✗	✓✓✗	
strategoxt	0.17M2pre15760	15760	✗	✗	✗	✗	✗		✗	✗	✗	✗	✓✓✗	
strategoxt	15757-bad	15757	✗	✗	✗	✗	✗		✗	✗	✗	✗	✓✗✗	
strategoxt	15755-bad	15755	✗	✗	✗	✗	✗		✗	✗	✗	✗	✓✗✗	
strategoxt	15754-bad	15754	✗	✗	✗	✗	✗		✗	✗	✗	✗	✓✗✗	

Run static/dynamic analyses

LTP GCOV extension - code coverage report

Current view: directory - src/libexpr
Test: app.info
Date: 2006-11-14 — Instrumented lines: 1842
Code covered: 86.2 % — Executed lines: 1588

Filename	Coverage	
attr-path.cc	85.3 %	29 / 34 lines
eval.cc	90.8 %	356 / 392 lines
expr-to-xml.cc	92.9 %	52 / 56 lines
get-drvs.cc	79.4 %	77 / 97 lines
get-drvs.hh	50.0 %	2 / 4 lines
lexer.l	96.0 %	72 / 75 lines
nixexpr-ast.cc	100.0 %	121 / 121 lines
nixexpr-ast.hh	95.4 %	293 / 307 lines
nixexpr.cc	78.1 %	168 / 215 lines
nixexpr.hh	100.0 %	13 / 13 lines
parser.y	96.6 %	171 / 177 lines
primops.cc	66.7 %	234 / 351 lines

Produce downloadable releases

PHP-SAT, the PHP static analysis tool release php-sat-0.1pre286

This page provides release **php-sat-0.1pre286** of PHP-SAT, the PHP static analysis tool. It was generated automatically on 2006-11-14 22:13:35 UTC from revision 286 of the path /php-sat/trunk of its Subversion repository (the XML record of the build job is available).

Distribution

Binary archive for Microsoft Windows

- php-sat.zip (10642950 bytes; MD5 hash: 9ce5bb9f87a613803547cece51c1d451)

RPM for Red Hat 9.0

- php-sat-0.1pre286-1.i386.rpm (145051 bytes; MD5 hash: fcfdcd512e3c9e6e548d0bbbb0647bba)
- php-sat-0.1pre286-1.src.rpm (551573 bytes; MD5 hash: f06c9bfc1ac95041ce52ab61e7df64a9)

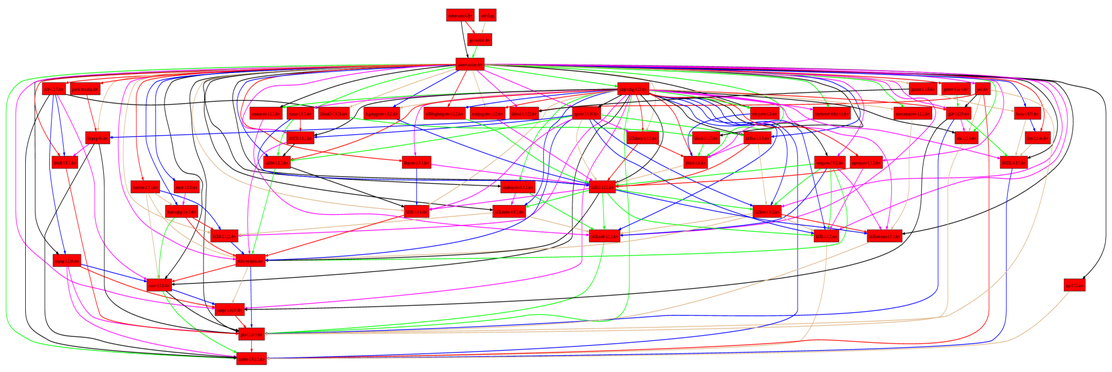
This RPM requires that the following packages are also installed:

- aterm-2.4.2-1.i386.rpm
- php-front-0.1pre287-1.i386.rpm
- sdf2-bundle-2.3.4pre15345-1.i386.rpm
- strategoxt-0.17M3pre15898-1.i386.rpm

SuSE RPM for SuSE 9.0

The Problem: Managing the Complexity of the Build Environment

- Non-trivial software packages typically have a **large number of dependencies**.



Example: build-time dependency graph of **Mozilla Firefox** (on Linux).

- So if we need to build a package with N dependencies on M configurations, then to install and manage those dependencies takes **N x M effort**!
- And what if there are **conflicting dependencies**? E.g., package A builds with GCC 3.3 but not GCC 4.1, while package B needs at least GCC 4.1.
- Finally, the (virtual) machines themselves need to be set up and maintained.

The Solution: The Nix Deployment System

- **Nix** (http://nix.cs.uu.nl/) is a **purely functional package management system**.
- Packages are built from pure functions (Nix expressions), i.e., the build result only depends on the declared inputs and never changes after it has been built.
- Packages are stored in a **Nix store** under a name that contains a **cryptographic hash of all package inputs**:

```
/
└ usr
  ├ bin
  │ ├ gcc
  │ ├ firefox
  │ └ acroread
  ├ lib
  │ ├ libgtk-x11.so.0
  │ ├ libstdc++.so.6
  │ └ libxml2.so.2
  ├ include
  │ ├ stdlib.h
  │ └ zlib.h
  └ man
    └ ...

/nix/store
├ 66qra1164b1s...-gtk+-2.10.9
│ └ lib
│   └ libgtk-x11-2.0.so.0
├ i33jmpd2p14r...-firefox-2.0.0.2
│ └ bin
│   └ firefox
├ kpqz9a4clx96...-gcc-4.1.1
│ ├ bin
│ │ └ gcc
│ └ lib
│   └ libstdc++.so.6
└ s749sha8pf73...-gcc-3.3.6
  ├ bin
  │ └ gcc
  └ lib
    └ libstdc++.so.5
```

Unique paths for different versions

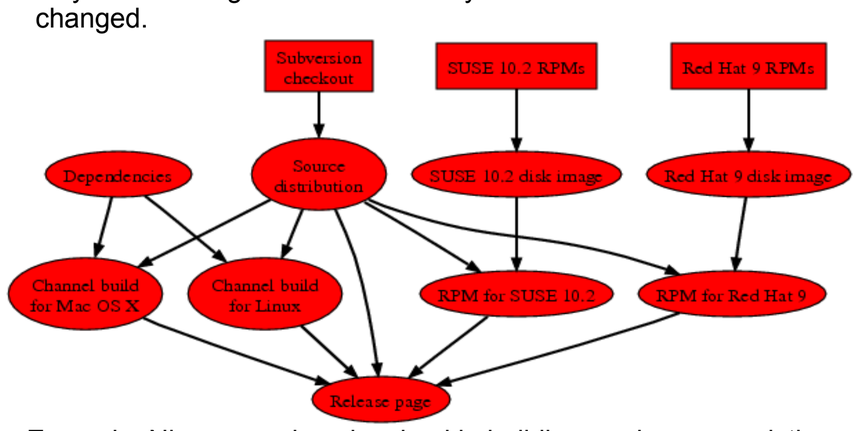
Why is this useful for a build farm?

- As a component build language, the Nix expression language is ideal for **describing the build tasks** to be performed.
- As a functional language, the Nix expression language makes it easy to **describe variants**.
- **Virtual machines** can be built and used on the fly in a Nix expression from a declarative specification.
- Nix **manages the storage** of the dependencies.
- Nix supports **distributed multi-platform builds** transparently.
- The hashing scheme + complete dependencies allow builds to be **reproduced** reliably.
- **Efficiency**: due to the hashing scheme, we only rebuild things that have actually changed.

```
{stdenv, fetchurl, perl}:

stdenv.mkDerivation {
  name = "hello-2.1.1";
  src = fetchurl {
    url = .../hello-2.1.1.tar.gz;
    md5 = "70c9ccf9fac0...";
  };
  buildInputs = [perl];
}
```

Example: Nix expression for Hello World



Example: Nix expressions involved in building a release consisting of Nix channel builds and RPMs for various platforms

Status and Research Directions

- Nix-based build farm in use at UU, TUD.
- Used by various open source projects: Stratego/XT, MetaEnvironment, Nix, NixOS...
- Future work: **automatic exploration of the configuration space** – try to select configurations that are more likely to exhibit problems.
- Future work: use static analyses to find potentially troublesome configurations, e.g., interference between `#ifdefs` in a C program

More information

- Nix web site: http://nix.cs.uu.nl/
- Example Nix build farms: http://nix.cs.uu.nl/dist/, http://buildfarm.st.ewi.tudelft.nl/