Automated Software Testing and Release with Nix Build Farms

Eelco Dolstra¹  Eelco Visser²

¹University of Utrecht, Faculty of Science, Department of Information and Computing Sciences

²Delft University of Technology, EWI, Department of Software Technology

March 23, 2007
**Build Farms**

*Build farm*: a set of machines that continuously builds and tests software components from a version management system, producing status reports and/or releases.
Goal: Building

- 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/...-db4-4.4.20/include -I/...-aterm-2.4.2/include -I../libutil -I../libstore -g -02 -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.../...-db4-4.4.20/include -I/...-aterm-2.4.2/include -I../libutil -I../libstore -g -02 -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: 'ATmakeApp13' 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_APPPL' was not declared in this scope`
  `nixexpr-ast.hh:15: error: 'AFun' was not declared in this scope`
Goal: Running test suites

- 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
**Build Farm Results for Package strategoxt**

<table>
<thead>
<tr>
<th>Package</th>
<th>Release</th>
<th>Rev</th>
<th>All</th>
<th>Source tarball</th>
<th>i686-linux</th>
<th>i686-darwin</th>
<th>powerpc-darwin</th>
<th>i686-cygwin</th>
<th>Red Hat 9.0</th>
<th>Fedora Core 2</th>
<th>Fedora Core 3</th>
<th>SuSE 9.0</th>
<th>Check</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>strategoxt</td>
<td>0.17M2pre15838</td>
<td>15830</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>0.17M2pre15837</td>
<td>15837</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>0.17M2pre15836</td>
<td>15836</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>0.17M2pre15835</td>
<td>15835</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>0.17M2pre15831</td>
<td>15831</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>0.17M2pre15819</td>
<td>15819</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>0.17M2pre15809</td>
<td>15809</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>0.17M2pre15799</td>
<td>15799</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>15784-bad</td>
<td>15784</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>0.17M2pre15779</td>
<td>15779</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>0.17M2pre15767</td>
<td>15767</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>0.17M2pre15761</td>
<td>15761</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>0.17M2pre15780</td>
<td>15780</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>15757-bad</td>
<td>15757</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>15755-bad</td>
<td>15755</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>strategoxt</td>
<td>15754-bad</td>
<td>15754</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## Build Farm Results for Package strategyx

**Objective:** Portability testing

- Windows XP, 32 bit
- Windows XP, 64 bit
- Linux, Intel, 32 bit
  - Red Hat
  - SUSE
  - ...
- Linux, Intel, 64 bit
- Linux, PowerPC
- Mac OS X, PowerPC
- Mac OS X, Intel
- Solaris, Sparc
- ...

<table>
<thead>
<tr>
<th>Package</th>
<th>Release</th>
<th>Rev</th>
<th>All</th>
<th>Source</th>
<th>i686-linux</th>
<th>i686-darwin</th>
<th>powerpc</th>
</tr>
</thead>
<tbody>
<tr>
<td>strategyx</td>
<td>0.17M2pre15838</td>
<td>15838</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>strategyx</td>
<td>0.17M2pre15837</td>
<td>15837</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>strategyx</td>
<td>0.17M2pre15836</td>
<td>15836</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>strategyx</td>
<td>0.17M2pre15835</td>
<td>15835</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>strategyx</td>
<td>0.17M2pre15831</td>
<td>15831</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>strategyx</td>
<td>0.17M2pre15819</td>
<td>15819</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>strategyx</td>
<td>0.17M2pre15809</td>
<td>15809</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>strategyx</td>
<td>0.17M2pre15799</td>
<td>15799</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>strategyx</td>
<td>15784-bad</td>
<td>15784</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>strategyx</td>
<td>0.17M2pre15779</td>
<td>15779</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>strategyx</td>
<td>0.17M2pre15767</td>
<td>15767</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>strategyx</td>
<td>0.17M2pre15761</td>
<td>15761</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>strategyx</td>
<td>0.17M2pre15760</td>
<td>15760</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>strategyx</td>
<td>15757-bad</td>
<td>15757</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>strategyx</td>
<td>15755-bad</td>
<td>15755</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>strategyx</td>
<td>15754-bad</td>
<td>15754</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Goal: Portability testing

Making all in nix-setuid-helper
if g++ -DHAVE_CONFIG_H -I. -I../.. -I/.. -I/./. . -I/.. -I/./...-term-2.4.2-fixes/include -I/..../libutil
-D_FILE_OFFSET_BITS=64 -g -02 -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, 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
Goal: Run analysis tools

Static analyses (e.g., Lint, FindBugs) or dynamic analyses (e.g., code coverage, Valgrind).

**LTP GCOV extension - code coverage report**

<table>
<thead>
<tr>
<th>Filename</th>
<th>Coverage</th>
<th>Instrumented lines</th>
<th>Executed lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>attr-path.cc</td>
<td>85.3 %</td>
<td>29 / 34 lines</td>
<td></td>
</tr>
<tr>
<td>eval.cc</td>
<td>90.8 %</td>
<td>356 / 392 lines</td>
<td></td>
</tr>
<tr>
<td>expr-to-xml.cc</td>
<td>92.9 %</td>
<td>52 / 56 lines</td>
<td></td>
</tr>
<tr>
<td>get-drvs.cc</td>
<td>79.4 %</td>
<td>77 / 97 lines</td>
<td></td>
</tr>
<tr>
<td>get-drvs.hh</td>
<td>50.0 %</td>
<td>2 / 4 lines</td>
<td></td>
</tr>
<tr>
<td>lexer.1</td>
<td>96.0 %</td>
<td>72 / 75 lines</td>
<td></td>
</tr>
<tr>
<td>nixexpr-ast.cc</td>
<td>100.0 %</td>
<td>121 / 121 lines</td>
<td></td>
</tr>
<tr>
<td>nixexpr-ast.hh</td>
<td>95.4 %</td>
<td>293 / 307 lines</td>
<td></td>
</tr>
<tr>
<td>nixexpr.cc</td>
<td>78.1 %</td>
<td>168 / 215 lines</td>
<td></td>
</tr>
<tr>
<td>nixexpr.hh</td>
<td>100.0 %</td>
<td>13 / 13 lines</td>
<td></td>
</tr>
<tr>
<td>parser.y</td>
<td>96.6 %</td>
<td>171 / 177 lines</td>
<td></td>
</tr>
<tr>
<td>primops.cc</td>
<td>66.7 %</td>
<td>234 / 351 lines</td>
<td></td>
</tr>
</tbody>
</table>
Build farm goals

Goal: Continuous integration

Some of the packages developed by us and our colleagues

Changes in one package can break another
**Goal: Release management**

If a build succeeds, the result can be made available as an installable package to users.

**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** (10842950 bytes; MD5 hash: 9ce5bb9f87a613803547cece51c1d451)

**RPM for Red Hat 9.0**

- **php-sat-0.1pre286-1.i386.rpm** (145051 bytes; MD5 hash: fcfecda52e39c9e6e548d0ba0647bba)
- **php-sat-0.1pre286-1.src.rpm** (551573 bytes; MD5 hash: f06c9bfc1ac95041e52ab51e7df54a9)

This RPM requires that the following packages are also installed:

- **atrm-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**
Current build farm tools

Examples

- Mozilla Tinderbox
- CruiseControl
- AntHill
- BuildBot
- SourceForge Compile Farm

Central Problem

*How do we manage the build environment?*
Problem: creating the build environment

- A package typically has a lot of build time dependencies that must be distributed to each build machine.
- Dependencies of Stratego/XT that have caused problems in the past:
  - Autoconf
  - Stratego/XT
  - Automake
  - Libtool
  - GCC
  - Sed
  - Make
  - ATerm
  - SDF
- $N$ dependencies, $M$ platforms
  - $\Theta(N \times M)$ effort to keep the build farm up to date.
- And what if there are conflicting dependencies?
Runtime dependencies

Firefox

- firefox-2.0.0.2
- libXext-1.0.2
- gtk+-2.10.9
- libXi-1.0.2
- libXrandr-1.1.2
- xlibs-wrapper
- libXinerama-1.0.1
- atk-1.12.4
- cairo-1.2.6
- pango-1.14.10
- libX11-1.1.1
- libXft-2.1.12
- libXrender-0.9.2
- libXt-1.0.4
- zlib-1.2.3
- libpng-1.2.16
- libtiff-3.8.2
- libjpeg-6b
- libIDL-0.8.7
- glibc-2.5
- fontconfig-2.4.2
- glib-2.12.9
- gcc-4.1.1
- freetype-2.3.1
- libXau-1.0.3
- libXdmcp-1.0.2
- libxcb-1.0
- expat-2.0.0
- libICE-1.0.3
- libSM-1.0.2
- perl-5.8.6
- coreutils-6.7
- xextproto-7.0.2
- xproto-7.0.10
- randrproto-1.1.2
- kbproto-1.0.3
- renderproto-0.9.2
- xineramaproto-1.1.2
- linux-headers-2.6.18.1
- inputproto-1.3.2
The Nix Deployment System

- Deployment system developed at Utrecht University: http://nix.cs.uu.nl/

- *Purely functional* package management: package builds only depend on declared inputs; never change after they have been built.

- Main features:
  - Enforce correct dependency specifications.
  - Support concurrent variants/versions.
  - Safe and automatic garbage collection of unused components.
  - Transparent source/binary deployment model.
  - Atomic upgrades/rollbacks.
  - Simple component language with variability support.
  - Mechanism, not policy; lots of different deployment policies can be defined using basic Nix mechanisms (e.g., channels).
  - Not just for software deployment but also service deployment.
The Nix Deployment System

- Deployment system developed at Utrecht University: http://nix.cs.uu.nl/
- *Purely functional* package management: package builds only depend on declared inputs; never change after they have been built.
- Main features:
  - Enforce correct dependency specifications.
  - Support concurrent variants/versions.
  - Safe and automatic garbage collection of unused components.
  - Transparent source/binary deployment model.
  - Atomic upgrades/rollbacks.
  - Simple component language with variability support.
  - Mechanism, not policy; lots of different deployment policies can be defined using basic Nix mechanisms (e.g., channels).
  - Not just for software deployment but also service deployment.
Central idea: store all components in isolation.

Unique paths:

```
/nix/store/jjp9pirx8b3nqs9k...-firefox
```

which is an 160-bit **cryptographic hash** of all inputs used to build the component:

- Sources
- Libraries
- Compilers
- Build scripts
- Build parameters
- System type
- ...

- **Prevent** undeclared **build time** dependencies.
- **Scan** for **runtime** dependencies.
- Deploy only **closures** under the **depends-on** relation.
Nix store

/nix/store
- 50nddzshprba...-gtk+-2.2.4
  - lib
    - libgtk-x11-2.0.so.0
- 5lmkmbs16z5s......-wxGTK-2.6.2
  - lib
    - libwx_gtk2-2.4.so
- v6ajzxqk84fy...-bittorrent-3.4.2
  - bin
    - btdownloadgui.py
- 4kd0ma2pxf6w...-gtk+-2.8.6
  - lib
    - libgtk-x11-2.0.so.0
- jjp9pirx8b3nqs9k...-firefox
  - bin
    - firefox
  - lib
    - libxpcom.so
    - libmozz.so
    - ...
Nix store

```
/nix/store
```

- `50nddzshprba...-gtk+-2.2.4`
  - `lib`
    - `libgtk-x11-2.0.so.0`

- `5lmkmbs16z5s......-wxGTK-2.6.2`
  - `lib`
    - `libwx_gtk2-2.4.so`

- `v6ajzxqk84fy...-bittorrent-3.4.2`
  - `bin`
    - `btdownloadgui.py`

- `4kd0ma2pxf6w...-gtk+-2.8.6`
  - `lib`
    - `libgtk-x11-2.0.so.0`

- `jjp9pirx8b3nqs9k...-firefox`
  - `bin`
    - `firefox`
  - `lib`
    - `libxpcom.so`
    - `libmozz.so`
    - `...`

Unique paths for different versions
Packages are built using *Nix expressions*:

```nix
{stdenv, fetchurl, perl}:

stdenv.mkDerivation {
  name = "hello-2.1.1";
  builder = ./builder.sh;
  src = fetchurl {
    md5 = "70c9ccf9fac07f762c24f2df2290784d";
  };
  inherit perl;
}
```
Nix expressions

Packages are built using *Nix expressions*:

```nix
{stdenv, fetchurl, perl}:

stdenv.mkDerivation {
  name = "hello-2.1.1";
  builder = ./builder.sh;
  src = fetchurl {
    url =
    md5 = "70c9ccf9fac07f762c24f2df2290784d";
  };
  inherit perl;
}
```

Function arguments
Packages are built using *Nix expressions*:

```nix
{stdenv, fetchurl, perl}:

stdenv.mkDerivation {
  name = "hello-2.1.1";
  builder = ./builder.sh;
  src = fetchurl {
    md5 = "70c9ccf9fac07f762c24f2df2290784d";
  };
  inherit perl;
}
```
source $stdenv/setup

PATH=$perl/bin:$PATH

tar xvfz $src
cd hello-*
./configure --prefix=$out
make
make install
Nix expressions

```nix
system/all-packages.nix

hello = import ../applications/misc/hello/ex-1 {
  inherit fetchurl stdenv perl;
};

perl = import ../development/interpreters/perl {
  inherit fetchurl stdenv;
};

fetchurl = import ../build-support/fetchurl {
  inherit stdenv; ...
};

stdenv = ...;
```
system/all-packages.nix

```nix
hello = import ../applications/misc/hello/ex-1 {
  inherit fetchurl stdenv perl;
};

perl = import ../development/interpreters/perl {
  inherit fetchurl stdenv;
};

fetchurl = import ../build-support/fetchurl {
  inherit stdenv; ...
};

stdenv = ...;
```
bittorrent = import ../tools/networking/bittorrent {
  inherit fetchurl stdenv wxGTK;
};

wxGTK = import ../development/libraries/wxGTK {
  inherit fetchurl stdenv pkgconfig;
  gtk = gtkLibs22.gtk;
};

firefox = import ../applications/browsers/firefox {
  inherit fetchurl stdenv pkgconfig perl zip libIDL libXi;
  gtk = gtkLibs24.gtk;
};
Implementing a Build Farm with Nix

Why is this useful for a build farm?

- The Nix expression language is ideal for describing the build tasks to be performed.
- The Nix expression language makes it easy to describe variant compositions.
- Nix manages the storage of components.
- Nix supports distributed builds in a transparent way.
- 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.
Implementing a Build Farm with Nix

Why is this useful for a build farm?

- The Nix expression language is ideal for describing the build tasks to be performed.
- The Nix expression language makes it easy to describe variant compositions.
- Nix manages the storage of components.
- Nix supports distributed builds in a transparent way.
- 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.
Why is this useful for a build farm?

- The Nix expression language is ideal for describing the build tasks to be performed.
- The Nix expression language makes it easy to describe variant compositions.
- Nix manages the storage of components.
- Nix supports distributed builds in a transparent way.
- 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.
Implementing a Build Farm with Nix

Why is this useful for a build farm?

- The Nix expression language is ideal for describing the build tasks to be performed.
- The Nix expression language makes it easy to describe variant compositions.
- Nix manages the storage of components.
- Nix supports distributed builds in a transparent way.
- 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.
Why is this useful for a build farm?

- The Nix expression language is ideal for describing the build tasks to be performed.
- The Nix expression language makes it easy to describe variant compositions.
- Nix manages the storage of components.
- Nix supports distributed builds in a transparent way.
- 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.
Implementing a Build Farm with Nix

Why is this useful for a build farm?

▶ The Nix expression language is ideal for describing the build tasks to be performed.
▶ The Nix expression language makes it easy to describe variant compositions.
▶ Nix manages the storage of components.
▶ Nix supports distributed builds in a transparent way.
▶ 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.
Building a release

What goes into a release?

- A source distribution.
- Binary distributions for a number of platforms. (Test sets are also run on each platform.)
  - RPM packages for Red Hat 9, Fedora Core, SUSE Linux, ...
  - Windows binaries
  - Nix channel builds for Linux, Mac OS X, Windows, ...
  - ...
- Build logs, analysis results, etc.
Building a source distribution

svnToSourceTarball is a function that checks out sources from a specific revision from a Subversion repository (as specified by input.

# Bring in some standard packages (compilers, etc.)
pkgs = import .../all-packages.nix;
pkgsLinux = pkgs {system = "i686-linux"};

strategoxtTarball = input: svnToSourceTarball input {
    stdenv = pkgsLinux.stdenv;
    buildInputs = [pkgsLinux.autoconf pkgsLinux.automake ...];
};
Performing a Nix channel build for Linux

nixBuild performs a channel build from a source distribution.

strategoxtBinary = input: nixBuild
  (strategoxtTarball input)
{
  stdenv = pkgsLinux.stdenv;
  buildInputs = [pkgsLinux.aterm pkgsLinux.sdf];
};
Building an RPM

umlBuild performs an RPM build from a source distribution in a User-Mode Linux virtual machine.

strategoxtRPM = input: diskImage: umlBuild diskImage (strategoxtTarball input);

redhatDiskImage = fillWithRPMs {
  fetchurl {url = ftp://.../RedHat/basesystem-8.0-2.rpm;}
  fetchurl {url = ftp://.../RedHat/bash-2.05b-20.i386.rpm;}
  fetchurl {url = ftp://.../RedHat/gcc-3.2.2-5.i386.rpm;}
  ...
};

suseDiskImage = fillWithRPMs { ... };

That is, we generate virtual machines on the fly from a specification.
Building a release page

makeReleasePage creates an HTML release page and other files that should be uploaded to a server.

strategoxtRelease = input: makeReleasePage {
  stdenv = pkgsLinux.stdenv;
  sourceTarball = strategoxtTarball input;
  binaries = [(strategoxtBinary input)];
  rpms = [
    (strategoxtRPM input suseDiskImage)
    (strategoxtRPM input redhatDiskImage)
  ];
};
pkgs = (import ../all-packages);

pkgsLinux = pkgs {system = "i686-linux"};

pkgsDarwin = pkgs {system = "powerpc-darwin"};

strategoxtBinary = pkgs: input: nixBuild
  (strategoxtTarball input)
{
  stdenv = pkgs.stdenv;
  buildInputs = [pkgs.aterm pkgs.sdf];
};

strategoxtBinaries = input: [
  (strategoxtBinary pkgsLinux input)
  (strategoxtBinary pkgsDarwin input)
];
Conclusion

The Nix build farm:

- Manages the complexity of the build environment.
- Has a functional component language that makes it easy to specify the configurations to build/test.
- Ensures reproducibility.
- Supports multi-platform builds.
- Is efficient: only changed subexpressions are rebuilt.
- Produces actual releases.

More information

http://nix.cs.uu.nl/

Example Nix build farms:

- http://nix.cs.uu.nl/dist/
- http://buildfarm.st.ewi.tudelft.nl/