TraCE: Transparent Configuration Environments

Martin Bravenboer Eelco Dolstra Eelco Visser

Institute of Information & Computing Sciences
Utrecht University, The Netherlands

February 4, 2005

Project Overview

Project goal

Improving the configuration of software systems

Team

- Eelco Visser (principal investigator)
- Martin Bravenboer (PhD student)
- Eelco Dolstra (PhD student)
- Gert Florijn (CIBIT)
- Doaitse Swierstra (promotor)
- Merijn de Jonge (was postdoc, now at Philips Research)

Variability in Software Systems

Variability is the ability to select the set of features for a particular instance of a software system.

Configuration of variability is realized through configuration mechanisms.

Dimensions of Software Configuration

Type of variability

- functionality
- version
- platform
- dependencies

Unit of configuration

- file
- package
- system

Moment of configuration

- development
- build
- distribution
- installation
- activation
- run

Different mechanisms at each point in configuration space

Dimensions of Software Configuration

Type of variability

- functionality
- version
- platform
- dependencies

Unit of configuration

- file
- package
- system

Moment of configuration

- development
- build
- distribution
- installation
- activation
- run

Different mechanisms at each point in configuration space

Dimensions of Software Configuration

Type of variability

- functionality
- version
- platform
- dependencies

Unit of configuration

- file
- package
- system

Moment of configuration

- development
- build
- distribution
- installation
- activation
- run

Different mechanisms at each point in configuration space

Dimensions of Software Configuration

Type of variability

- functionality
- version
- platform
- dependencies

Unit of configuration

- file
- package
- system

Moment of configuration

- development
- build
- distribution
- installation
- activation
- run

Different mechanisms at each point in configuration space

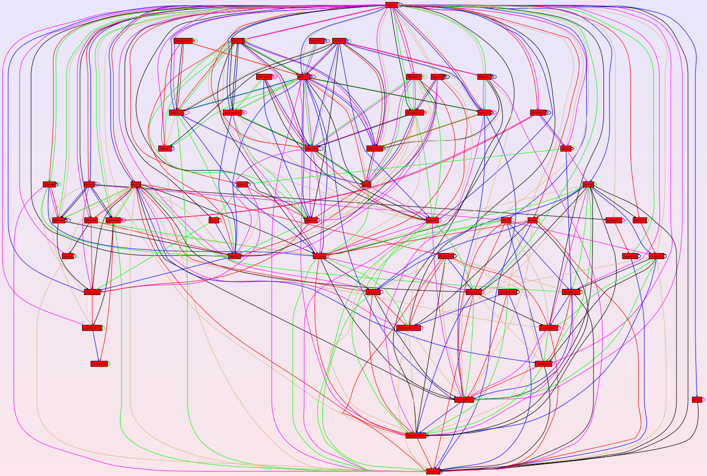
Example Problem: Conditional Compilation

```
#ifdef (HP9000_S800) /* If HP9000_S800 is defined, INT_SIZE */
#define INT_SIZE 32 /* is defined to be 32 (bits). */
#elif defined (HPVECTRA) && defined (SMALL_MODEL)
#define INT_SIZE 16 /* Otherwise, if HPVECTRA and */
#endif /* SMALL_MODEL are defined,INT_SIZE is */

#ifdef DEBUG /* If DEBUG is defined, display the */
printf("table element : \n"); /* table elements. */
for (i=0; i < MAX_TABLE_SIZE; ++i)
printf("%d %f\n", i, table[i]);
#endif
```

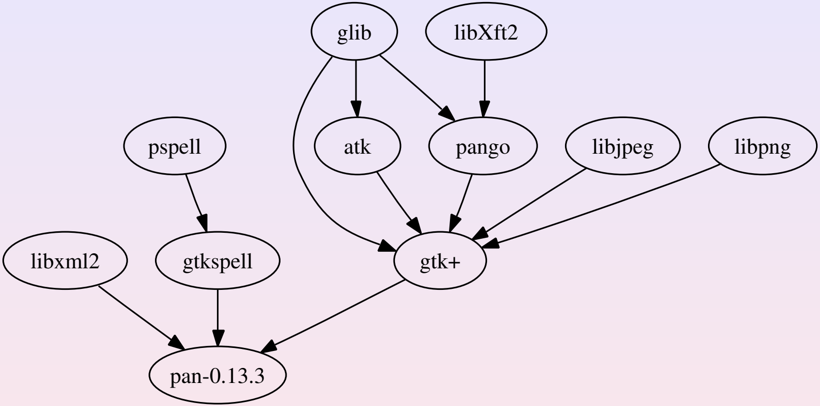
configuration with unhygienic lexical program transformations

Example Problem: Package Dependencies



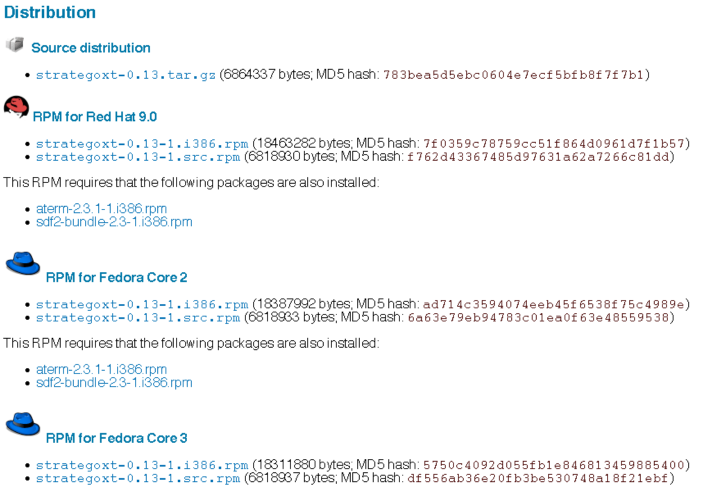
incomplete dependencies lead to unsafe deployment

Example Problem: Package Dependencies



incomplete dependencies lead to unsafe deployment

Example Problem: Release Management



Distribution

Source distribution

- strategoxt-0.13.tar.gz (6864337 bytes; MD5 hash: 783bea5d5ebc0604e7ecf5bfb8f7f7b1)

RPM for Red Hat 9.0

- strategoxt-0.13-1.i386.rpm (18463282 bytes; MD5 hash: 7f0359c78759cc51f864d0961d7f1b57)
- strategoxt-0.13-1.src.rpm (6818930 bytes; MD5 hash: f762d43367485d97631a62a7266c81dd)

This RPM requires that the following packages are also installed:

- aterm-2.3.1-1.i386.rpm
- sdf2-bundle-2.3-1.i386.rpm

RPM for Fedora Core 2

- strategoxt-0.13-1.i386.rpm (18387992 bytes; MD5 hash: ad714c3594074eeb45f6538f75c4989e)
- strategoxt-0.13-1.src.rpm (6818933 bytes; MD5 hash: 6a63e79eb94783c01ea0f63e48559538)

This RPM requires that the following packages are also installed:

- aterm-2.3.1-1.i386.rpm
- sdf2-bundle-2.3-1.i386.rpm

RPM for Fedora Core 3

- strategoxt-0.13-1.i386.rpm (18311880 bytes; MD5 hash: 5750c4092d055fb1e846813459885400)
- strategoxt-0.13-1.src.rpm (6818937 bytes; MD5 hash: df556ab36e20fb3be530748a18f21ebf)

manually releasing many packages often does not scale

TraCE Goals

Transparent configuration of software systems

at all levels of granularity and

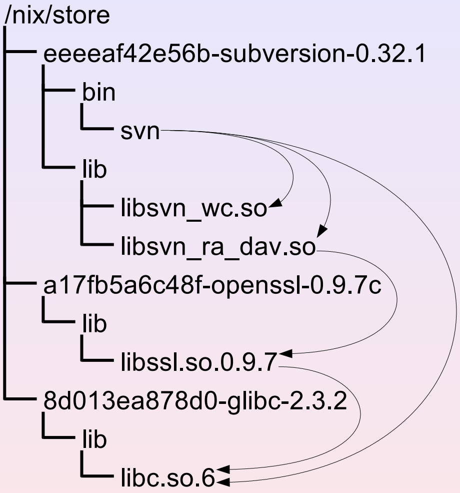
at all moments on the development/deployment timeline.

Example Solution: User-Interface Configuration

```
public void createLayout(){
  this {
    content = panel {
      border = line border borggreen
      layout = border layout {
        center = label {
          text = "Welcome"
          border = raised etched border
          doublebuffered = true
        }
        south = new JButton("Ok")
      }}};
  this.pack();
  this.setVisible(true);
}
```

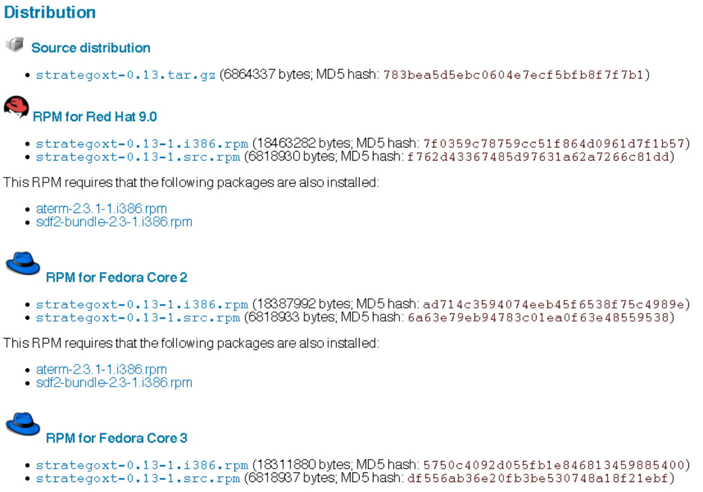
high-level domain-specific configuration supported by hygienic transformations on code structure

Example Solution: Dependency Closures



Safe deployment by computing complete closure of dependency relation

Example Solution: Automatic Release Management



Distribution

Source distribution

- strategoxt-0.13.tar.gz (6864337 bytes; MD5 hash: 783bea5d5ebc0604e7ecf5bfb8f7f7b1)

RPM for Red Hat 9.0

- strategoxt-0.13-1.i386.rpm (18463282 bytes; MD5 hash: 7f0359c78759cc51f864d0961d7f1b57)
- strategoxt-0.13-1.src.rpm (6818930 bytes; MD5 hash: f762d43367485d97631a62a7266c81dd)

This RPM requires that the following packages are also installed:

- aterm-2.3.1-1.i386.rpm
- sdf2-bundle-2.3-1.i386.rpm

RPM for Fedora Core 2

- strategoxt-0.13-1.i386.rpm (18387992 bytes; MD5 hash: ad714c3594074eeb45f6538f75c4989e)
- strategoxt-0.13-1.src.rpm (6818933 bytes; MD5 hash: 6a63e79eb94783c01ea0f63e48559538)

This RPM requires that the following packages are also installed:

- aterm-2.3.1-1.i386.rpm
- sdf2-bundle-2.3-1.i386.rpm

RPM for Fedora Core 3

- strategoxt-0.13-1.i386.rpm (18311880 bytes; MD5 hash: 5750c4092d055fb1e846813459885400)
- strategoxt-0.13-1.src.rpm (6818937 bytes; MD5 hash: df556ab36e20fb3be530748a18f21ebf)

completely automatic creation of source and binary distributions

Results

Software deployment (Nix)

SCM'03	Integrating Software Construction and Software Deployment
ICSE'04	Imposing a Memory Management Discipline on Software Deployment
LISA'04	Nix: A Safe and Policy-Free System for Software Deployment.

Build-level composition (AutoBundle/Koala)

ICSR'04	Decoupling Source Trees into Build-Level Components

Code-Level configuration (Stratego/XT)

OOPSLA'04	Concrete Syntax for Objects

Dynamic component composition (XTC)

Thesis'05	Transformation Tool Composition

What do we have to offer?

Ready for showtime

The *Nix Deployment System*

- ... nice features ...
- web service configuration and deployment
- continuous integration
- automatic release management

Under research

- Source-level configuration
- Configuration of compositions

TraCE Workshop

- Overview of Nix
- Opportunities for further research and collaboration

Part II

TraCE Workshop

TraCE workshop

- Overview of TraCE research topics (past, present and future) with interactive discussion.
  - Nix
  - Code-level configuration
- Questions:
  - Relevance for industry
  - Other research questions?
  - ...

Project Overview

Project goal

Improving the configuration of software systems

Team

- Eelco Visser (principal investigator)
- Martin Bravenboer (PhD student)
- Eelco Dolstra (PhD student)
- Gert Florijn (CIBIT)
- Doaitse Swierstra (promotor)
- Merijn de Jonge (was postdoc, now at Philips Research)

Results

Software deployment (Nix)

- SCM'03 Integrating Software Construction and Software Deployment
- ICSE'04 Imposing a Memory Management Discipline on Software Deployment
- LISA'04 Nix: A Safe and Policy-Free System for Software Deployment.

Build-level composition (AutoBundle/Koala)

- ICSR'04 Decoupling Source Trees into Build-Level Components

Code-Level configuration (Stratego/XT)

- OOPSLA'04 Concrete Syntax for Objects

Dynamic component composition (XTC)

- Thesis'05 Transformation Tool Composition

Source-level configuraton

Problems

Poor and different techniques, e.g.

- Lexical pre-processing
- Conditional includes
- Source file selection
- Code generation
- Flexible, but difficult to use and select

Goals

- Unify configuration mechanisms in source code
- Improve mapping of configuration from environment to source code

Source-level configuration (2)

Solutions

- Abstract over concrete configuration mechanisms
  - Generate concrete use
  - Select concrete mechanism
- Separation of configuration issues
  - Domain-specific languages and generators
  - Avoid mixing real code and configuration issues: separate meta-level interfaces

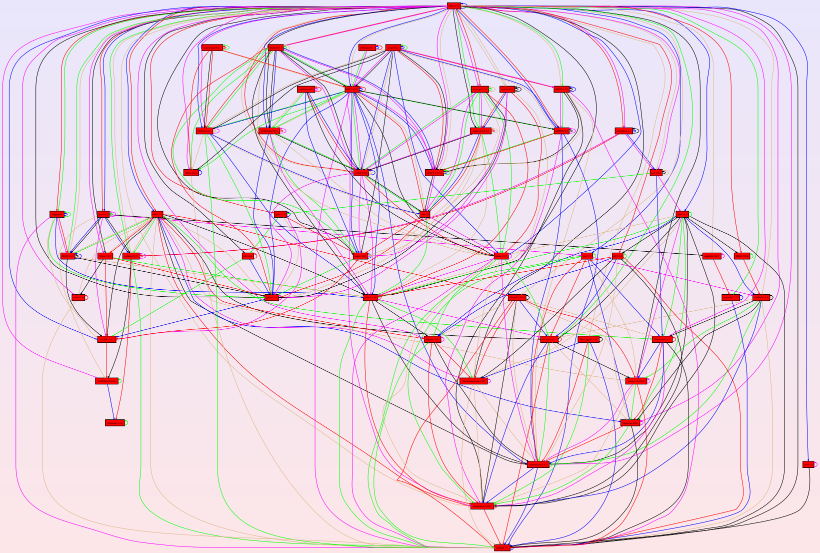
Techniques

- Program manipulation and generation
- Domain-specific language design, implementation, and embedding

The Nix Deployment System

- Software deployment: the art of **transferring software** (packages) from one machine to another (and managing it).
- The hard part: packages should **work the same** on the target machine.
  - "DLL hell"
  - "Dependency hell"
  - Labour-intensive

So why is this hard?



Nix

- Central idea: store all packages in isolation.
- Unique paths:

```
/nix/store/605332199533e73b...-gtk+-2.2.4
```

which is an MD5 hash of **all** inputs used to build the package:

  - 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.

Advantages

- Safe deployment
- Effective composition language
- User environments
- Less packaging effort
- Build farms / automated release management
- Efficient upgrade deployment
- Policy-freeness
- Support for service deployment

Safe deployment

- Hashing gives variability support for free, so no overwriting of dependencies
- Full dependency graph
- Complete deployment
- Side-by-side versioning
- Automatic garbage collection of unused components

Composition language

- *Nix expressions*: simple functional language to describe components and compositions; easy to express variants

```
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;
};
```

User environments

- *User environments* are components of symlinks to activated components
- Can be per-user/process/service/...
- Allow atomic upgrading / downgrading (rollbacks)

Less packaging effort

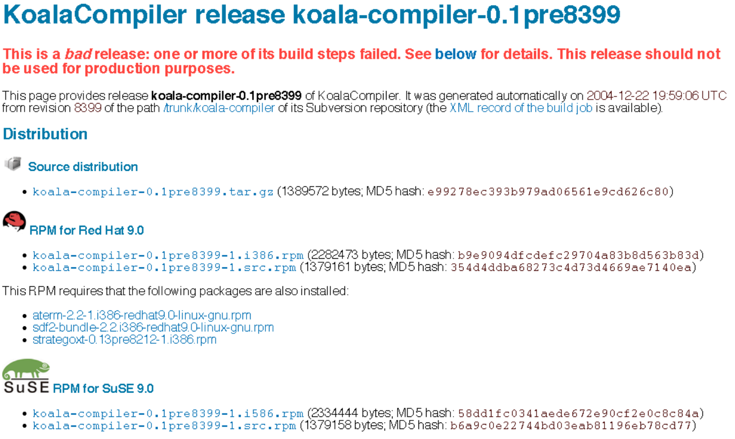
- Nix expressions describe a source deployment model
- No need to explicitly to do binary packaging due to *transparent binary deployment*: binaries can be *cached* in a shared repository

Build farms

- Nix is a good basis for a build farm because:
  - Expression language ideal for describing build tasks
  - Expression language makes it easy to describe variant compositions
  - Nix manages the dependencies
  - Supports for distributed builds
  - Hashing scheme + complete dependencies allow builds to be reproduced reliably
  - Efficiency: due only rebuild things that have changed
- Useful for:
  - Continuous integration testing
  - Portability testing
  - *Automated release management* — successful builds are releases

Release management

- Successful builds are automatically *released* as stable or unstable releases.
- Releases can be automatically *pushed to* / *pulled by* clients.



KoalaCompiler release koala-compiler-0.1pre8399

This is a *bad* release: one or more of its build steps failed. See below for details. This release should not be used for production purposes.

This page provides release **koala-compiler-0.1pre8399** of KoalaCompiler. It was generated automatically on 2004-12-22 19:59:06 UTC from revision 8399 of the path /trunk/koala-compiler of its Subversion repository (the XML record of the build job is available).

Distribution

Source distribution

- `koala-compiler-0.1pre8399.tar.gz` (1389572 bytes; MD5 hash: `e99278ec393b979ad06561e9cd626c80`)

RPM for Red Hat 9.0

- `koala-compiler-0.1pre8399-1.i386.rpm` (2282473 bytes; MD5 hash: `b9e9094dfcdefc29704a83b8d563b83d`)
- `koala-compiler-0.1pre8399-1.src.rpm` (1379161 bytes; MD5 hash: `354d4ddba68273c4d73d4669ae7140ea`)

This RPM requires that the following packages are also installed:

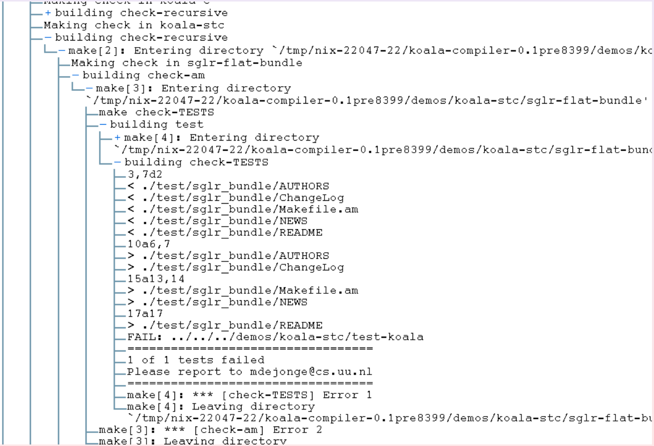
- aterm-2.2-1.i386-redhat9.0-linux-gnu.rpm
- sdf2-bundle-2.2.i386-redhat9.0-linux-gnu.rpm
- strategoxt-0.13pre8212-1.i386.rpm

SuSE RPM for SuSE 9.0

- `koala-compiler-0.1pre8399-1.i586.rpm` (2334444 bytes; MD5 hash: `58dd1fc0341aede672e90cf2e0c8c84a`)
- `koala-compiler-0.1pre8399-1.src.rpm` (1379158 bytes; MD5 hash: `b6a9c0e22744bd03eab81196eb78cd77`)

Release management

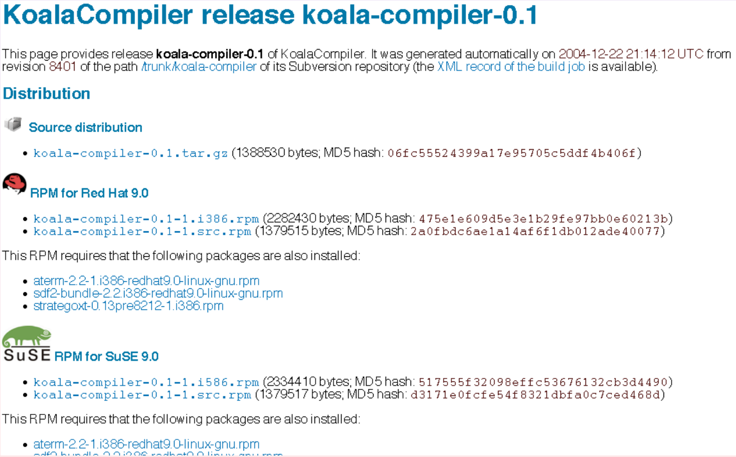
- Successful builds are automatically *released* as stable or unstable releases.
- Releases can be automatically *pushed to* / *pulled by* clients.



```
Making check in koala-c
+ building check-recursive
Making check in koala-stc
- building check-recursive
  - make[2]: Entering directory `/tmp/nix-22047-22/koala-compiler-0.1pre8399/demos/ko
    Making check in sglr-flat-bundle
    - building check-am
      - make[3]: Entering directory
        `/tmp/nix-22047-22/koala-compiler-0.1pre8399/demos/koala-stc/sglr-flat-bundle'
        make check-TESTS
        - building test
          + make[4]: Entering directory
            `/tmp/nix-22047-22/koala-compiler-0.1pre8399/demos/koala-stc/sglr-flat-bund
          - building check-TESTS
            3,7d2
            < ./test/sglr_bundle/AUTHORS
            < ./test/sglr_bundle/ChangeLog
            < ./test/sglr_bundle/Makefile.am
            < ./test/sglr_bundle/NEWS
            < ./test/sglr_bundle/README
            10a6,7
            > ./test/sglr_bundle/AUTHORS
            > ./test/sglr_bundle/ChangeLog
            15a13,14
            > ./test/sglr_bundle/Makefile.am
            > ./test/sglr_bundle/NEWS
            17a17
            > ./test/sglr_bundle/README
            FAIL: ../../../demos/koala-stc/test-koala
            ==================================
            1 of 1 tests failed
            Please report to mdejonge@cs.uu.nl
            ==================================
            make[4]: *** [check-TESTS] Error 1
            make[4]: Leaving directory
            `/tmp/nix-22047-22/koala-compiler-0.1pre8399/demos/koala-stc/sglr-flat-bu
        make[3]: *** [check-am] Error 2
        make[3]: Leaving directory
```

Release management

- Successful builds are automatically *released* as stable or unstable releases.
- Releases can be automatically *pushed to* / *pulled by* clients.



KoalaCompiler release koala-compiler-0.1

This page provides release **koala-compiler-0.1** of KoalaCompiler. It was generated automatically on 2004-12-22 21:14:12 UTC from revision 8401 of the path /trunk/koala-compiler of its Subversion repository (the XML record of the build job is available).

Distribution

Source distribution

- koala-compiler-0.1.tar.gz (1388530 bytes; MD5 hash: 06fc55524399a17e95705c5ddf4b406f)

RPM for Red Hat 9.0

- koala-compiler-0.1-1.i386.rpm (2282430 bytes; MD5 hash: 475e1e609d5e3e1b29fe97bb0e60213b)
- koala-compiler-0.1-1.src.rpm (1379515 bytes; MD5 hash: 2a0fbdc6ae1a14af6f1db012ade40077)

This RPM requires that the following packages are also installed:

- aterm-2.2-1.i386-redhat9.0-linux-gnu.rpm
- sdf2-bundle-2.2.i386-redhat9.0-linux-gnu.rpm
- strategoxt-0.13pre8212-1.i386.rpm

RPM for SuSE 9.0

- koala-compiler-0.1-1.i586.rpm (2334410 bytes; MD5 hash: 517555f32098effc53676132cb3d4490)
- koala-compiler-0.1-1.src.rpm (1379517 bytes; MD5 hash: d3171e0fcfe54f8321dbfa0c7ced468d)

This RPM requires that the following packages are also installed:

- aterm-2.2-1.i386-redhat9.0-linux-gnu.rpm
- sdf2-bundle-2.2.i386-redhat9.0-linux-gnu.rpm

Efficient upgrade (patch) deployment

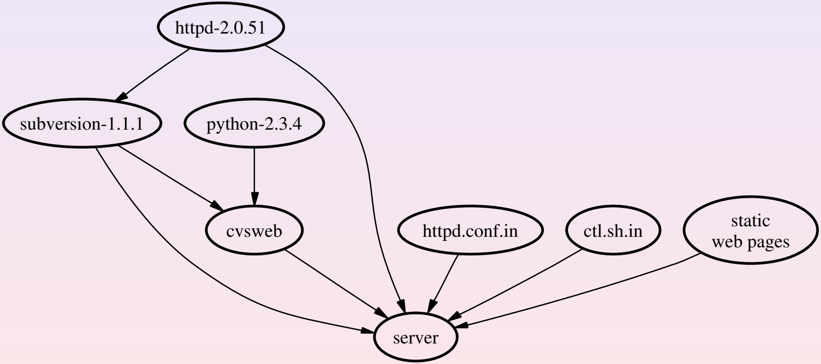
- New versions of components can be deployed as *patches* from older versions
- Completely transparent to users; downloader selects shortest sequence of full or patch downloads
- Also transparent to packagers; integrated into release management

Policy-freeness

- Easy to define new deployment policies, e.g.,
  - Push/pull models
  - Channels
  - Manual
  - Multi-level
  - Whether to do source-only, binary-only, source/binary deployment
  - What variants to pre-build
  - ...

Service deployment

- Deploying a service is (almost) the same as deploying software.
- Example: Subversion server at **svn.cs.uu.nl**.



Status

- Stable implementation.
- Open source.
- Available at `http://www.cs.uu.nl/groups/ST/Trace/Nix`
- Also: Nix Packages collection; large set of common Linux components
- Documentation:
  - Manual
  - Several papers (ICSE-2004, LISA-2004, submitted paper on patch deployment)

Future work

- Security aspects
    - Multi-user stores
    - Ensuring security fix deployment
- "Low-level" build management
- Deployment for distributed systems; mass deployment (incl. grids)
- Configuration selection; e.g., automatically finding configurations meeting constraints