EiffelStudio Internals

Emmanuel Stapf
Eiffel Software
April 12th 2006

Overview of EiffelStudio

- EiffelStudio by numbers
- General overview
- Compiler
  - Validation
  - Code generation
- User interface – EiffelStudio
- Repository
- Runtime
- Projects
- Q&A

A few numbers

- Command-line compiler only:
  - 2100 classes (460 for libraries)
  - 440 000 lines of code (120 000 for libraries)
- Full graphical IDE:
  - 4200 classes (1100 for libraries)
  - 980 000 lines of code (280 000 for libraries)
- C code:
  - 100 000 lines of code

Overview 1 - EiffelStudio's architecture

- EiffelStudio is made of four parts:
  - Core libraries (EiffelBase, EiffelVision2, Gobo…)
  - Core compiler
  - Command line interface
  - Graphical interface
- The graphical IDE contains the command line compiler.
- Command-line compiler can be compiled stand-alone.

Overview 2 - Compilation process

- More at http://eiffelsoftware.origo.ethz.ch/index.php/Eiffel_Compilation_Explained
- Degree 6: finding classes
- Degree 5: parsing classes
- Degree 4: inheritance analysis
- Degree 3: type checking

Overview 2 – Compilation process (2)

- Degree 2/1: melting
- Degree -1: freezing
- Degree -2,-3: finalization
  - Degree -2: process polymorphism
  - DCR: Dead Code Removal
  - Degree -3: code generation

Compiler – AST

- All classes representing *AST* nodes are descendants of *AST_EIFFEL* and have the *_AS* suffix.
- Parser written using gelex/geyacc.
- Parser has many faces:
  - Syntax checker: no AST, useful for syntax validation.
  - Light parser: keeps only nodes needed for validation.
  - Full parser (aka roundtrip parser): preserves all information about Eiffel text (code, blanks and comments).

Compiler – Classes

- Every class has an associated *CLASS_I* instance.
- *CLASS_I* stores information about the file holding the class text: modification date, class name, associated cluster.
- Classes that are part of the system also have an associated *CLASS_C* instance.
- *CLASS_C* stores relations between classes as well as its features.

Compiler – Types

- All types appearing in an AST are transformed into instances of *TYPE_A*.
- *TYPE_A* descendants:
  - *CL_TYPE_A*
  - *GEN_TYPE_A*
  - *TUPLE_TYPE_A*
  - *LIKE_FEATURE*
  - *FORMAL_A*
  - …

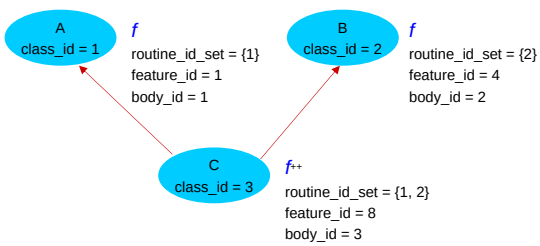
Compiler – Features

- The features of a class are stored in *CLASS_C* into an instance of *FEATURE_TABLE*.
- A *FEATURE_TABLE* is a container of *FEATURE_I*, indexed by feature names and, for fast lookup, by “routine IDs”.
- Descendants of *FEATURE_I*:
  - *PROCEDURE_I*
  - *DYN_FUN_I*
  - *ATTRIBUTE_I*
  - *EXTERNAL_I*
  - …

Compiler – IDs

- Class ID: identifier given to each class.
- Routine ID: identifier given to each feature globally for polymorphism
- Feature ID: identifier given to each feature within a class
- Body ID (aka Body Index): identifier given to a feature text

Compiler – IDS



Compiler – Code Generation

- At degree 3 each feature is transformed into a *BYTE_CODE* instance, a tree of *BYTE_NODE*s.
- Different types of code generation:
  - Melting
  - Freezing
  - Finalization
  - .NET freezing
  - .NET finalizing
  - Java freezing
  - Java finalizing

Dynamic dispatch

- Based on routine IDs
- Each routine ID is associated with a virtual table indexed by the dynamic type of an object at runtime.
- Generated code looks like:

  a.f (args) ⇔ routine [dynamic_type (a)] (args)

EiffelStudio – Editor

- Designed as a library.
- Configured by EiffelStudio to add:
  - Code completion
  - Pick and drop
  - Syntax highlighting
- Used for displaying code, but also results of formatters (views: flat, contract, interface…)
- *TEXT_PANEL* is the ancestor to all editors

EiffelStudio – Context tools

- Controlled by *EB_CONTEXT_TOOL*
- Information outputs:
  - Compilation global process, system information
  - Errors
  - Warnings
  - C compiler output
- Executing commands from EiffelStudio: svn status, svn update, svn commit...

EiffelStudio – Diagram tool

- Uses graph library as data structure for internal representation:
  - Inherits from *EG_NODE*
  - Supports “physics” (force directed layout)
- Drawing done using model cluster of EiffelVision2 (*EV_MODEL_WORLD*)
- Two models are supported:
  - BON (*BON_CLASS_DIAGRAM*)
  - UML (UML subset, *UML_CLASS_DIAGRAM*)

EiffelStudio - Queries

- Unification of classes/features/metrics facilities through a query language
- Grammar not fully specified yet
- What we have in mind: something like

```
select classes
from cluster=base
where count(features) > 10
```

- Work still in progress

EiffelStudio – Navigation

- New search facility (*EB_SEARCH_TOOL*):
  - Multiple scope: class, cluster, multiple clusters, system
  - Regular expression support
  - Search bar add-on to all editors
- Clusters and classes: *EB_CLUSTER_TOOL* and *EB_CLASSES_TREE*
- Features tree: *EB_FEATURES_TOOL* and *EB_FEATURES_TREE*

EiffelStudio – Navigation (2)

- Pebbles used for Pick and Drop are descendants of *STONE*: *CLASSI_STONE*, *CLASSC_STONE*, …
- Communication between all graphical elements is done through a stone (instance of *STONE*)

EiffelStudio – Navigation (3)

- *STONE* descendants:
  - *CLASSI_STONE*: non-compiled class
  - *CLASSC_STONE*: compiled class
  - *CLUSTER_STONE*: cluster/group/library/assembly
  - *FEATURE_STONE*: feature in context of a class
  - *ERROR_STONE*: compilation error
  - *OBJECT_STONE*: object in debugger
  - …

EiffelStudio – Navigation (4)

- Locate a class or feature through an instance of *EB_ADDRESS_MANAGER*
- Used under two forms:
  - As toolbar
  - As modal dialog from context tool
- But same semantics

EiffelStudio – Main window

- *EB_DEVELOPMENT_WINDOW*
  - Top level window in EiffelStudio
  - Handles all tools (clusters, features, context tool, editor, search,…) and their layout
  - Handles tool synchronization through stones
  - Handles creation of menus and commands
  - Two state: developing or debugging

Repository

- Under trunk you have:
  - Delivery:
    - Files used to build a complete installation of EiffelStudio
    - Scripts to build a complete installation of EiffelStudio
  - Src
  - free_add_ons: contributions made outside EiffelSoftware used by or distributed with EiffelStudio

Repository (2)

- Under Src:
  - C_library: libpng, zlib
  - bench: EiffelStudio source code and runtime
  - build2: EiffelBuild source code
  - com_wizard: EiffelCOM Wizard source code
  - common: parsers and AST classes
  - dotnet: .NET specific tools for importing .NET assemblies
  - examples: examples included in EiffelStudio delivery
  - help: source code of wizards for project creation
  - library:
  - library.net:
  - tools: various tool useful for developing

Documentation

- Source code for building doc_builder is at trunk/Src/tools/doc_builder
- Documentation is written in XML and then converted to HTML using doc_builder
- For more details read:

http://eiffelsoftware.origo.ethz.ch/index.php/Documentation

Runtime

- Handles:
  - Memory management and garbage collection
  - Equality and copy
  - Generic conformance
  - Object traversal
  - Debugging facilities for EiffelStudio
  - Threading

Runtime binaries

- Runtime: C/run-time/lib[mt][ebench|wkbench|finalized].[a|so]
- Estudio: C/ipc/deamon/estudio
- Helper for incremental objects storing in compiler: C/compiler/lib[mt][w]compiler.a
- Helper for debugging: C/ipc/ewb/lib[mt][w]ewb.a
- Helper for launching C compilation: C/platform/libplatform.a

Contributions

- Best contributions will be integrated to EiffelStudio
- What are “best” contributions?
  - Useful for all/most Eiffel programmers
  - Working
  - Clean
  - Documented
  - Elegant design
  - Contracted
- Prize for TEETH 2006 (Top EiffelStudio ETH contribution)!

Already in the works for 5.7

- Tabbed editor
- Fully customizable layout
- New interface for editing project configurations
- Query language
- Contextual menus instead of pick and drop

Potential good projects

- Code completion:
  - Add stub routines for inherited deferred routines
  - Add preconditions to a routine by analyzing preconditions of routines used
  - Add predefined code snipet
- Add new type of refactoring
- New wizards to create classes (e.g. if it is a Vision2 window, then add vision2 library automatically to project configuration)

More potential good projects!

- Tooltip in editor for both showing routines contract and attribute/local/argument value when debugging
- Redo error and warning reporting
- Detect syntax and semantics errors while typing
- Auto-correction facilities
- Integrate EiffelBuild into EiffelStudio

More!

- See Wiki:
  http://eiffelsoftware.origo.ethz.ch/index.php/Category:Projects

Useful links

- http://www.eiffel.com
- http://docs.eiffel.com
- http://eiffelsoftware.origo.ethz.ch
- https://eiffelsoftware.origo.ethz.ch/svn/es

Q&A

Any questions?

Thanks and happy Eiffeling!