Fixing DBC

Aaron Bingham
bingham@cenix-bioscience.com

What is DbC?

Big Idea:

Formally document client and provider responsibilities, and have the system automatically check the documentation against the implementation.

Principles

- Contracts are part of the system documentation
- Contracts are written as logical assertions about program behavior
- Contracts are verified automatically (usually at runtime)

Why DbC?

- Documentation quality
  - precision and accuracy
- Implementation quality
  - DbC complements testing
  - Simpler code through clear responsibilities
  - Fewer defensive programming checks reduces code size and error rate
- Supports design and design-implementation transition

Implementation comparison

Feature	Eiffel	Contract	Aspects	Plösch	IPDBC	PyDBC
OLD	yes (3)	yes (1)	yes (2)	yes (2)	no	no
Return values	yes	yes	yes	?	yes	no (4)
Parameters in postcondition	yes	yes	yes	yes	yes	no (4)
Precondition weakening	yes	yes	yes	?	no	no
Violations raise exceptions	yes	yes	yes	no (7)	yes	yes
Contracts visible in docs	yes	yes	yes	yes	no	no
Private assertions hidden in docs	yes	no	no	no	n/a	n/a
Named assertions	yes	no	no	no	no	no
Private attribute names	n/a	no (4)	no (4)	?	yes	yes
Module and function contracts	n/a (5)	yes	yes	no	no	no
Integrated type checking	n/a (6)	no	no	yes	no	no

(1) Shallow copies of explicitly listed values
(2) Deep copies
(3) Copy depth depends on storage declarations
(4) Support could be added relatively easily
(5) Every function and every variable in Eiffel must be part of some class
(6) Eiffel is statically typed
(7) Violations are logged to a file

Conclusions: what we still need

- Contract and Aspects come close
- Both have similar bugs in inheritance semantics
- Both need support for transparent private-attribute name-mangling
- Aspects needs to at least indicate the line number of the failing assertion
- Documentation tools: hide assertions involving private and protected attributes
- A tool to control contract checking at package, module, class, and method level without editing affected module
- We're not far off!

If you're interested in getting involved:
bingham@cenix-bioscience.com