

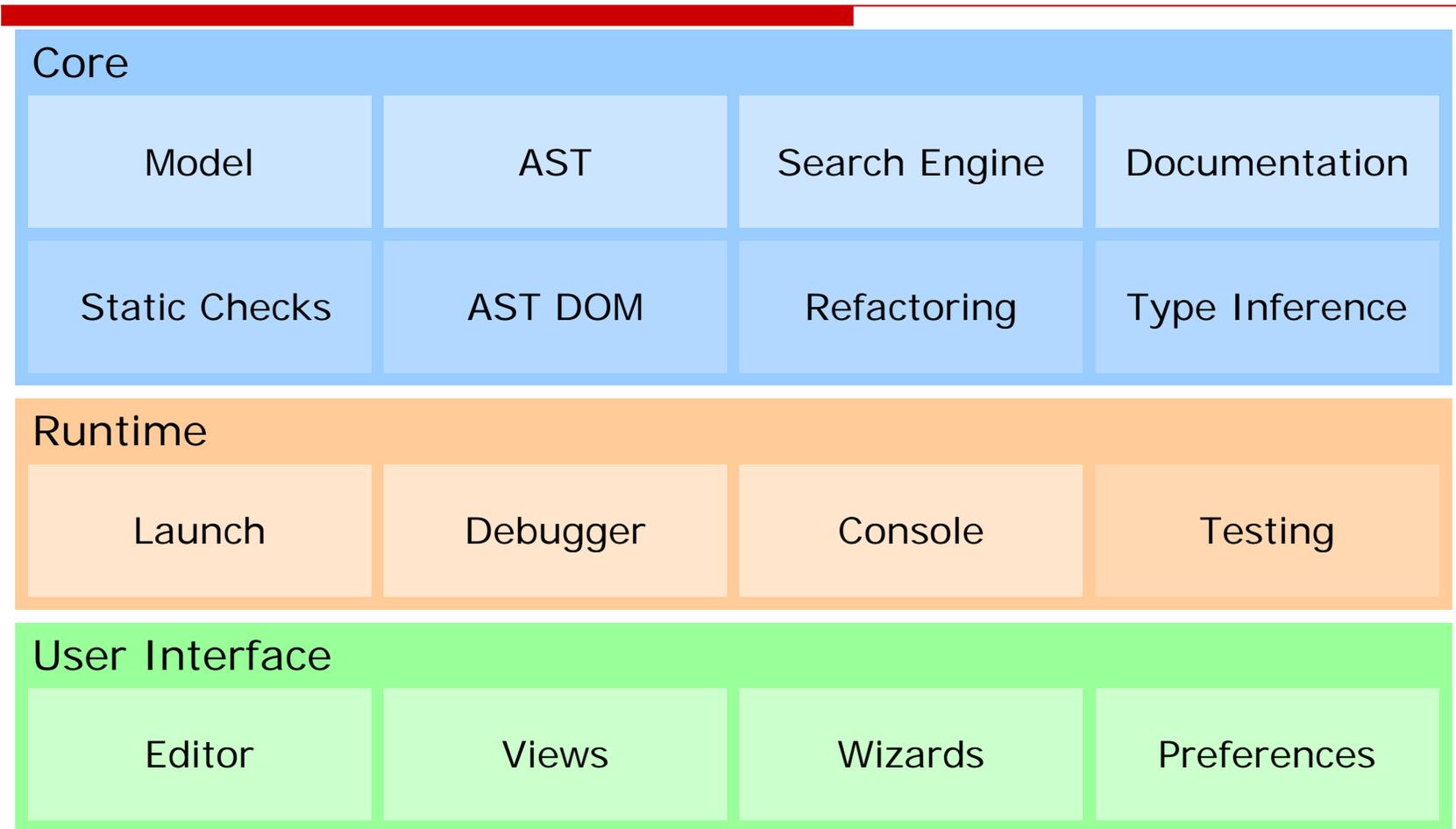
Dynamic Languages Toolkit

Presented by **Andrey Tarantsov**

Dynamic Languages Toolkit

- ❑ Serves as a foundation for scripting and dynamically-typed language IDEs
- ❑ Generalizes JDT code and follows its architecture
- ❑ Provides exemplary Python, Ruby and TCL IDEs

What's Inside DLTK?



Structural Project Model

- Structural elements:
 - Projects
 - Code folders and code libraries
 - Source modules
 - Types, Functions, Variables
 - Package and import declarations
- Environment configuration

Abstract Syntax Tree

- Extensible with language-specific node types and attributes
- Used by completion, search and selection engines

Type Inference

- Under research
- Attempt to implement base algorithm in language-independent fashion
- Used to improve:
 - code analysis
 - code assistance
 - refactoring
 - code verification and error detection

Code Analysis

- Based on Core DLTk infrastructure
- Research to provide common code analysis in a language-independent way

Launch and Debug

- ❑ Environment configuration based on different interpreter types and their installations
- ❑ Full featured Eclipse-based debugger for all scripting languages provided
- ❑ Remote debugging and launching
- ❑ Debug based on open DBGp protocol (xdebug.org)

Interactive Console

- ❑ Common console protocol
- ❑ Remote console
- ❑ Code completion & assistance
- ❑ All standard features

Editor

- Easily extensible with:
 - Folding
 - Indenting
 - Formatting
 - Code assistance / Selection
 - Highlighting / Advanced highlighting

Views

- ❑ Script Explorer
- ❑ Outline
- ❑ Call Hierarchies
- ❑ Navigation (packages, types, functions)
- ❑ Type Hierarchy
- ❑ Quick Outline / Quick Hierarchy

More UI

- ❑ Project properties
- ❑ Wizards
- ❑ Preference pages
- ❑ Search UI

DLTK Extensibility

Core	Model —Structure parser	AST —Source parser	Search Engine —Match locator —Match parser —Index requestor	Documentation —Documentation providers
	Static Checks	AST DOM	Refactoring	Type Inference
Runtime	Launch —Installation types —Environment configuration	Debugger —Interpreter-side implementation	Console —Interpreter-side implementation	Testing —Test framework support
	Editor —Highlighting —Folding —Indenting —Code assist	Views	Wizards	Preferences —Configuration
UI				

Codebase Statistics

	TruStudio	DLTK (current state)
Common	3.8 Mb	3.4 Mb
PHP	0.9 Mb	—
Python	0.8 Mb	0.3 Mb
TCL	—	0.3 Mb
Ratio	4:1	10:1

Which Languages is DLTK for?

Perfect Fit

- Python
- Ruby
- Perl
- PHP
- VBScript
- Smalltalk
- ActionScript 3
- ECMAScript 4
- and much more...

Good Fit

- TCL
- Lua
- ECMAScript 3
- Objective C
- C++

Don't Fit

- Lisp
- Prolog
- Scheme

Benefits for Community

- ❑ End users receive high-quality Python, Ruby and TCL IDEs with modern JDT-alike features
- ❑ Developers may implement support for the language of their choice with less effort
- ❑ Academic organisations get a base for research (type inference, code analysis, etc)
- ❑ Benefits for other Eclipse projects interested in scripting languages support
- ❑ Learn from JDT generalization experience

Current Community

- Community site at www.eclipsedltk.org
- Complete infrastructure: nightly and integration builds, CVS, bug tracker
- Interest and contributions from other projects and companies
 - Cisco Systems, Inc. (TCL support)
 - Eclipse Perl IDE (e-p-i-c.sourceforge.org)

Initial Team

- xored software, Inc.
 - Andrey Platov
 - Andrey Sobolev
 - Andrey Tarantsov
 - Dmitriy Kovalev
 - Mikhail Kalugin
 - Yuri Baburov

Deliverable Schedule

- January 23, 2007
 - Preliminary DLTK Core and TCL IDE complete

- March 1, 2007
 - DLTK Core 1.0 complete
 - Python IDE complete
 - Ruby IDE complete
 - Documentation complete

Future Directions

- ❑ Type inference improvements
- ❑ Code analysis improvements
- ❑ Refactoring improvements
- ❑ JDT integration
- ❑ More languages to be supported

Thank you!

- ❑ Questions?
- ❑ Please take a look on screenshots following

Screenshots

The image displays several overlapping screenshots from an IDE:

- Code Editor (Top Left):** Shows a `class Breakpoint:` with methods `delete`, `enable`, and `disable`. A tooltip for `enable` lists actions like `enabled Breakpoint - Breakpoint`.
- Code Editor (Top Center):** Shows a `def forget(self):` method and a `def setup(self, f, t):` method. A tooltip for `setup` shows its implementation.
- Code Editor (Bottom Left):** Shows a `def pm():` method. A context menu is open over it, with options like `Undo`, `Revert File`, `Open Declaration F3`, and `Show in Script Explorer`.
- Code Editor (Bottom Center):** Shows `import` and `from re` statements. A tooltip lists imported modules like `client`, `cmd`, `cmd Impr`, `dbg.py.dbg_engine`, `dbg.py.dbg_protocol`, and `dbg.py.python.dbg_engine`.
- Script Explorer (Right):** A tree view showing a project structure with folders like `package`, `Interpreter Libraries [tclsh]`, and `http1.0`. The `http_code(token)` method is selected.
- Hierarchy (Bottom Right):** A class hierarchy diagram showing `Person` as a base class for `Student` and `Teacher`. A list of methods for `Person` is shown: `getGender()`, `getName()`, `internalRemove()`, and `work()`.

More Screenshots

The image displays five overlapping screenshots from an IDE:

- Top Left:** A snippet of PHP code showing a protected function and a class structure.
- Top Right:** A 'Problems' window showing a list of errors and warnings. The table below summarizes the content:

	Description	Resource	In Folder	Location
✖	Syntax Error:un...	this.php	test3	line 4
✖	Syntax Error:ex...	this.php	test3	line 8
⚠	constant expres...	const.php	test2	line 5
⚠	assignment has ...	noeffect.php	test2	line 3
⚠	Unused paramet...	unused.php	test3	line 2

- Middle:** A code editor showing PHP functions: `function second_test ()` and `function third_test ()`. A warning icon is present next to line 13, which contains the text `Unreachable code: "Don't print\n";`.
- Bottom Left:** A 'Debug' console window showing a thread [1814] suspended at a breakpoint. The console lists 'recurse line: 7' and 'final line: 14'.
- Bottom Right:** A 'Variables' window showing a variable `$vendor` with the value `"xored software"`. A context menu is open over the variable with options: `Select All`, `Copy Variables`, and `Change Value...`.

Even More Screenshots

The screenshot displays three overlapping windows from an IDE:

- Console Window:** Shows a TCL script defining a `fact` procedure and its execution. The output is `120`.
- TCL Interpreters Dialog:** A dialog box for managing interpreters. It contains a table of installed interpreters:

Name	Location	Type
<input checked="" type="checkbox"/> tclsh.exe	C:\Tcl\bin\tclsh.exe	Generic TCL install
<input type="checkbox"/> wish.exe	C:\Tcl\bin\wish.exe	Generic TCL install

Buttons for `Add...`, `Edit...`, `Copy...`, `Remove`, and `Search...` are visible on the right side of the dialog.

- Code Editor:** Shows a TCL script defining a `sum` procedure and a `puts` command. A tooltip for the `sum(a, b)` procedure is displayed over the code, showing its arguments and description.