EclipseCon 2005
Poster Submission

Executable UML for Eclipse Platform

Personal Data

Main speaker

Speaker Name	Vadim Gurov
Job Title	Senior Software Developer
Affiliation	eVelopers Corporation
Primary Contact Info	vgurov@evelopers.com http://unimod.sf.net http://www.evelopers.com +7(812)324-3211
Primary Address	14, Bolshaya Raznochinnaya, korpus 5, office 517, St.Petersburg, 197110, Russia
Biographical Information	Vadim Gurov is a Senior Software Developer at the eVelopers Corporation. He has a master's degree in Computer Science from the St.Petersburg State University of Information Technologies, Mechanics and Optics and now is a post-graduate student in the same university. His research passion is model-driven development, applied UML and Web application modeling. Vadim has 10 years of software development experience with focus on OO programming.

Co-authors

Speaker Name	Maxim Mazin
Job Title	Software Developer
Affiliation	eVelopers Corporation
Primary Contact Info	mmazin@evelopers.com http://unimod.sf.net http://www.evelopers.com +7(812)324-3211
Primary Address	14, Bolshaya Raznochinnaya, korpus 5, office 517, St.Petersburg, 197110, Russia
Biographical Information	Maxim Mazin is a lead developer in UniMod project at eVelopers Corporation. He has a bachelor's degree in applied mathematics from the St.Petersburg State University of Information Technologies, Mechanics and Optics and now is a graduate student (master degree) at the same university. The subject of his bachelor's degree research dedicated to state machine validation and verification algorithms. He is focused on automata-based approach in programming research. He is interested in technologies that increase development tools usability.

Speaker Name	Maxim Korotkov
Job Title	Software Developer
Affiliation	eVelopers Corporation
Primary Contact Info	mkorotkov@evelopers.com http://unimod.sf.net http://www.evelopers.com +7(812)324-3211
Primary Address	14, Bolshaya Raznochinnaya, korpus 5, office 517, St.Petersburg, 197110, Russia
Biographical Information	Maxim Korotkov is a member of UniMod development team at eVelopers Corporation and a student of State University of Information Technologies, Mechanics and Optics in St.Petersburg, Russia. His major research interests include Eclipse plug-ins development and Executable UML. He is also interested in graph layout algorithms and language semantics related problems.

Poster Submission

Technology Exchange Title	Executable UML for Eclipse Platform
Audience level	Intermediate
Key words	UML, Executable UML, Eclipse, GEF, MDA, FSM, Finite State Machine
Brief Session Description	The idea of *Executable UML* is becoming increasingly popular now. There are a lot of *UML* editing tools, but they neither support the idea of *Executable UML* nor provide convenient facilities for diagram editing, as existing editing tools for textual programming languages do. *UniMod* is an open source plug-in for *Eclipse* platform that addresses these issues. *UniMod* implements editor for *UML Class* and *Statechart* diagrams using Graphical Editing Framework. With a help of adding *interpretation rules* to these diagrams *UniMod* realizes the idea of *Executable UML*. Also *UniMod* ports textual programming languages code assist technologies to UML diagrams editing.
Delegate Prerequisites	Interest in Eclipse and UML tools
On-site equipment requirements	Wintel PC with Microsoft PowerPoint 2002 installed

Abstracts

The idea of *Executable UML* is becoming increasingly popular now. There are a lot of *UML* editing tools, but they neither support the idea of *Executable UML* nor provide convenient facilities for diagram editing, as existing editing tools for textual programming languages (*TPL*) do. *UniMod* is an open source plug-in for *Eclipse* platform that addresses these issues. *UniMod* implements editor for *UML Class* and *Statechart* diagrams using Graphical Editing Framework (*GEF*). With a help of adding *interpretation rules* to these diagrams *UniMod* realizes the idea of *Executable UML*.

Modern editing tools for *TPL*, *Eclipse JDT* and *IDEA*, for example, provide such code assist technologies as syntax and semantics check, auto-completion and quick fix, code formatting and refactoring, launch and debugging from *IDE*. These technologies are generalized to diagram editing and implemented in *UniMod*.

TPL editing tools check if program belongs to given language and highlight code fragments containing syntax errors. Undefined variables usage, invocation of non-existent methods, etc. are treated as semantics errors and also highlighted. As for *UML* diagrams, there is a set of constraints defined by *UML* specification that must be satisfied by well-formed diagram. *UniMod* defines some additional constraints, such as attainability of every state on Statechart diagram and completeness of set of outgoing transitions for every state.

While the user edits diagrams, *UniMod* plug-in validates them in background and highlights the figures breaking the constraints. For every broken constraint, *Marker* is created and shown in *Problems* view. Eclipse *Marker* allows to have associated position in text, but for diagrams we need to have associated graphical figure. To solve this, special map is created that for every *Market* stores set of associated figures.

Implementation of auto-completion and quick fix for *TPL* is usually based on grammar of programming language and set of semantic rules. For diagrams *UniMod* builds these technologies at the top of diagrams' constraints. For every broken constraint *UniMod* provides set of possible resolutions. For example, if some state on your Statechart diagram is unattainable, *UniMod* suggests you to add transition to this state from some attainable state.

To launch *TPL* program it needs to be compiled and passed to some runtime engine (*OS* or *JVM*). Usually any modern *IDE* allows compiling and launching *TPL* program in one click. To port this approach for diagrams, on user's launch request, *UniMod* converts *UML* diagrams content into *XML*-description and starts its own runtime engine that interprets generated *XML*-description. Diagram execution is implemented using *Eclipse* launching framework.

Traditional debugging of programs is based on operator-by-operator code tracing with variables' values analysis. In *UniMod* application behavior is defined using *UML Statechart* diagram, so debugging is based on state-by-state diagram tracing with trigger events, guard conditions and output actions analysis.

Finally, an important point is that *UniMod* project is an attempt to design and implement programming language of the next generation — graphical programming language — based on *Structural Finite State Machine* paradigm, *UML* notation and *Eclipse* platform. *UniMod* is a successful attempt to port *TPL* code assist technologies to diagrams.