

Conceptual Programming with a Linguistic Processor

English research synopsis of the 2011 author's abstract

Research question

How can software systems understand domain-specific instructions written in natural language and turn them into program behaviour without requiring every user to learn a conventional programming language?

The dissertation investigates conceptual programming, domain-specific languages and a linguistic processor as a bridge between human descriptions of problems and executable object-oriented software.

The proposed system: NALAPS

NALAPS combines morphological analysis, syntactic-semantic analysis, conceptual processing, conceptual graph construction, Java class specifications and code generation. Rather than treating a sentence merely as a sequence of words, the system maps domain concepts and their relations into a formal representation that can be interpreted by software.

Its knowledge base associates domain vocabulary with object-oriented classes and their behaviour. A natural-language specification tool links Java program components with the terminology of a particular application domain. A dynamic object system represents the meaning of processed text, and reverse processing supports the reconstruction of conceptual relations.

Practical applications explored

The dissertation describes generating software tests from textual issue descriptions; using natural language to configure business rules; controlling objects in a simulated 3D environment; and constructing genealogical relations from descriptions in ordinary language.

The work also presents an IntelliJ IDEA integration, bringing the creation of domain-language knowledge into the software development environment.

Relationship to modern AI

The central objective resembles that of modern AI coding assistants: translate human intention into executable software. The underlying mechanism is different. NALAPS uses explicitly defined linguistic analysis, domain knowledge and formal conceptual structures, whereas modern large language models learn statistical patterns from training data.

This makes the 2011 research relevant to discussions of structured reasoning, grounded semantics, domain-specific agents and verification. It does not claim to anticipate today's transformer architectures or generative models.

Original research

Andrey S. Lebedev (2011). Development and Investigation of a Conceptual Programming System Using a Linguistic Processor. Moscow Institute of Electronics and Mathematics. Russian dissertation and author's abstract are included separately.

Important: This is an English research synopsis prepared from the original Russian dissertation. It is not a complete translation of the thesis or of the author's abstract.