

From Natural Language to Executable Software

Extended English synopsis of the 2011 dissertation

Research context and motivation

Software changes frequently arise as requests written in business language. Translating those requests into modifications to program logic is difficult because business concepts and programming structures are expressed differently. The thesis positions natural-language domain-specific programming as one approach to reducing that gap.

It reviews morphological and syntactic parsers, semantic analysis, natural-language programming, conceptual programming, ontologies and the Web Ontology Language (OWL). It also examines augmented transition networks, conceptual dependency and conceptual graphs.

Architecture of NALAPS

The system is decomposed into a linguistic processor, a conceptual processor, a natural-language specification mechanism for Java, a code generator, a dynamic object system and reverse processors. A visual editor for augmented transition networks supports the creation of language-processing rules.

The morphology component uses Yandex Mystem; syntactic-semantic processing derives relationships among concepts. Domain ontologies are represented using object-oriented classes, which are connected to language expressions through configuration metadata.

Representing meaning

The thesis proposes representing the meaning of natural-language text as a dynamic system of Java objects, complemented by a conceptual graph. It provides a formal model for mappings between linguistic structures, conceptual graph elements and program objects using sets and graph-theoretic notation.

This intermediate representation helps isolate the meaning of a domain statement from its surface wording and makes it possible to associate concepts with operations defined in code.

Bidirectional processing and development tools

Beyond translating an input sentence into a conceptual representation, the work explores reverse transitions from changes in the dynamic object model back into a conceptual graph and natural-language descriptions.

The proposed IntelliJ IDEA integration allows programmers to specify domain vocabulary and links between Java classes and words or expressions while working in an ordinary development environment.

Demonstrated applications

Software verification: transforming natural-language reports of defects or requested changes into automatically generated tests.

Business rules: enabling configuration of software behaviour through natural-language instructions rather than low-level edits.

Interactive environments: using textual instructions to manipulate objects in a virtual 3D room.

Knowledge representation: deriving genealogical relationships and a family tree from a written description.

A connection with today's AI systems

Large language models have radically expanded the ability to interpret open-ended instructions. Yet using generated text or code in dependable applications still raises questions of domain grounding, formal representations, safety and reproducibility. NALAPS is an earlier symbolic approach to this interface between human meaning and executable structure.

The relationship is conceptual rather than architectural: this was not a neural model, and its capabilities should not be confused with present-day generative AI.

Publication information

Defended in 2011 at the Moscow Institute of Electronics and Mathematics (MIEM), Russia. The thesis is a candidate-of-technical-sciences dissertation in speciality 05.13.11.

The Russian dissertation and the author's abstract, with the original research figures, formulae and mathematical derivations, are the authoritative versions. This synopsis is provided for English-speaking readers.

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.