The Selection Problem and Evaluating Method for Architectural Design Tools of Computer Systems

Aleksandr Penskoi
Faculty of Software Engineering and Computer Systems
ITMO University, Saint-Petersburg, Russia
aleksandr.penskoi@gmail.com

ITMO UNIVERSITY

Architectural Design Tools

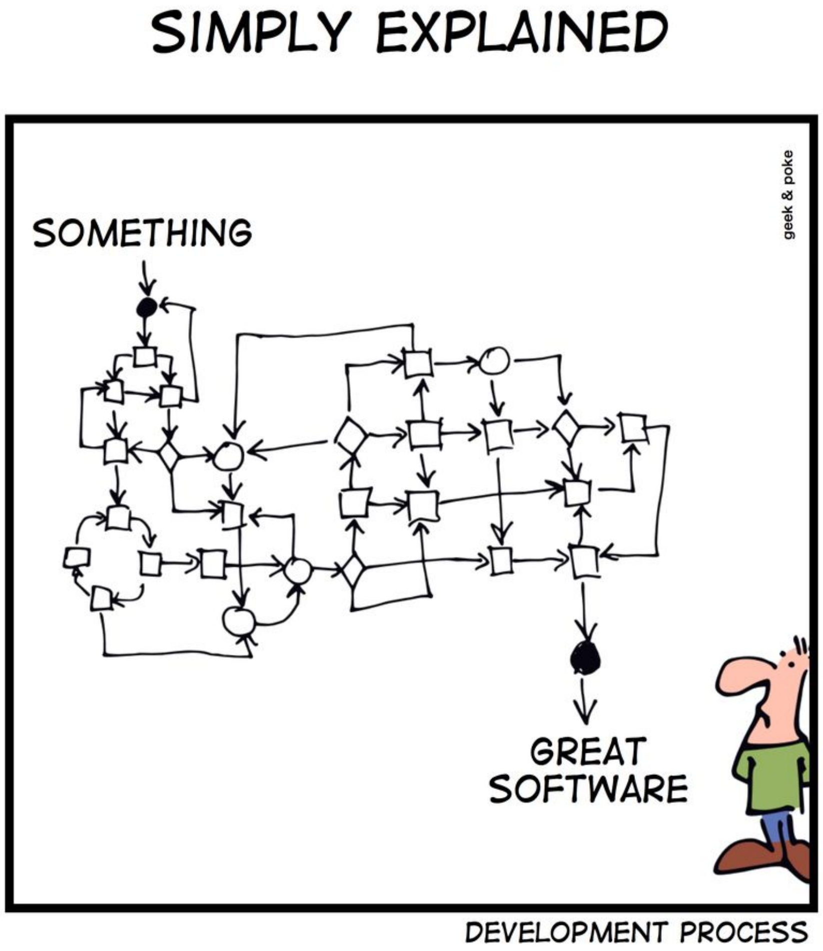
Architecture (of a system) — fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution

ISO/IEC/IEEE 42010:2011
Systems and software engineering -- Architecture description

Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled

Eoin Woods (Software Architect, Investment Bank, London, UK),
http://www.sei.cmu.edu/architecture/start/glossary/community.cfm

Examples of architectural design tools: architectural styles, architectural description languages, development and system analyzing methods, models of computations, programming styles, paradigms and languages, etc.



Problem statement

Example of reaction to a new architecture level design tool:

- I can resolve the task without it, why I should spend time on learning?
- Why do you consider your way of thinking is better than mine?
- Can you prove it by quantification of the effect?

How to compare architectural level design tools
of the same type for a particular class of tasks?

Outline:

- Examples: OMG Essence, System Engineering, OOP vs FP, and practical design problem.
- Features and Problems of Architectural Design Tools Comparing.
- Comparative analysis based on criteria with partially ordered estimates.

Why it is so important?

- Wrong tools can significantly increase the project budget and fail it.

- How we can move from “computer system developing” to “computer system engineering” with:
  - Predictable budget and duration?
  - Predictable system properties?
  - Reproducible results?

- How can we write papers on such important things?

Не хочу видеть никаких сумасшедших торговцев – ты что, не видишь, что тут битва идёт!

Источник: FIATECH

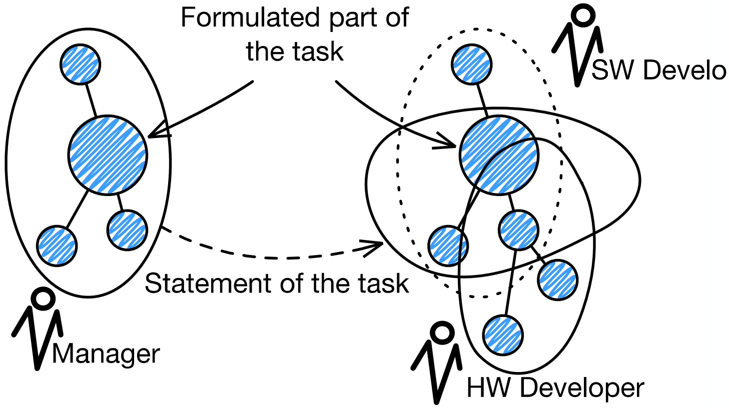
Why it is so hard?

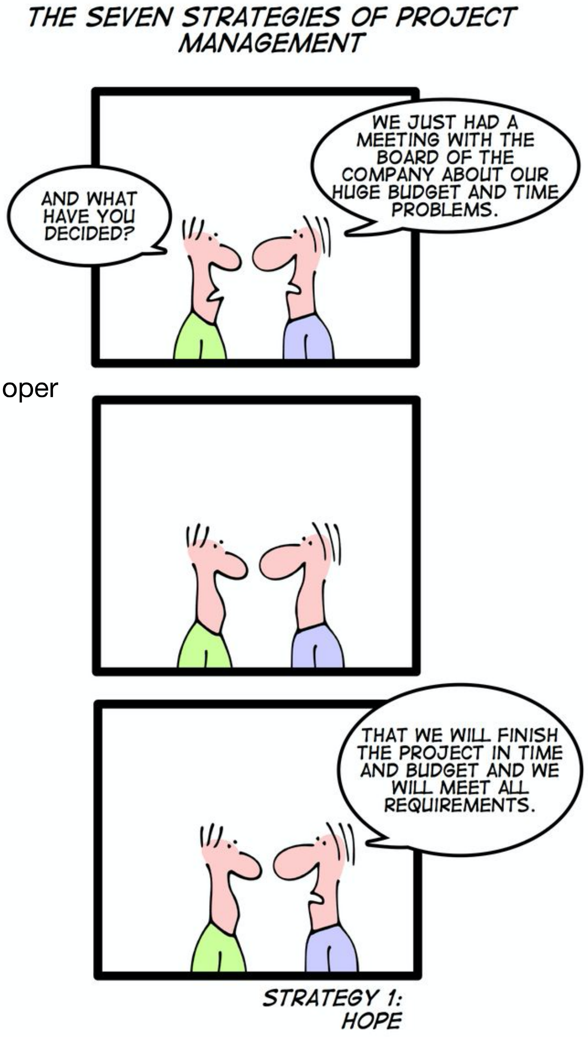
- Business
  - Human Resources
  - Risks
  - Time to market

- Reputation, Management, Team
  - Real reasons
  - Real goals

- Personal bias and fashion

- Very complex technical trade-offs




THE SEVEN STRATEGIES OF PROJECT MANAGEMENT
WE JUST HAD A MEETING WITH THE BOARD OF THE COMPANY ABOUT OUR HUGE BUDGET AND TIME PROBLEMS.
AND WHAT HAVE YOU DECIDED?
THAT WE WILL FINISH THE PROJECT IN TIME AND BUDGET AND WE WILL MEET ALL REQUIREMENTS.
STRATEGY 1: HOPE

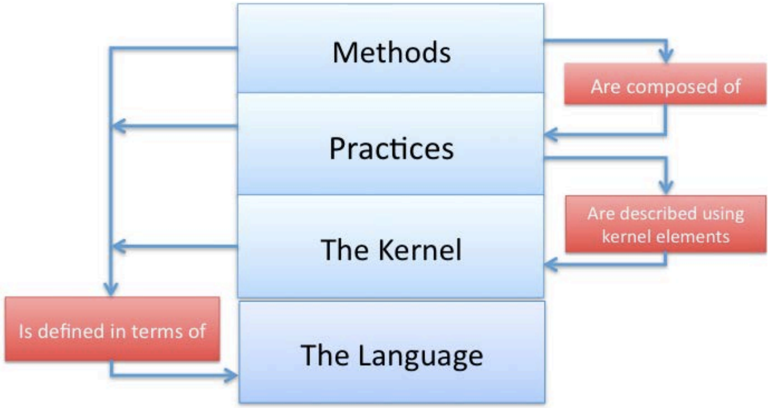
We will speak about the last one.

Example #1: OMG Essence
How to select software development methodology?

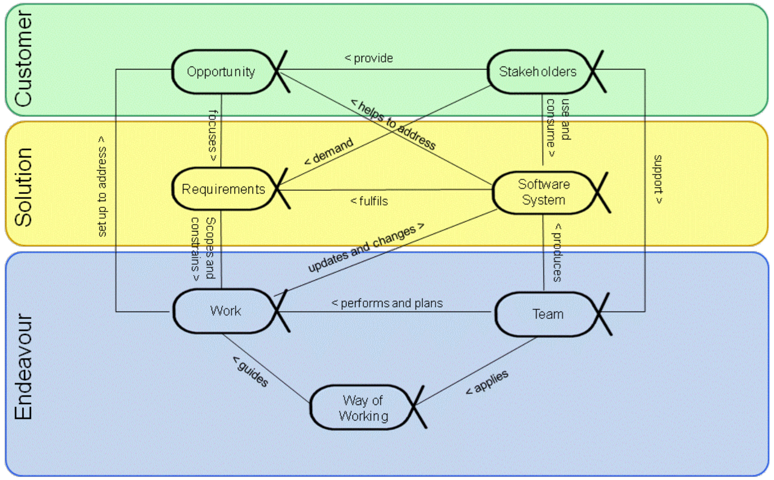
Ivar Jacobson is a computer scientist and software engineer, known as major contributor to UML, Objectory, Rational Unified Process, aspect-oriented software development and **Essence**.

- In the 80s the problem was formulated: industry need a methodology for software development.
- In 2009 the Software Engineering Method and Theory (SEMAT) initiative was launched.
- In 2014 the first version of Essence standard was published (280 pages).
- Today, OMG Essence (the common ground for defining software development practices) is the basis of many courses on software engineering methodology.

I. Jacobson, “Technical Trends Discover the Essence of Software Engineering,” CSI Commun., pp. 12–14, Jul. 2011.



The Method Architecture



The Kernel Alphas

Example #2: System Engineering
Is it applicable for Software-Intensive Systems?

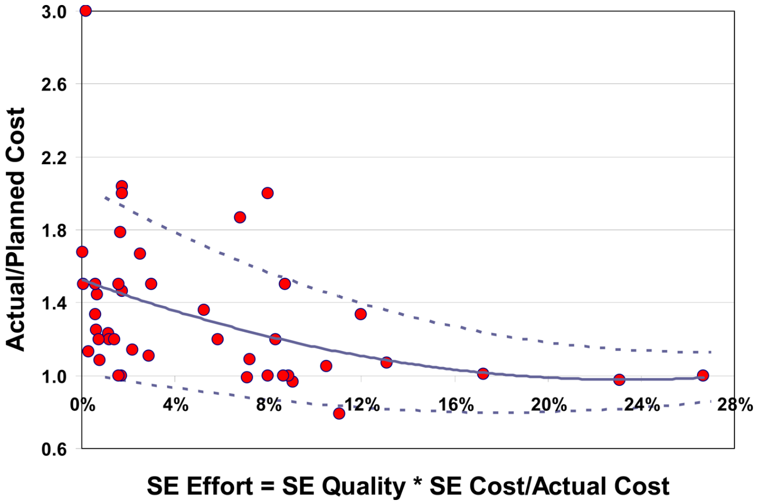
System Engineering (SE) is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on … evolving solutions while considering the complete problem, from system concept exploration through system disposal.

The Guide to the Systems Engineering Body of Knowledge (SEBoK), V. 1.3.

Some quantitative results for software-intensive systems:

- 161 software projects in the COCOMO II database collected over a 25-year period.
- Conclusion: SE practice allow to reducing risk of cost overrun and schedule overrun.

Boehm, B., Valerdi, R., & Honour, E. (2008). The ROI of systems engineering: Some quantitative results for software-intensive systems. Syst. Eng., 11(3), 221–234. https://doi.org/10.1002/sys.v11:3



Example #3: Object-oriented programming

- Object-oriented programming is an exceptionally bad idea which could only have originated in California

(c) Edgar Dijkstra

- OOP has many conceptual and practical issues, but heavy wide speared.

Partridge, C (1996). Business Objects: Re-Engineering for Re-Use, Butterworth Heinemann, 1996

- We don't have the benefits of OOP over procedural programming.

Potok, T. E., Vouk, M., & Rindos, A. (1999). Productivity analysis of object-oriented software developed in a commercial environment. Software: Practice and Experience, 29(10), 833-847.

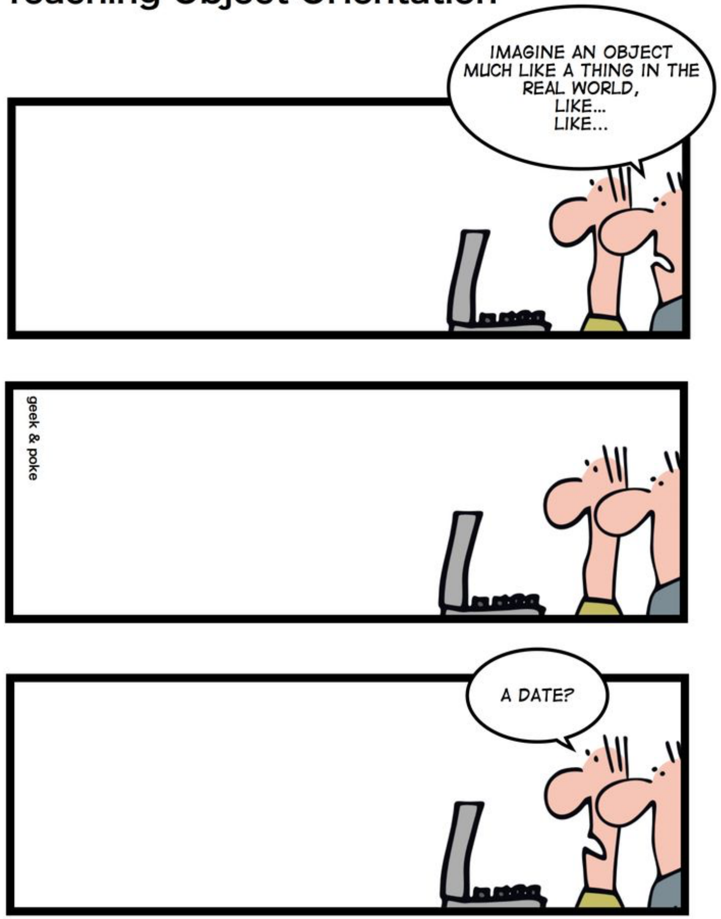
- Current trends in programming languages:
  - add functional style features;
  - reducing mutable state;
  - avoid inheritance.

C++, Java, Go, Rust

- OOP is hard to use properly.

Sierra, Kathy, and Bert Bates. Head First Java: A Brain-Friendly Guide. O'Reilly Media, Inc., 2005.
Fowler, M. Refactoring: improving the design of existing code. Addison-Wesley, 2018.

Teaching Object Orientation


IMAGINE AN OBJECT MUCH LIKE A THING IN THE REAL WORLD, LIKE... LIKE...
geek & poke
A DATE?

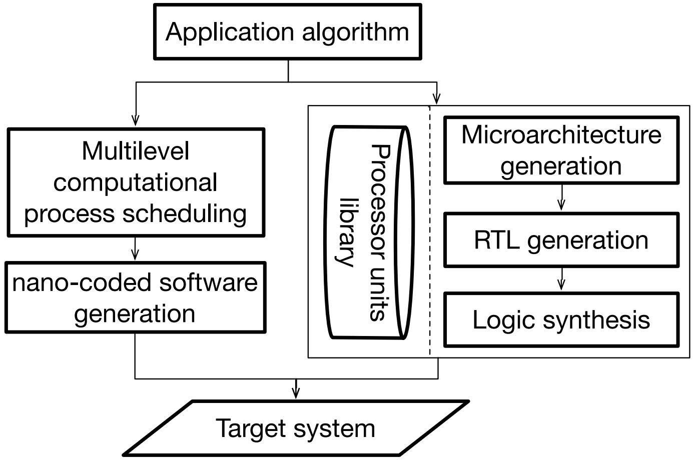
Today: `java.util.Date`

Example #4: Design Real-Time ASIP with Complex Multi-Level Organisation

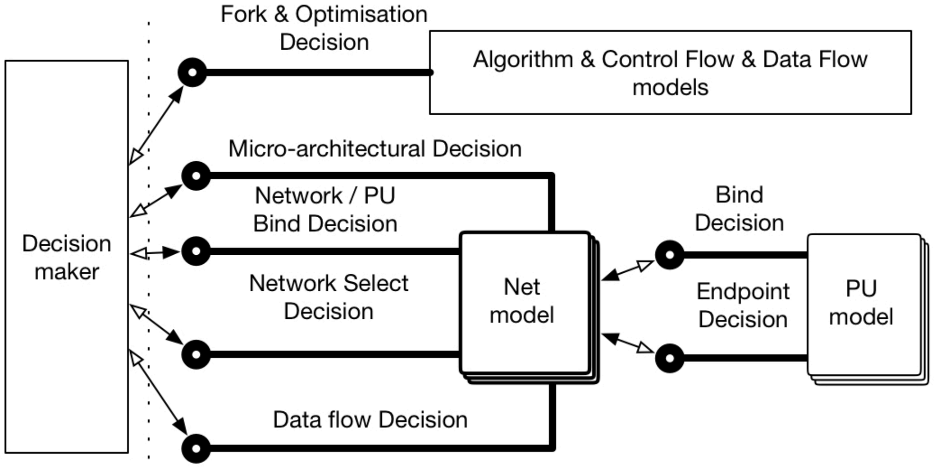
NITTA project is a hard real-time ASIP with the hybrid No Instruction Set Computing - Transport Triggered Architecture (NISC-TTA) architecture. A target system includes:

- a synthesizable HDL project with specialized processor units (from the standard library and/or user-defined), interconnect infrastructure and distributed control units;
- a nano-coded software, which defines system behavior according to an application model/ algorithm and, if it applies, specification of system interaction protocol.

Synthesis method based on the transport-oriented model of the target processor, which can represent all possible system behaviors for CAD.


Application algorithm
Multilevel computational process scheduling
nano-coded software generation
Processor units library
Microarchitecture generation
RTL generation
Logic synthesis
Target system

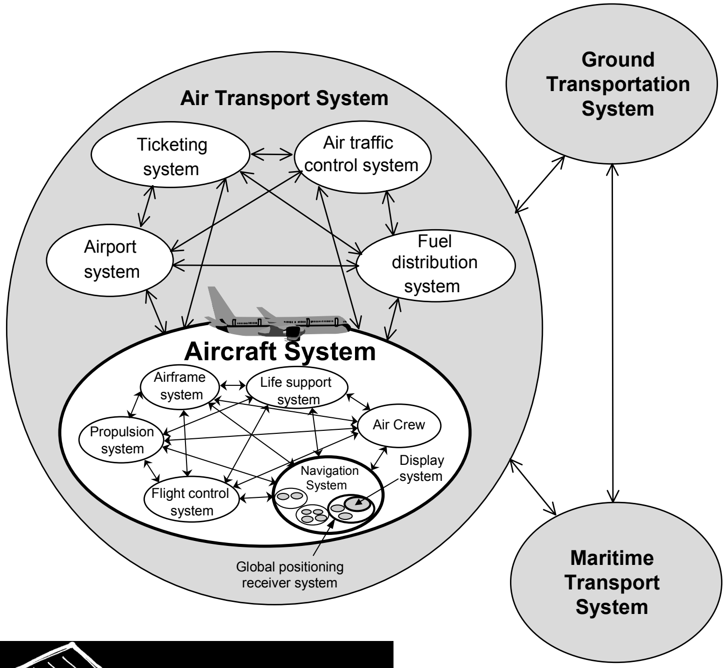
Design flow


Decision maker
Fork & Optimisation Decision
Algorithm & Control Flow & Data Flow models
Micro-architectural Decision
Network / PU Bind Decision
Network Select Decision
Net model
Bind Decision
Endpoint Decision
PU model
Data flow Decision

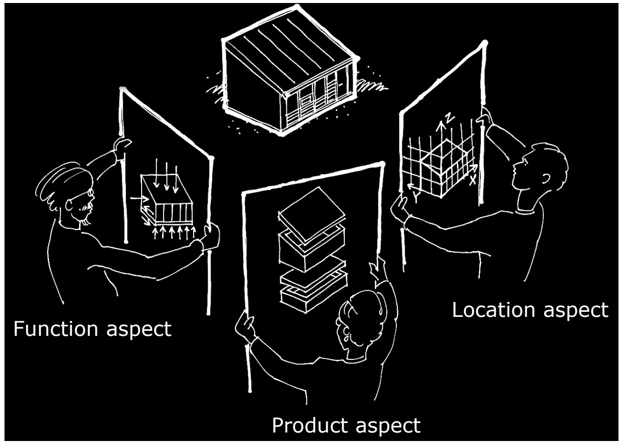
Synthesis method architecture

Problems of Architectural Design Tools Comparing

- Low formalization of design space of problem. Different tasks required different questions, viewpoints, and granularity level.
- Low formalization of compared tools. Usually, a common ground for comparing different tools is not allowed.
- Dependency between tool selection and consequences (after architectural design and implementation). The human factor.
- Problems of experimental evaluation:
    - Task selection for an experiment.
    - Experiment organization.
    - High influence of the human factor.
    - Existed industrial experience usually not accessible.



ISO 15288



ISO 81346

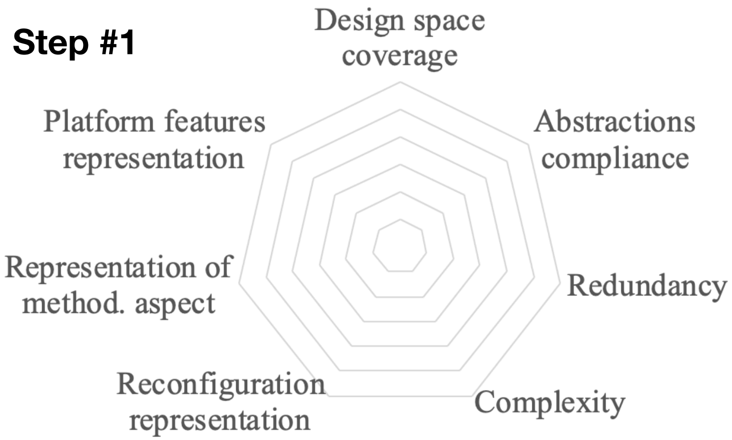
Comparative analysis based on criteria with partially ordered estimates

Example: comparing architectural styles for design and documenting multi-level embedded systems.

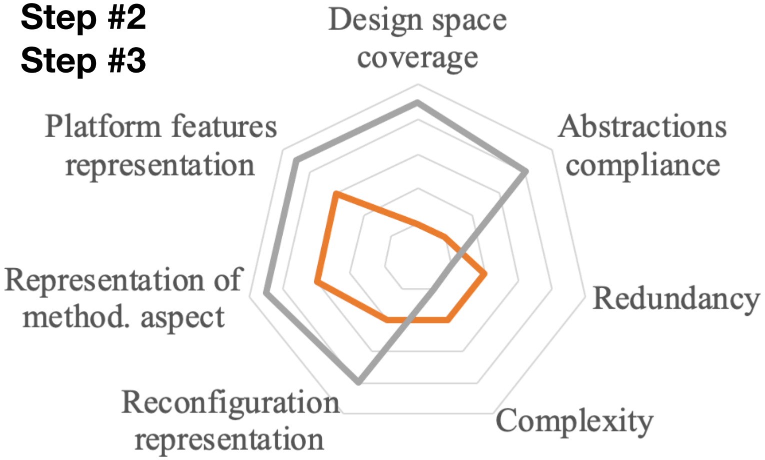
Method steps:

1. Definition of criteria with partially ordered estimates. Criteria can be objective or subjective and should be represented as axis on radial diagrams.
2. Evaluation of compared objects. Estimate doesn't need to be normalized for single or multiple criteria, all that needed is order and equality relationship.
3. Reduction of the number of compared objects by removing duplicates and poor options.
4. Comparative analysis. Finding proper use-cases for available tradeoffs or challenges for creating new tools.

Step #1


Design space coverage
Abstractions compliance
Redundancy
Complexity
Reconfiguration representation
Representation of method. aspect
Platform features representation

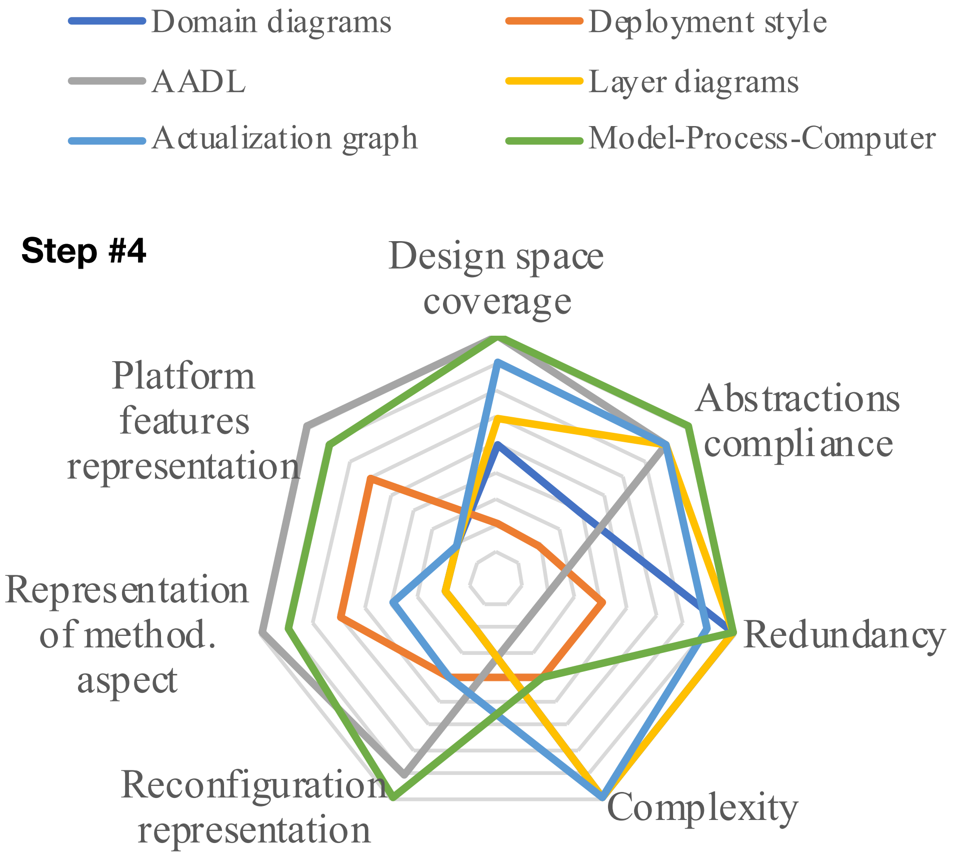
Step #2
Step #3



Step #4 on the next slide

Conclusion:

- Some of the existed examples of architectural design tools comparing reviewed.
- Problems of architectural design tools comparing are analyzed.
- Proposed the method of comparative analysis of architectural design tools based on criteria with partially ordered estimates with the following properties:
  - the analytical method based on explicit and traceable expert estimates;
  - support of "black-box" and "white-box" analyzing;
  - method oriented to highlight differences, not to obtain the single best solution;
  - method provides the means for interpreting the results in different application cases.



Thank you!

aleksandr.penskoi@gmail.com