Faculty of Mathematics and Physics
Charles University in Prague

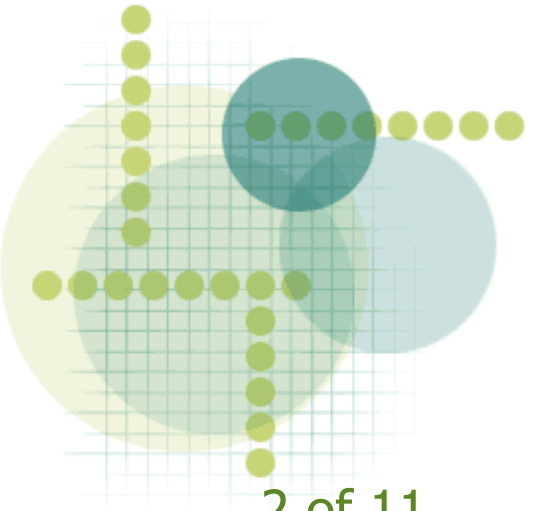
Optimizing of Software Connectors Code Generator's Performance

Pavel Petřek, MSc.

Supervisor: RNDr. Tomáš Bureš, Ph.D.

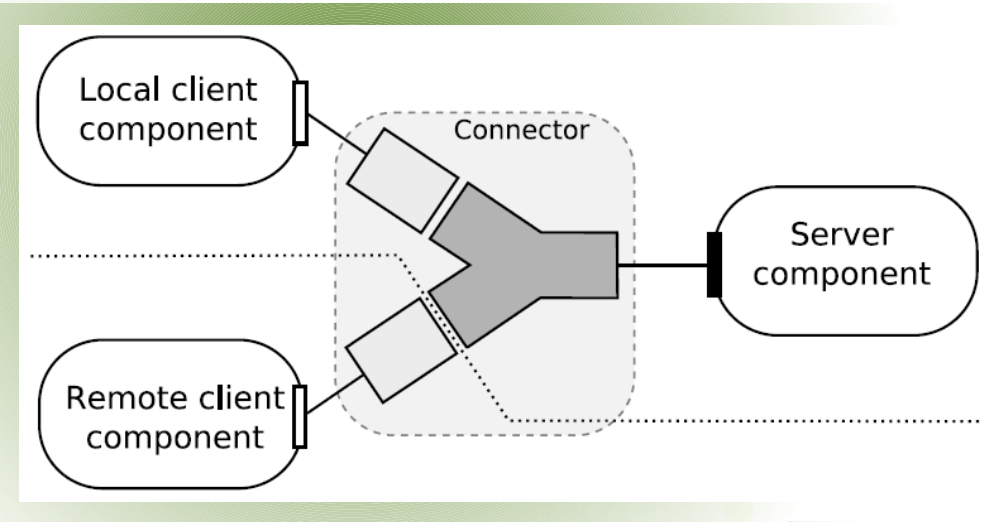
Agenda

- Introduction of connectors and their generation
- Introduction of the generation bottleneck
- Proposed solution
- Conclusion



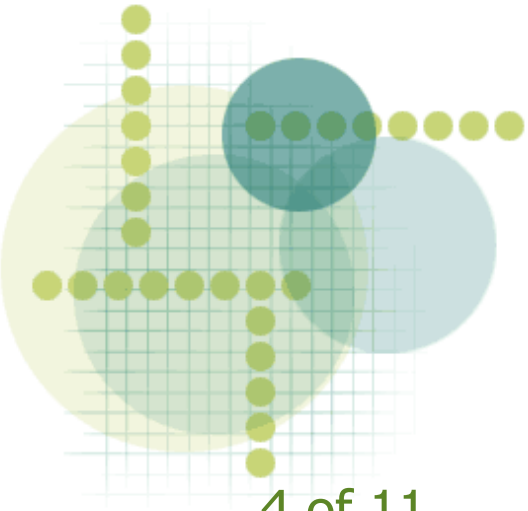
Introduction of connectors

- What are connectors
  - Entities that interconnect software components
  - Configurable communication layer
  - Connectors can be generated according to a concrete set of components
- Connector's lifecycle
  - Design-time
  - Deployment-time
  - Run-time



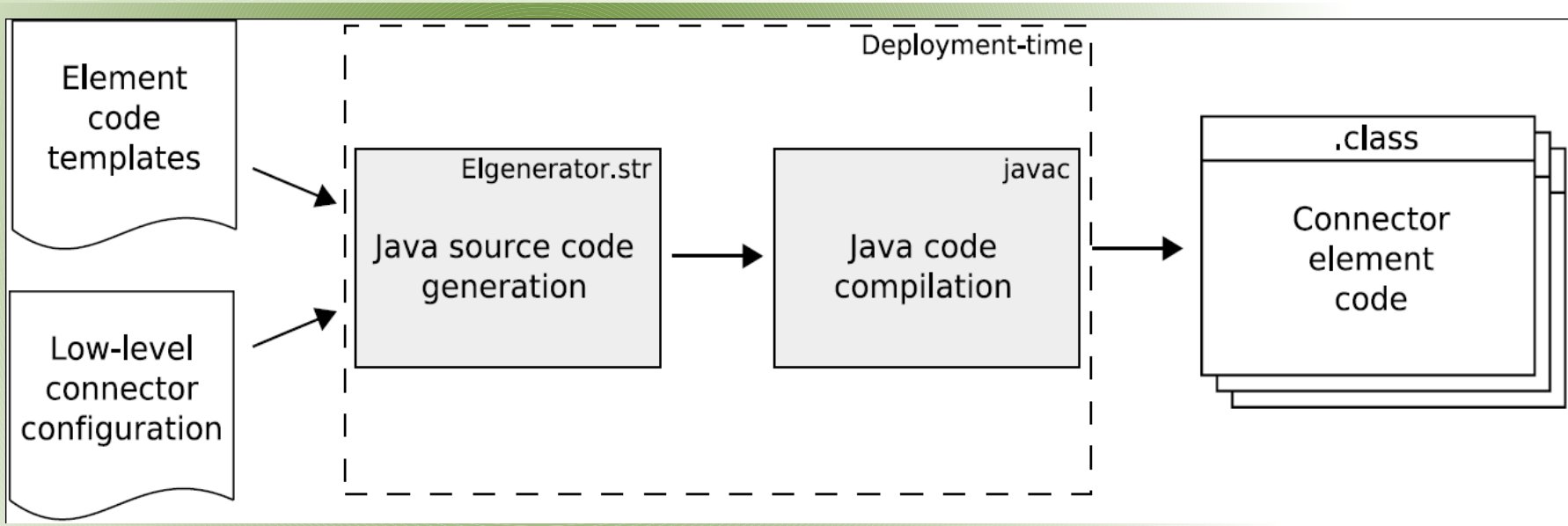
Generation of connectors

- ConGen software developed in DSRG at MFF
- Two stages of generation
  - Architecture resolution
  - **Connector's code generation**



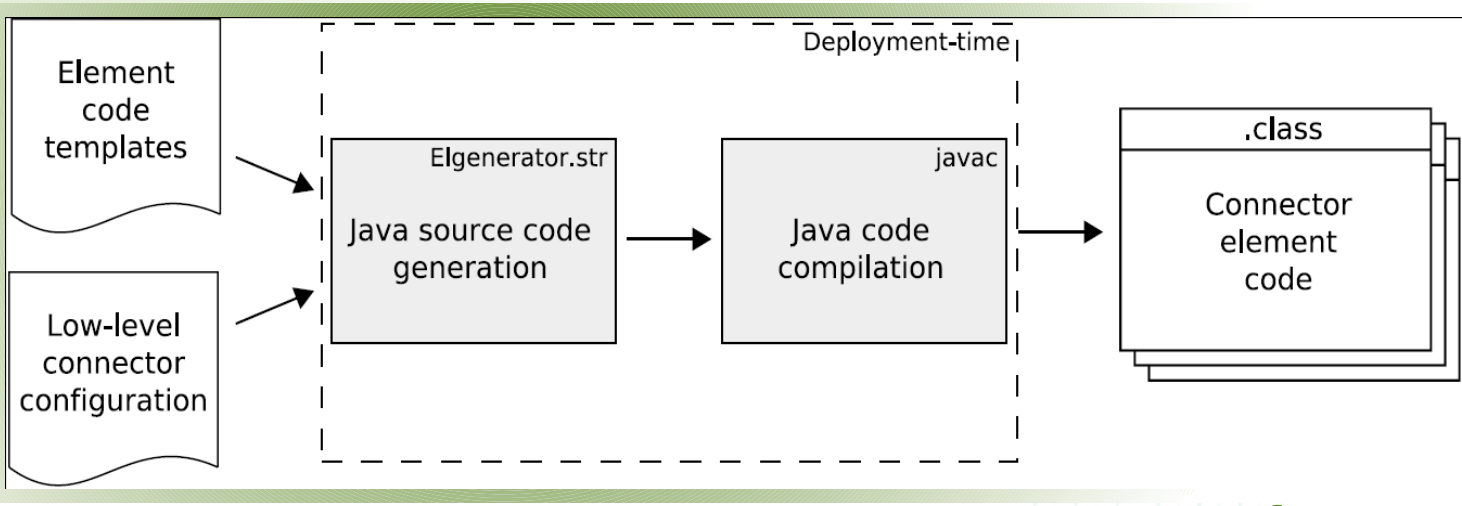
Code generation

- Code generation is based on code templates
- Source code is generated from a template
- The generated source code is compiled to real connector's code



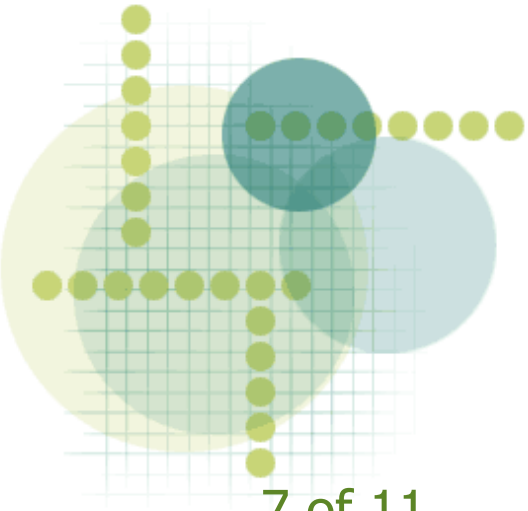
Code generation bottleneck

- The generation proceeds at the deployment-time
- SDK is needed for the compilation of the source code
- My goal was to optimize the SDK requirement
- My proposed solution was **the precompilation of the templates**



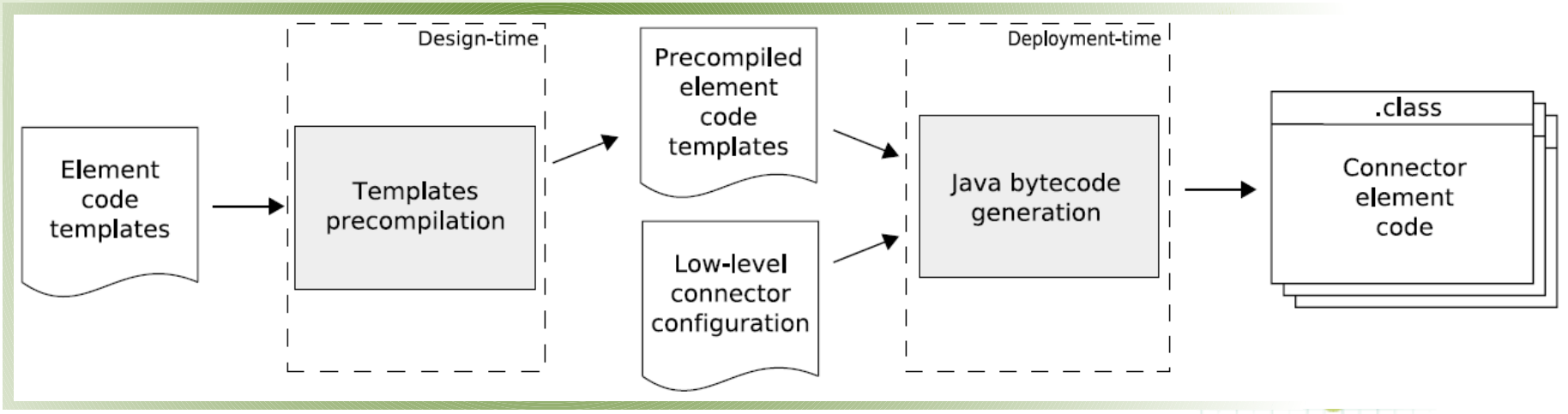
Proposed solution – part 1

- Source code templates are precompiled to a form that does not require Java SDK for the compilation of the connector's code
- Template syntax is preserved
- Precompilation step is non-invasive extension to existing ConGen



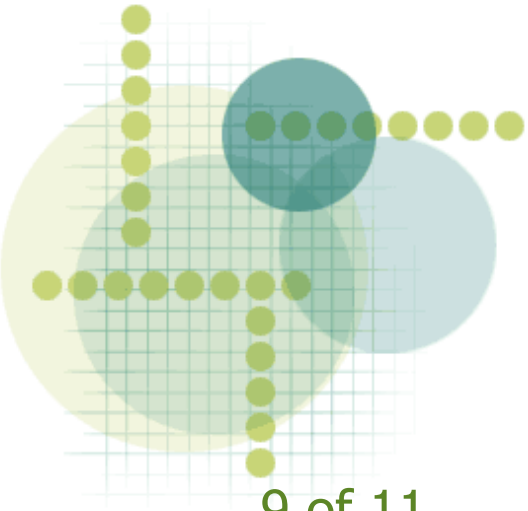
Proposed solution – part 2

- At the design-time **the Java source code manipulation** and Java SDK are used for obtaining the precompiled form of template
  - Java SDK is always at the design-time
- At the deployment-time **the bytecode manipulation** is used to produce the final connector's class files



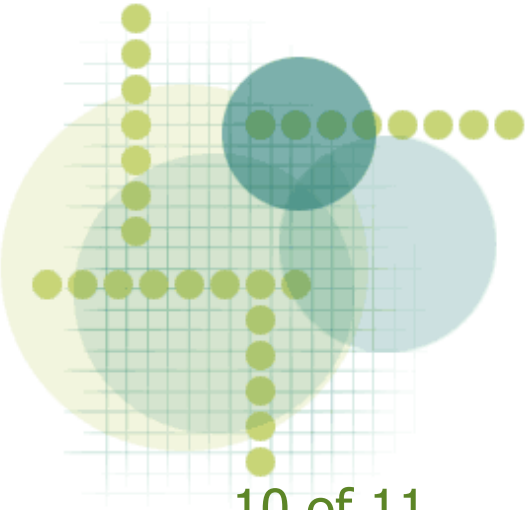
Conclusion

- Producing of class files is more than 5 times faster using precompiled templates
- Java source code compilation is no more needed at the deployment-time
- Solution works with the original templates
- Precompilation step is fully optional

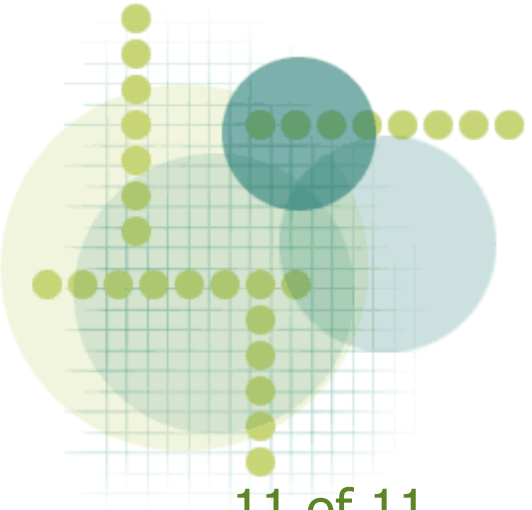


Questions and answers

?



Thank you for your attention...



Future of Java code compilation

- Static references are part of the Java language concept => will not be removed
  - may be extended => extension of my solution
- Java promises WORA (Write Once, Run Anywhere), any JVM (HW, SW) must be equally able to run the compiled bytecode => compile-time optimizations are minus
  - Many projects preparing code for certain platform, but it proceeds on already compiled code

Solution rationales

- Alternative solution was to integrate a Java compiler into ConGen
  - Too weak compilers
  - Or too large compilers
- Precompilation means no additional requirement to environment and performance benefits