

Bachelor Thesis

A Language Server and IDE Plugin for CPAChecker

Adrian Leimeister

Software and Computational Systems Lab
LMU Munich

15.07.2020

Motivation

Powerful IDEs are available, with support for all kind of different development tools.

Options for integrating formal verification into a graphical development workflow exist,

but:

- offer only rudimentary integration (Predator)
- rely on old IDE versions (CMBC/CProver)
- require seperate installation of tools
- are only available for specific IDEs

Goal

Implement a CPAchecker Language Server Protocol (LSP) server and a client plugin for an IDE

Requirements:

- implement LSP server for CPAchecker
- allow CPAchecker configuration for basic use
- allow verification via CPAchecker locally
- allow verification via VerifierCloud
- visualize results in Eclipse via LSP client plugin

Software Components: Language Server Protocol (LSP)

The Problem: a Matrix

	Go	Java	TypeScript	...	Language N
Emacs	Plugin 1	Plugin 2	Plugin 3	...	Plugin N
Vim	Plugin 1+N	Plugin 2+N	Plugin N*2
VSCoDe	...				
...					
IDE M					Plugin N*M

N Languages for M IDEs requires N*M Plugins!

Software Components: Language Server Protocol (LSP)

LSP: split tool support into two parts, client and server

The Solution: Clients and Servers

Language	Server	IDE	Client
Go	Server 1	Emacs	Client 1
Java	Server 2	Vim	Client 2
TypeScript	...	VSCoDe	...
...		...	
Language N	Server N	IDE M	Client M

N Languages for M IDEs requires only N servers and M plugins/clients!

Software Components: Language Server Protocol (LSP)

So, is LSP the solution to all Problems?

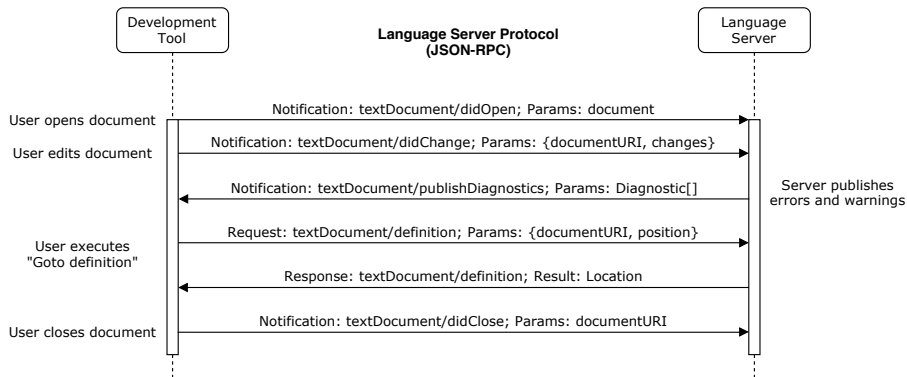
No:

- debugging not handled by the LSP
- server specific configuration in client

⇒ Number of Plugins required is more than $N+M$, but reusable "intelligent" server

Software Components: Language Server Protocol (LSP)

Example communication between IDE and language server:



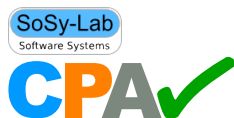
Software Components: Eclipse IDE

- IDE with lots of features
- support for lots of tools
- written in Java
- extensible with plugins (everything is a plugin)
- basic LSP client plugin, LSP4E



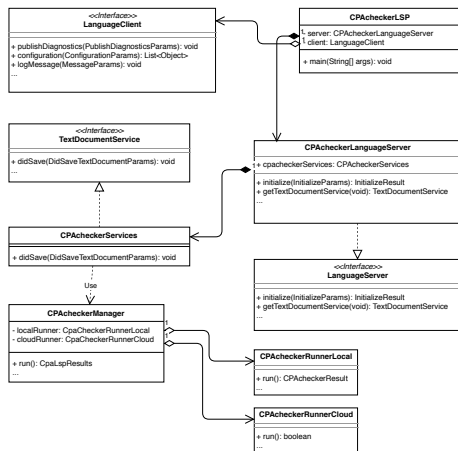
Software Components: CPAchecker

- framework for formal verification
- written in Java
- extensible by implementing "Configurable Program Analysis" (CPA) interface
- cloud verification available via API
- program converted to control flow automata (CFA)
- CFA is tested with CPAs against specification
- can be configured to produce violation witness



Implementation: Language Server

- started by language client
- communicates with client via JSON-RPC
- receives notification on file save
- requests configuration from client
- starts the configured runner
- sends result back to client



Implementation: Language Client

- based on LSP4E
- includes an installation of CPAchecker
- enables basic configuration of verification tasks
- interacts with Eclipse via extension points
- starts LSP server
- sends and receives messages to/from LSP server
- visualizes received information

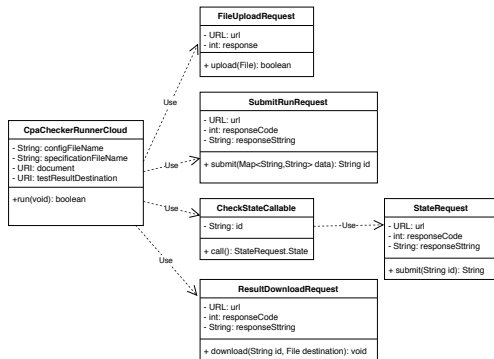
Extension points

- content type: extension of C source file
- content type binding: open content type with C editor
- content type mapping: start LSP server when content type is opened
- ...

Implementation: Interfacing with the VerifierCloud

Communication with the VerifierCloud using HTTP requests:

- calculate file hash
- upload file to cloud
- submit run using file hash and configuration
- periodically check for completion
- download result



Implementation: Interfacing with CPAchecker

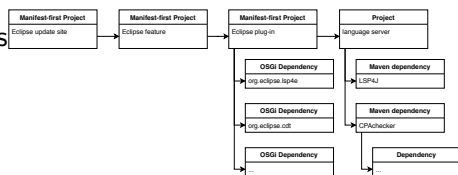
CPAchecker classes are used directly.

Main tasks for interfacing with CPAchecker:

- CPAchecker configuration creation
- message handling
- result processing

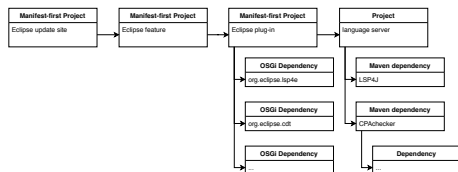
Implementation: Build Process

- built with Maven using Tycho
- Maven and Tycho have different repository types
- Eclipse plugins are OSGi bundles
- plugin dependencies are OSGi bundles
- plugin dependencies defined by OSGi manifest
- Maven Eclipse plugin projects are called "Manifest-first"



Implementation: Build Process

- Maven project dependencies defined by POM
- "POM dependencies" not loaded by Eclipse runtime
- cpachecker-lsp language server not an OSGi bundle
- OSGi manifest generation possible, called "POM-first"
- Problem: "Manifest-first" and "POM-first" incompatible in multi-module project



The Solution:

- keep cpachecker-lsp as "POM dependency"
- add cpachecker-lsp build artefact to plugin output
- add cpachecker-lsp to plugin classpath

Evaluation: Method

Survey using Google Forms among students and employees at SoSy-Lab.

Gathering feedback:

- problems regarding installation process
- sufficiency of configuration options
- satisfaction with presentation of results
- ideas for improvement

Evaluation: Results

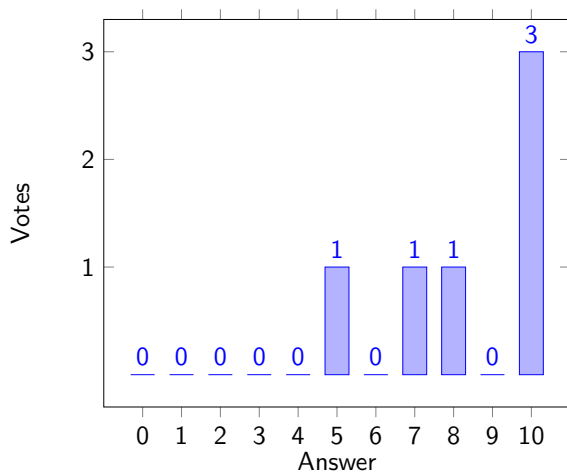
Installation:

- no problems
- caveat: requires at least Eclipse 2019-06

Evaluation: Results

Configuration:

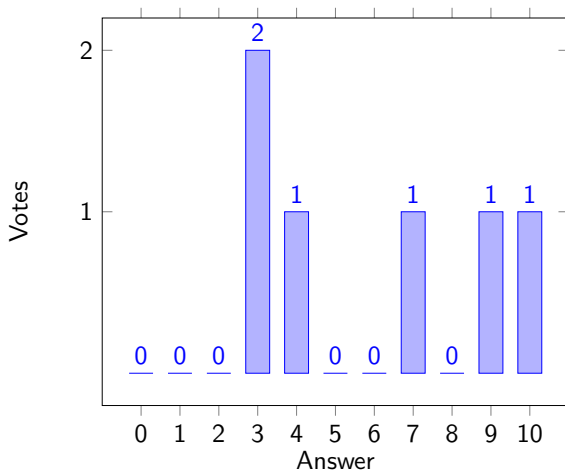
Are the configuration options enough to cover your use case?



Evaluation: Results

Presentation of Results:

Are you satisfied with the presentation of the results?



Evaluation: Results

Bugs:

- wrong Java version used for starting the server ✓
- configuration corruption ✓
- confusing notification ✓
- cloud verification not working ✓
- error markers not disappearing after fixing an error ✗
- language server stopped working ✗

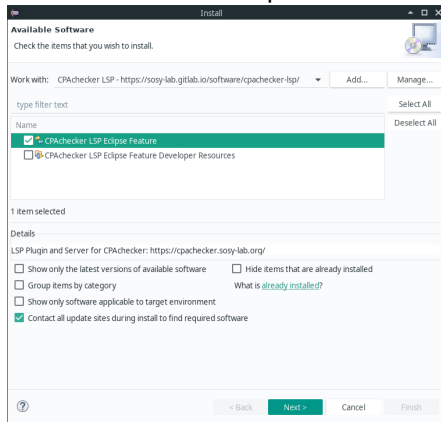
Evaluation: Results

Ideas for Improvement:

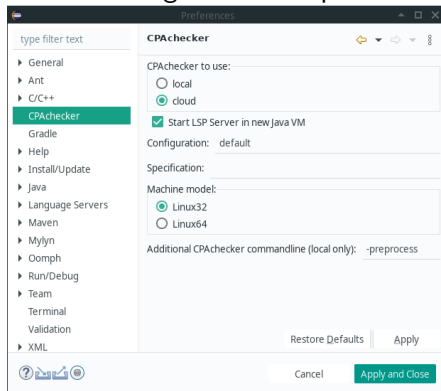
- improvement of error marker position ✓
- more configuration options ✓
- manual starting of verification task ✗
- per file configuration ✗
- better presentation of witnesses ✗

CPAchecker LSP: Installation and Configuration

Installation via update site



Settings inside Eclipse



CPAchecker LSP: Demonstration

The screenshot shows an IDE window titled "example.c" containing the following C code:

```
1 int main() {
2     int i = 0;
3     int a = 0;
4
5     while (1) {
6         if (i == 20) {
7             goto LOOPEND;
8         } else {
9             i++;
10            a++;
11        }
12        goto ERROR;
13        if (i != a) {
14            goto ERROR;
15        }
16    }
```

Below the code editor, the "Problems" tab is active, displaying the following log messages from the CPAchecker Language Server:

```
CPAchecker Language Server (org.eclipse.lsp4e.LanguageServerWrapper@7caf0725)
[Log] [2020-07-13 21:10:10] [INFO] ] Verification started
[Log] [2020-07-13 21:10:12] [INFO] ] Uploading File to VerifierCloud
[Log] [2020-07-13 21:10:12] [INFO] ] Submitting run configuration to VerifierCloud
[Log] [2020-07-13 21:10:14] [INFO] ] Waiting for verification run completion
[Log] [2020-07-13 21:10:18] [INFO] ] Downloading results to temporary files
[Log] [2020-07-13 21:10:18] [INFO] ] Unpacking results
[Log] [2020-07-13 21:10:18] [INFO] ] Verification result: false
[Log] [2020-07-13 21:10:18] [SEVERE] ] Property Violation: error label in line 12
```


Future Work

- Language clients for more IDEs
- Better/more configuration options
- Better presentation of results
- Interactive witnesses using the Debug Adapter Protocol

Conclusion

Fulfilled goals:

- implement Language Server Protocol (LSP) server for CPAchecker: ✓
- allow CPAchecker configuration for basic use: ✓
- allow verification via CPAchecker locally: ✓
- allow verification via VerifierCloud: ✓
- visualize results in Eclipse via LSP client plugin: ✓

Additionally:

- easy installation
- easy to bring to other IDEs

Sources

<https://langserver.org/>

<https://www.cprover.org/eclipse-plugin/>

<https://www.fit.vutbr.cz/research/groups/verifit/tools/predator/>

<https://monteverdi.informatik.uni-freiburg.de/tomcat/Website/>

<https://microsoft.github.io/language-server-protocol/overviews/lsp/img/language-server-sequence.png>

<https://projects.eclipse.org/projects/technology.lsp4e>

<https://www.eclipse.org/ide/>

<https://cpachecker.sosy-lab.org/>