Clone Detection in IEC 61499 using Metainformation

Elene Kutsia

BACKGROUND

INTRODUCTION

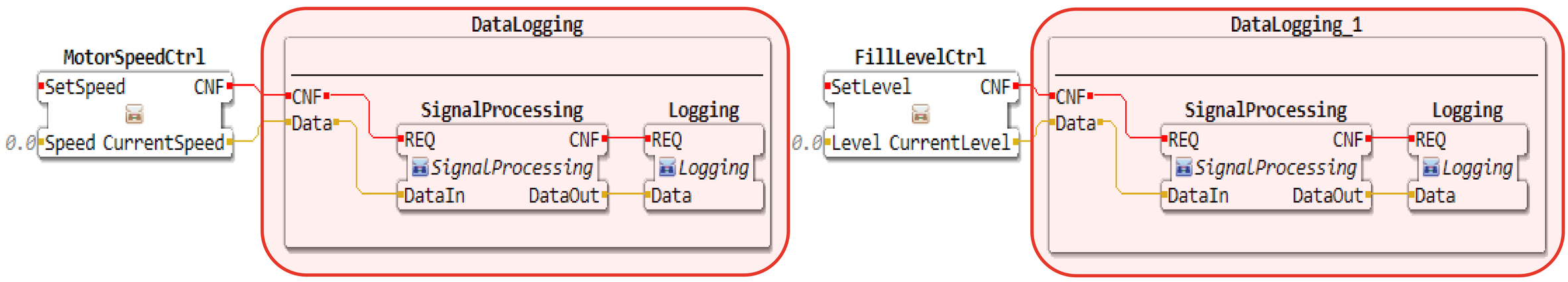
Clone-and-own reuse during software development leads to duplicated code, which can propagate bugs, bloat code and increase overall maintenance effort. Our goal is to **detect type 1 and type 2 clones in the context of IEC 61499-based control software applications**.

EXISTING DETECTION APPROACHES

Many different detection approaches have been proposed: textual, metric and Abstract Syntax Tree comparisons. Most have been applied to text-based code. However, **for IEC 61499**, a **domain-specific visual programming language**, **there is a lack** of research wrt clone detection.

PROPOSED SOLUTION & ALGORITHM

We detect exact clones by **extracting meta information from IEC 61499 applications**. Our algorithm calculates a hash value of each Function Block (FB) inside a SubApplication (SubApp). This hash value is summed up and compared against the hash of other SubApps. If the hashes end up being identical, a clone has been detected.



An example of two identical SubApps (DataLogging and DataLogging_1) being detected as clones.
MotorSpeedCtrl and FillLevelCtrl are each SubApps taht contain an FB Network inside.

```
1: function hashMatch(subAppList, subAppMap)
 2: for all subApps in subAppList do
  3: key ← getSubAppHash(subApp)
  4: if subAppMap contains key then
   5: add SubApp to subAppMap
  6: else
   7: cloneList ← new List
   8: cloneList ← subApp
   9: subAppMap ← (key, cloneList)
  10: end if
 11: end for
12: end function
```

The basic algorithm that goes through all untyped SubApps and calculates their hash.

```
1: function getSubAppHash(subApp)
 2: allFBs ← new List
 3: if subAppNetwork is not empty then
  4: add all network elements to allFBs
 5: end if
 6: subAppMap ← allConnectedPins
 7: subAppHashMap ← allNodeHashes
 8: subAppHash ← 0
 9: for all entries in subAppHashMap do
  10: subAppHash ← Objects.hash(entries)
 11: end for
12: end function
```

The hash code for each SubApp, generated by Objects.hash, is summed iteratively.

CONCLUSION & FUTURE WORK

We evaluated our clone detection prototype with several projects provided by our industry partner, a plant building company in the metallurgical industries, and also on academic example projects. It **successfully detected** clones of type 1 and 2. In future work, we will focus on extending the tool to also detect type 3 clones (clones with added or deleted statements).

JKU LINZ INSTITUTE OF TECHNOLOGY

Elene Kutsia
elene.kutsia@jku.at
Christian Doppler Lab VaSiCS, LIT | CPS Lab