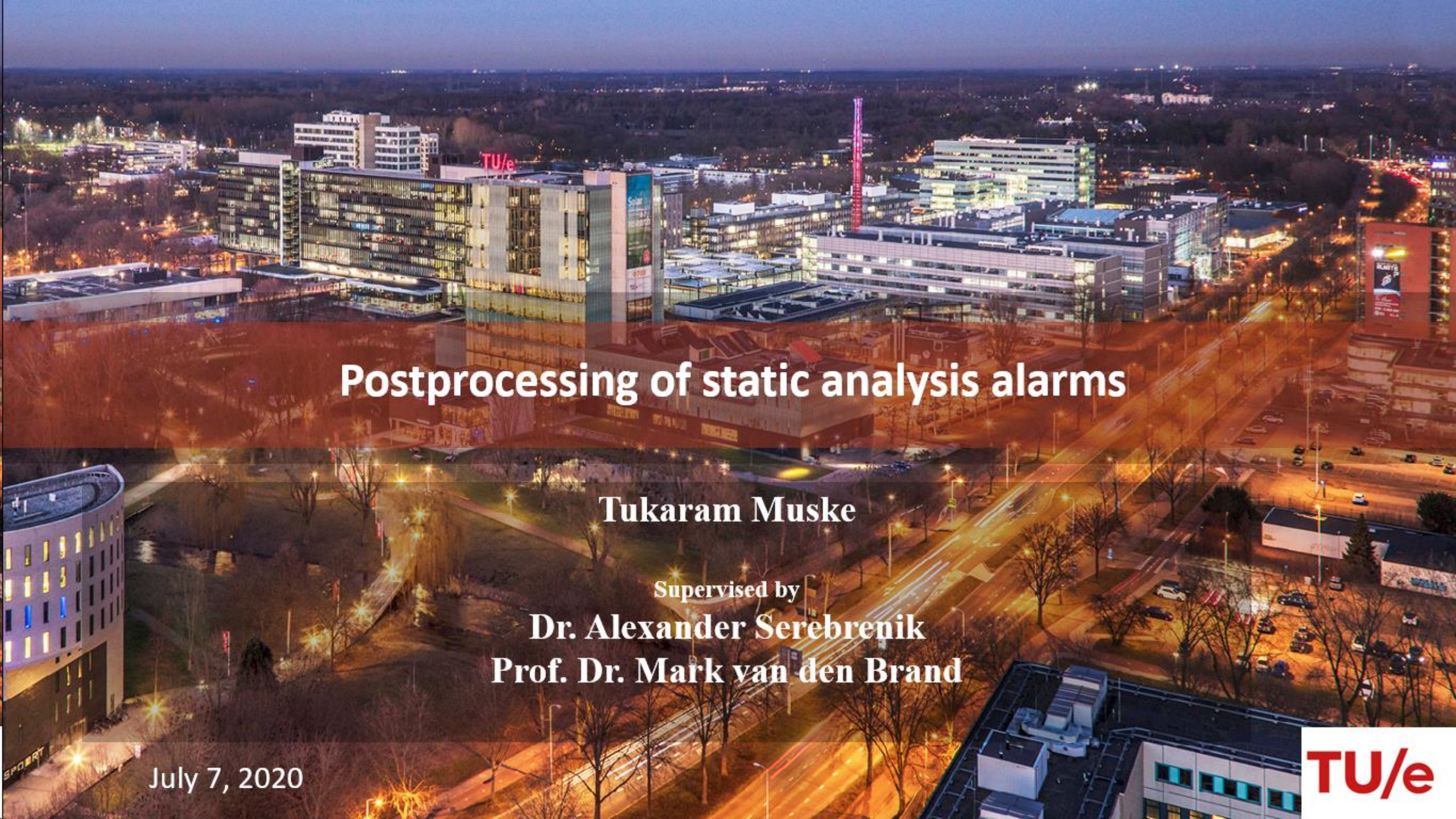


Postprocessing of static analysis alarms

Tukaram Muske

Supervised by

Dr. Alexander Serebrenik

Prof. Dr. Mark van den Brand

July 7, 2020

TU/e

Static Analysis

The prevalence of Software systems






Detection of bugs in software systems

Software Testing

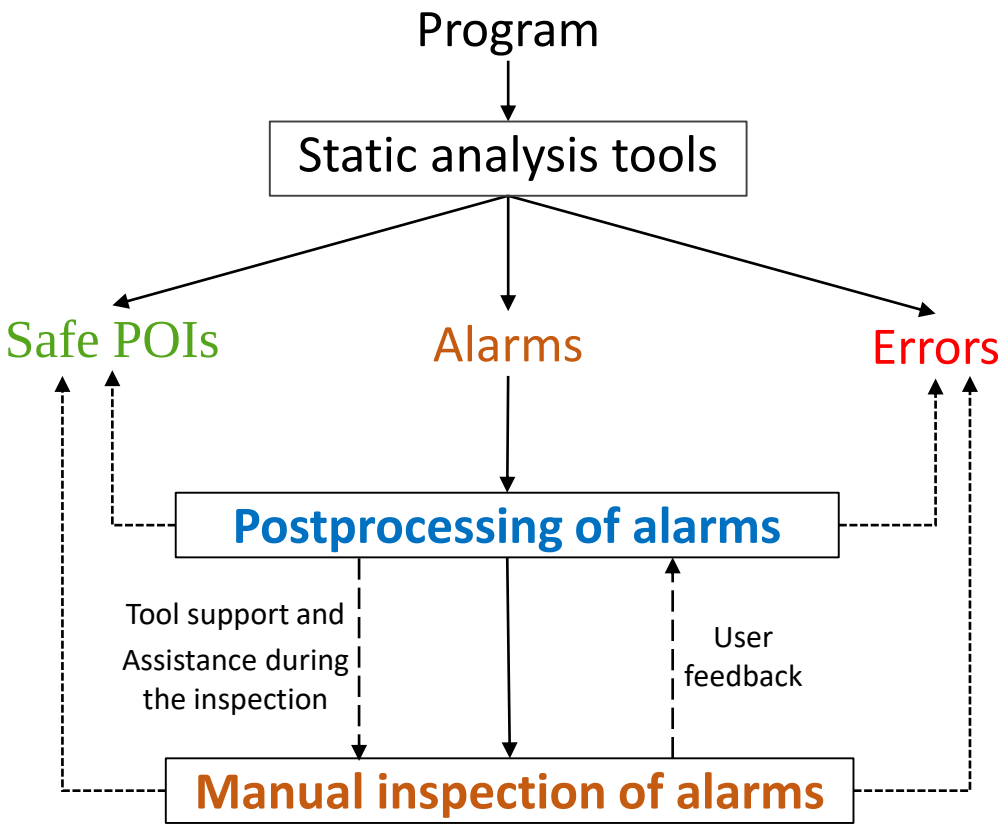
- Detects bugs by executing the code
- Can be used to show the presence of bugs, but never to show their absence[1]

Static Analysis

- Analyzes the code without executing it
- Useful for both: finding of bugs as well as proving their absence

[1] Edsger W. Dijkstra. Structured programming. In Classics in Software Engineering, pages 41–48. Yourdon Press, 1979

The Problem



❖ The problem of large number of alarms

- 40 alarms per every thousand lines of code
- 50 false positives for every accurately reported error
- 35% to 91% of reported alarms are false positives
- Manual inspection of an alarm takes 3 to 8 minutes

Static analysis tools are useful, however, they are not commonly used in practice!

Literature Review

Literature review of approaches and techniques for postprocessing of alarms was missing.

1. Literature review

- Clustering techniques fail to group a high percentage of similar alarms.
- Postprocessing alarms generated on partitioned-code suffers from redundancy.
- AFPE techniques take considerable amount of time.
- Code changes are not taken into account to postprocess delta alarms.

This work has been published in SCAM 2016.

Repositioning of Alarms

Literature review of approaches and techniques for postprocessing of alarms was missing.

1. Literature review

Clustering techniques fail to group a high percentage of similar alarms.

2. Repositioning of alarms

Repositioning of alarms

```
1.   void foo(int x){
2.     if(…) x = input();
3.     //assert(x!=0);// Repositioned Alarm
4.     if(…)
5.       t1 = 1 / x;  //Alarm
6.     else
7.       t2 = 3 / x;  //Alarm
8.   }
```

Repositioning	Reduction up to	Median reduction	Average reduction
Original	20%	7.25%	6.47%

The work based on repositioning of alarms has been published at ISSTA 2018.

Repositioning of Alarms

Literature review of approaches and techniques for postprocessing of alarms was missing.

1. Literature review

Clustering techniques fail to group a high percentage of similar alarms.

2. Repositioning of alarms

The reduction obtained is limited.
(Reason – conservative assumption about the controlling conditions of alarms)

3. Improved Repositioning

Repositioning of alarms

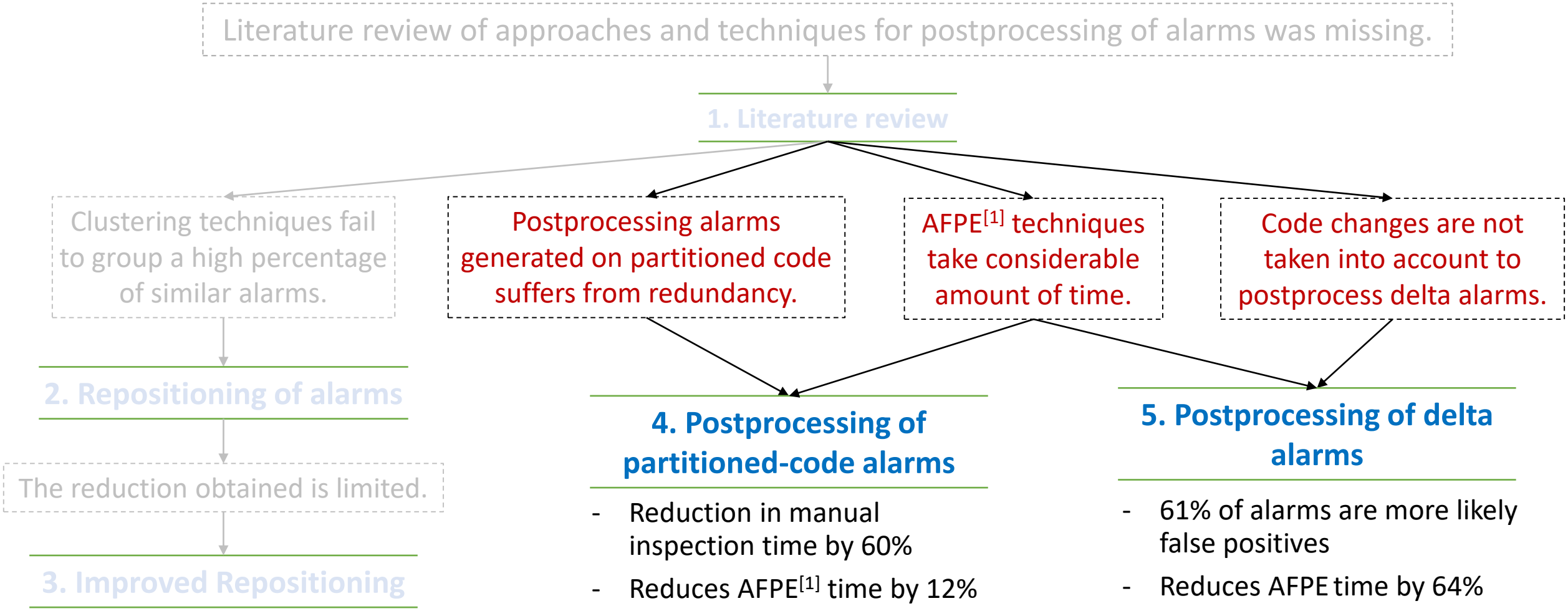
```
1.   void foo(int x){
2.     if(…) x = input();
3.     //assert(x!=0);// Repositioned Alarm
4.     if(…)
5.       t1 = 1 / x;  //Alarm
6.     else
7.       t2 = 3 / x;  //Alarm
8.   }
```

Repositioning	Reduction up to	Median reduction	Average reduction
Original	20%	7.25%	6.47%
Improved	+36%	+10.48%	+10.5%

The work based on repositioning of alarms has been published in ISSTA 2018.

The work based on improved repositioning is published in APLAS 2019.

Other Improvements



A part of our work based on postprocessing of alarms generated on partitioned-code is published in ICSME 2014.

[1] AFPE = Automated false positives elimination

What's the Summary?

Ensuring software systems are bug-free is becoming more important and challenging than before.

Existing postprocessing techniques can be improved by considering the type of applications and analyses.

Plentitude of approaches and techniques exist, their effective combinations need to be studied.