Developing Scientific Applications Using Generative Programming

Ritu Arora

Purushotham Bangalore

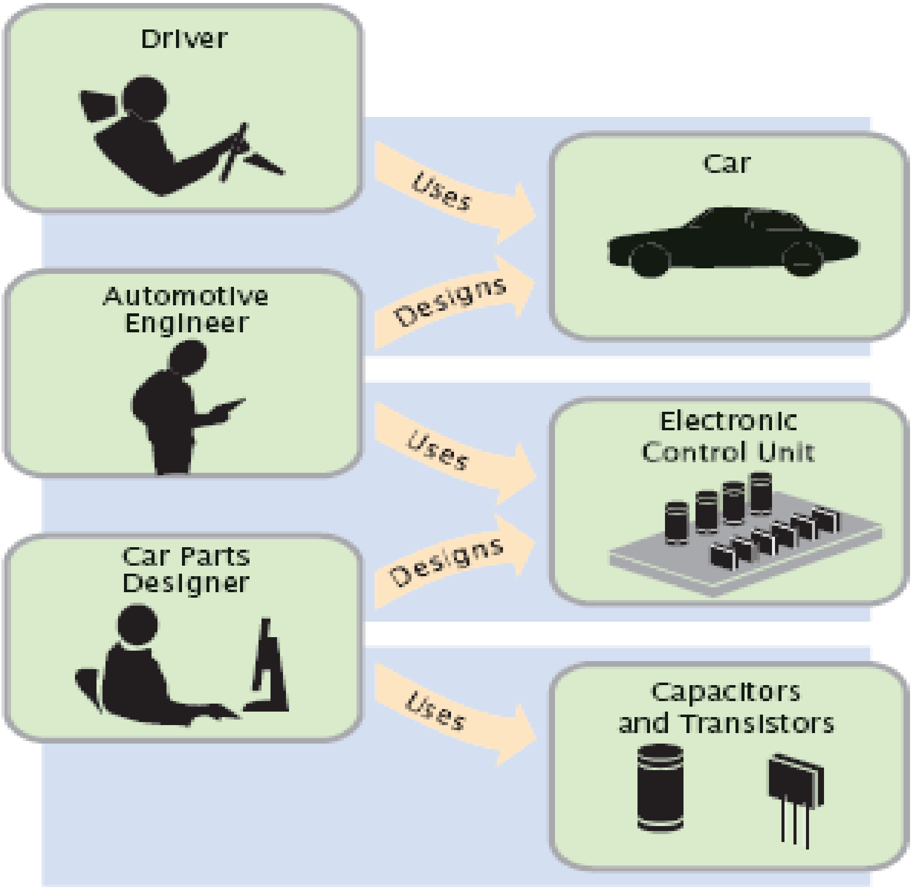
Marjan Mernik

Basic Concepts Related to the Talk

- Abstraction
- Domain-Specific Languages (DSLs)
- Program Transformation
- Crosscutting Concern
- Checkpointing

Abstraction

1. A representation that captures only essential aspects of something, reducing the complexity apparent to the abstraction's user
2. **Hides** details



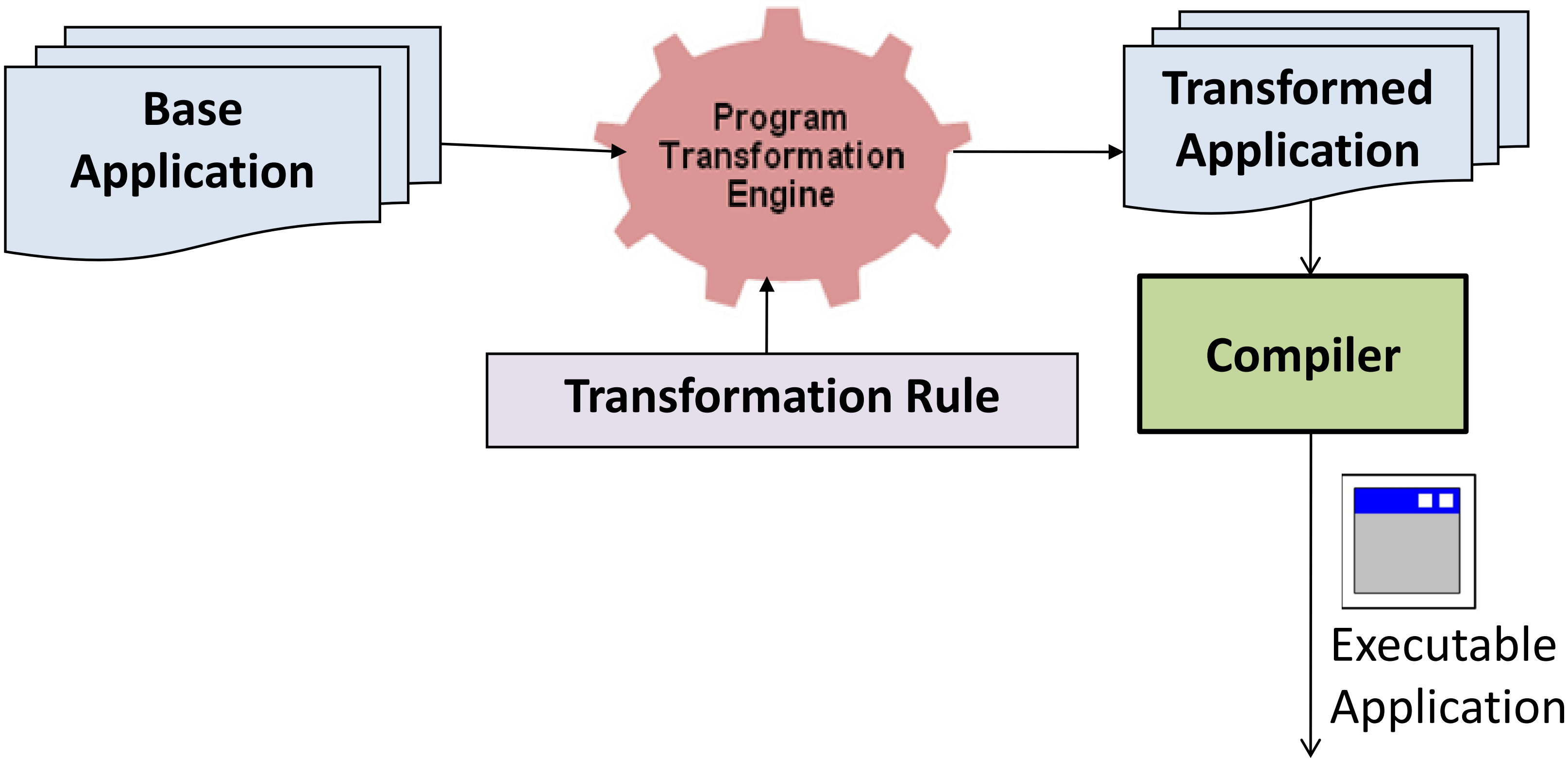
Levels of abstraction in automotive design

Domain-Specific Language (DSL)

- DSLs are
  - high-level languages with a very narrow domain and a very high-level of abstraction
  - less comprehensive than general-purpose languages
  - more expressive in their domain

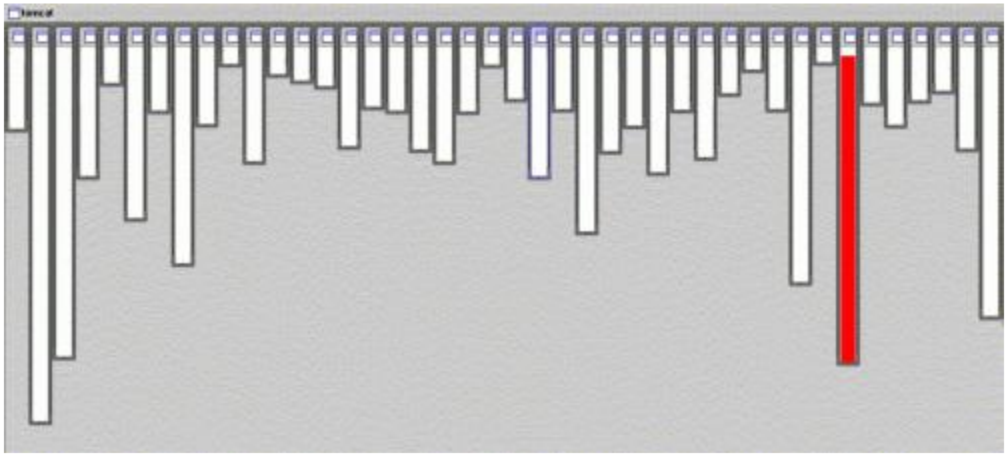
```
begin
    left
    up
    right
    down
end
```

Program Transformation

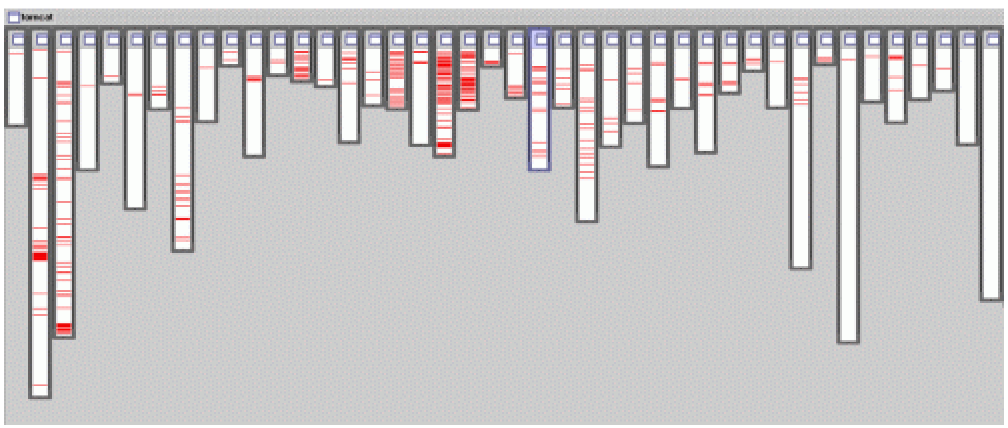


Transformed Program = f_{weave} (Base Code, New Requirement)

Crosscutting Concern



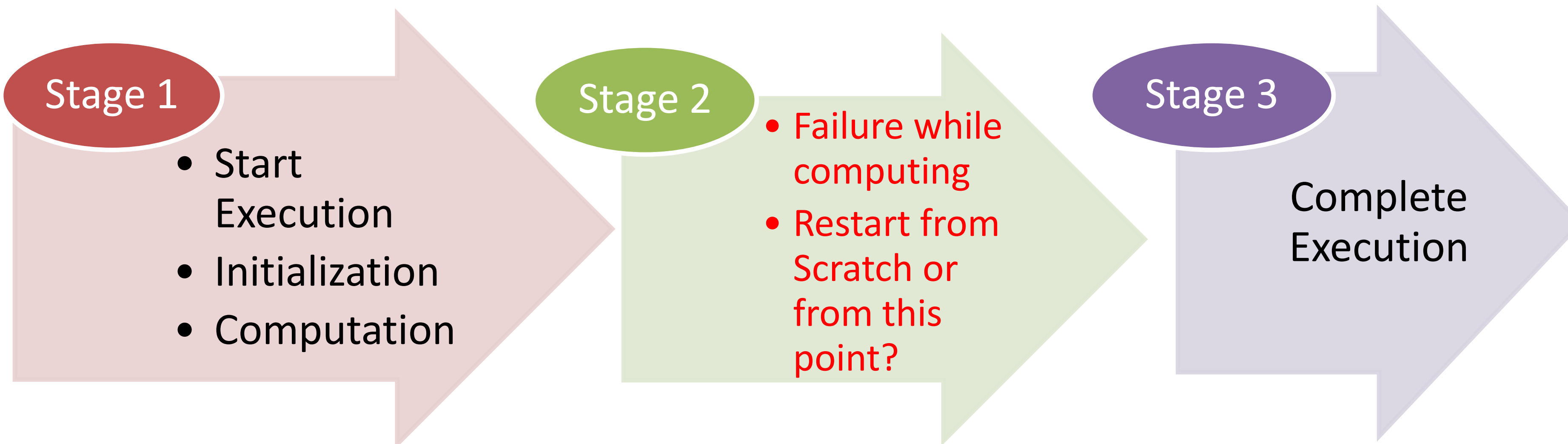
- XML Parsing in Apache Tomcat   Server



- Logging in Apache Tomcat Server

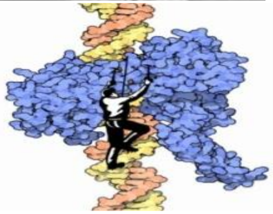
Source: http://www.parc.com/research/projects/aspectj/

Checkpointing



- Checkpointing: System-level or Application-level
- Application-Level Checkpointing (ALC)
  - Checkpointing mechanism is directly inserted into the application
  - Critical Application variables & data structures are saved



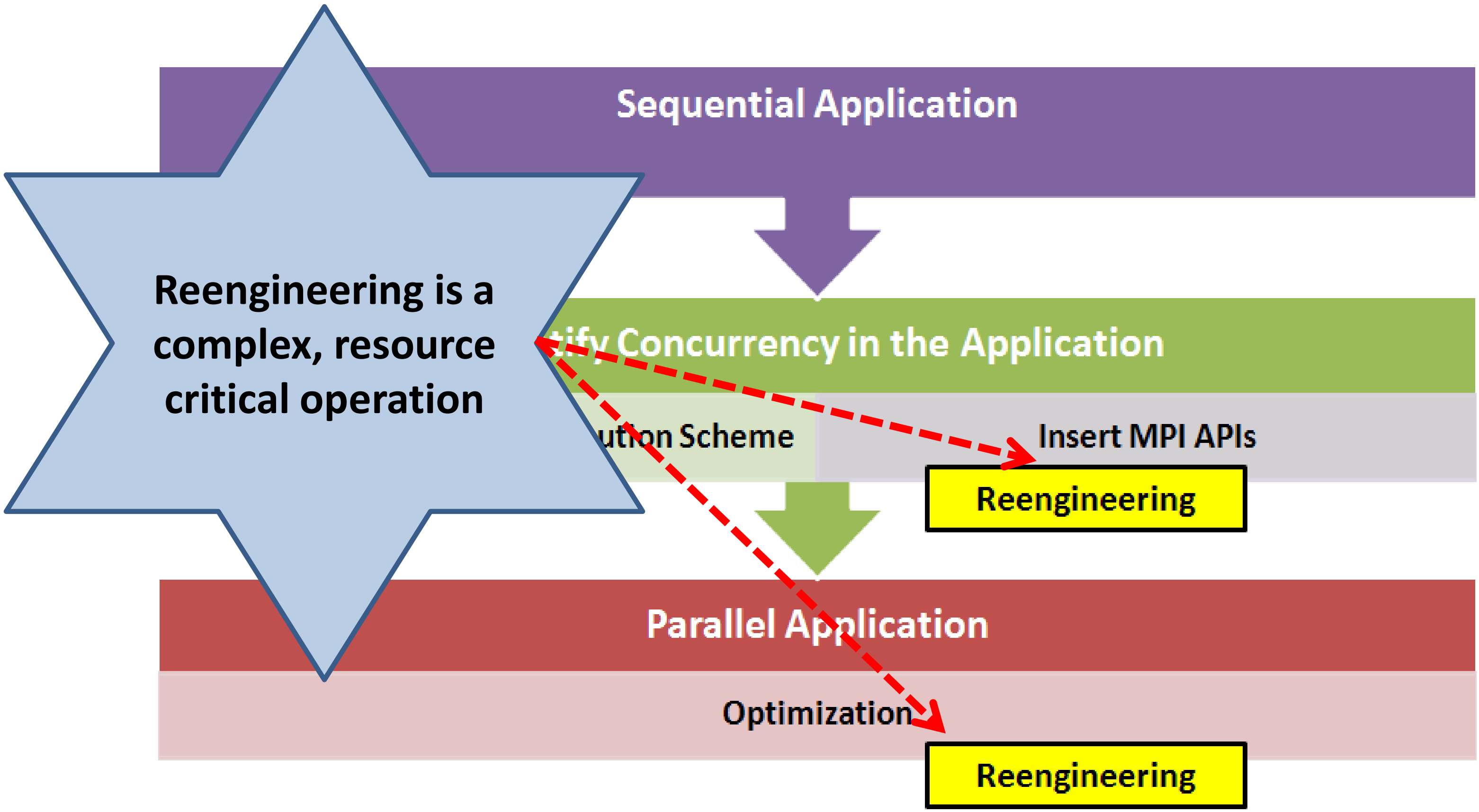
High Performance Computing

- Who are the end-users?
- End-Users are increasingly relying upon high-performance clusters that amplify computing power
- Multi-Core & multi-processor architectures
  - Difficult to program
  - Time to production long

Domain experts feel burdened

Multi-Core Programming

- Most often done via explicit parallelization using APIs like MPI (offers speed & portability)



Challenges in Multi-Core & Multi-Processor Programming

1. Provides a poor layer of abstraction
2. Development and debugging cost is usually high
3. Parallelization often becomes a reengineering activity
4. Necessitates intrusive changes to the sequential application (high maintenance)
5. The code becomes complex, difficult to maintain, and difficult to reuse
6. No well-established rules, guidelines or patterns for designing a parallel application
7. Data decomposition, mapping of computational tasks to processor and synchronization is all explicit

Desired Improvements

- Raising the level of abstraction of the parallel programming
- Semi-automation of the process of non-intrusive synthesis of parallel programs
- Separation of sequential and parallel code constructs
- Promote reusability and modularity in HPC applications

Proof-of-Concept Using ALC

- Problems
  - ALC is a crosscutting concern
  - Invasive reengineering of legacy applications is involved
  - Repeated code constructs across applications
  - Coupling between problem and solution space
- Also Observed
  - A pattern for application-level Checkpointing and Restart (CaR)
    - What is consistent across various application?
    - What varies from application to application?
  - The consistent parts of the code for CaR can be abstracted in high-level language constructs to
    - promote code reusability & correctness
    - increase expressiveness
  - Checkpointing involves overheads => can be undesirable at times

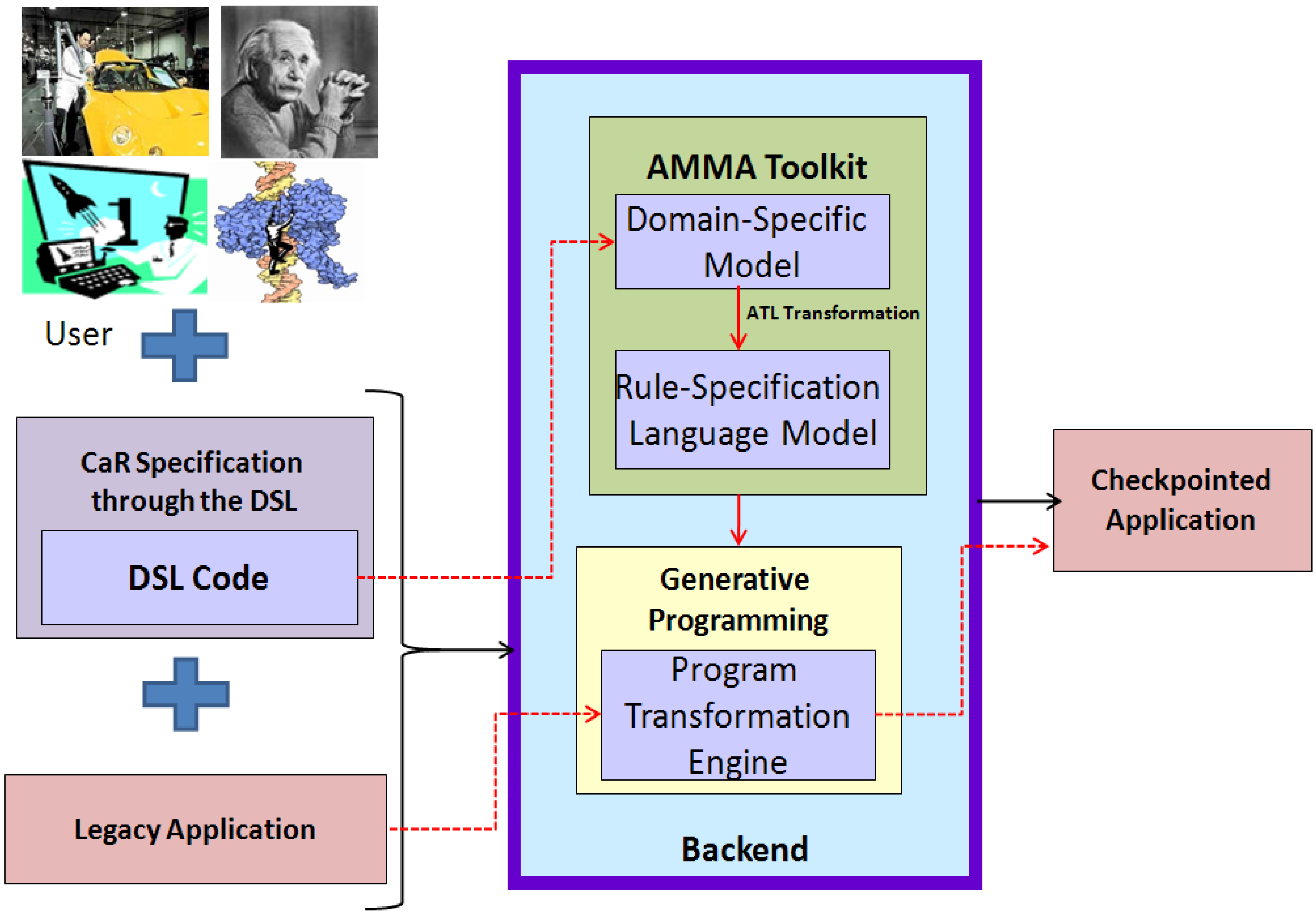
Research Goals

- Abstract out the common (reusable) code
- ALC mechanism should be implemented non-intrusively
- Separation of CaR specifications from its implementation
- The development time and cost should be reduced
- The checkpointing feature should exist as a pluggable module

The Solution - At A Very High-Level

- Develop a high-level language for specifying the CaR mechanism (DSL)
- Develop code components
- Generate the CaR code semi-automatically from the end-user specifications using mappings and code components (Program Transformation Engine and a mapping language)
- Insert the generated code into the base application (Program Transformation)

Implementation Approach



Base Code Snippet

```
for(i=0;i<numGenerations;i++){
  printf("Gen: %d ", i);
  pickchroms(fitness,popcurrent,popnext);
  mutation(popnext,popcurrent);
  equate(popcurrent, popnext);
  evaluatePop(popcurrent,mydata,fitness);
  printGenFit(popcurrent,fitness,(int)time);
}
```

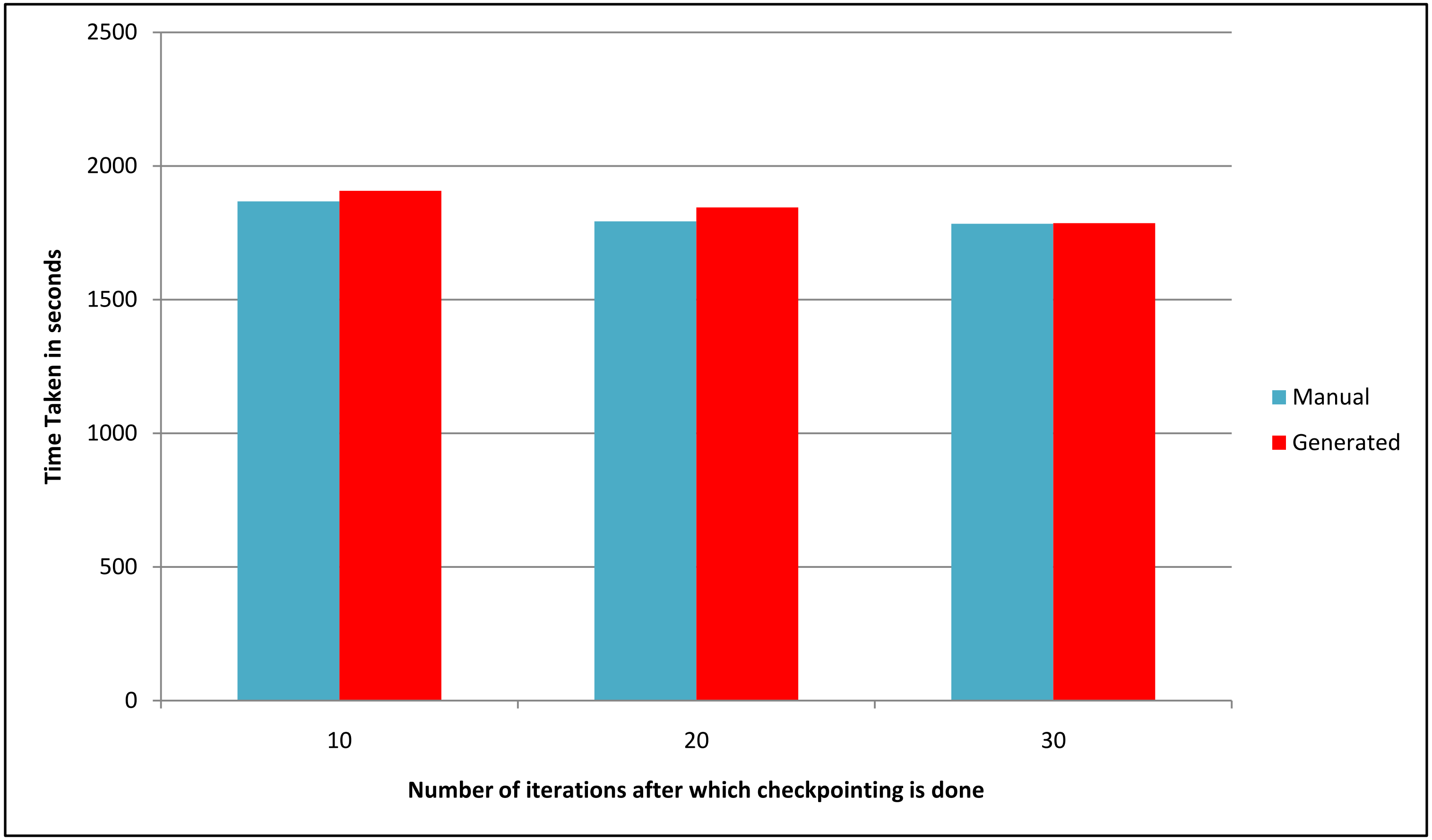
Sample DSL code

```
beginCheckpointing:
after execution("printGenFit")
&& (frequency = 10) && (loopVar = "i" )
{
   SaveInt(time,"restartTime")
   SaveIntArray2D(popcurrent, numChrom, numCentroid,
                                 "restartPopCurrent")
}
```

```
beginInitialization: around execution ("fOpenClose")
{
      ReadIntVarFromFile (time, "restartTime")
      ReadIntArray2DFromFile  (popcurrent, numChrom,
                    numCentroid, "restartPopCurrent")
      |
      ReadIntArray2DFromFile(popcurrent, numChrom,
                              numCentroid, "initial")
}
```

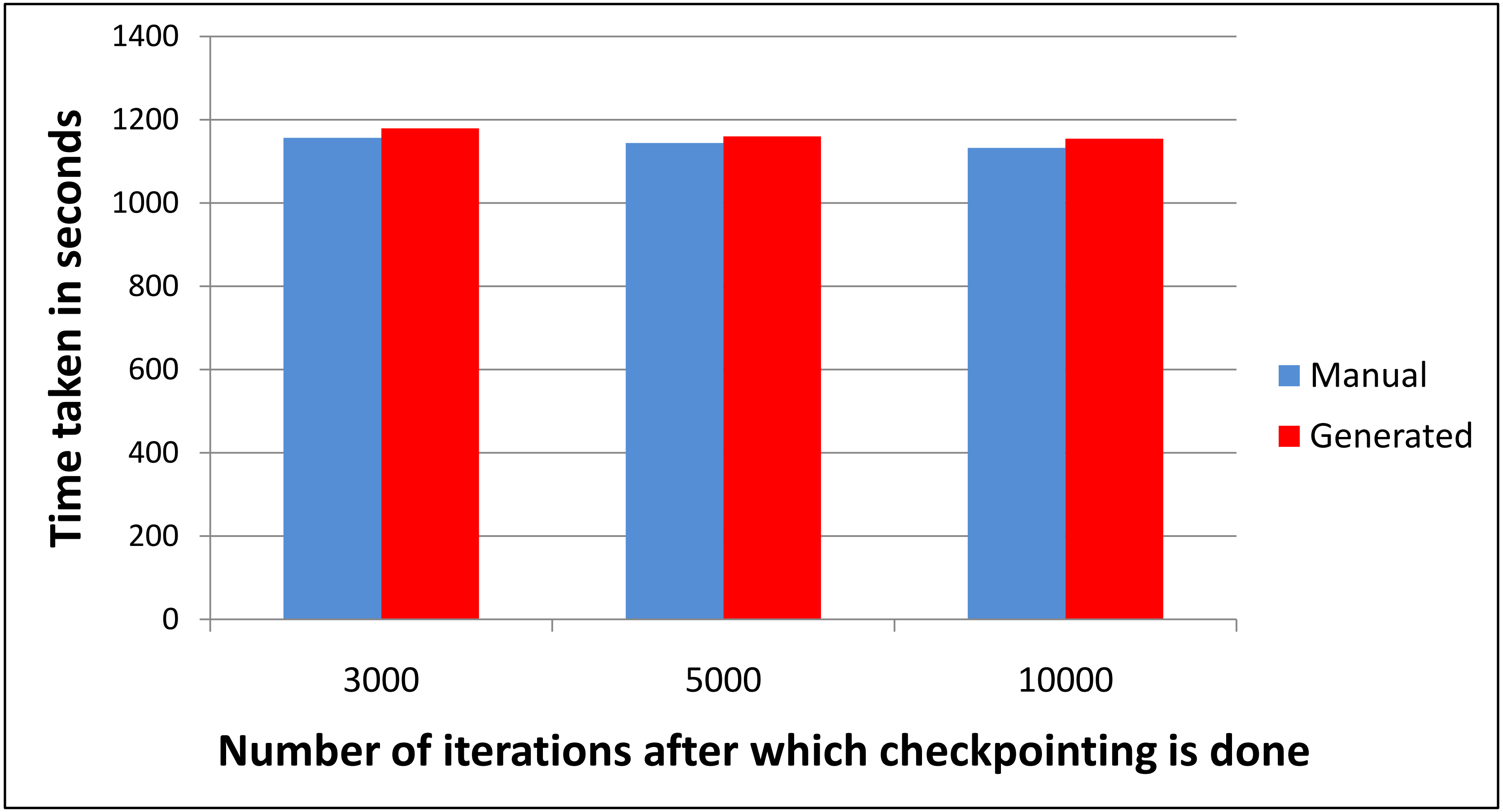
```
if (i % 10 == 0){
  newInputFile = fopen("restartPopCurrent.txt",
  "w")
  storeVar = fopen("restartTime.txt", "w");
  fprintf(storeVar, "%d  ", time);
  for (k= 0; k < numChrom; k++){
   for (j = 0; j < numCentroid; j++){
    fprintf(newInputFile,"%d", popcurrent[k][j]);
   }
   fprintf(newInputFile, "\n");
   }
    fclose(newInputFile);
    fclose(storeVar);
 }
```

Sequential Genetic Algorithm for Content-Based Image Retrieval



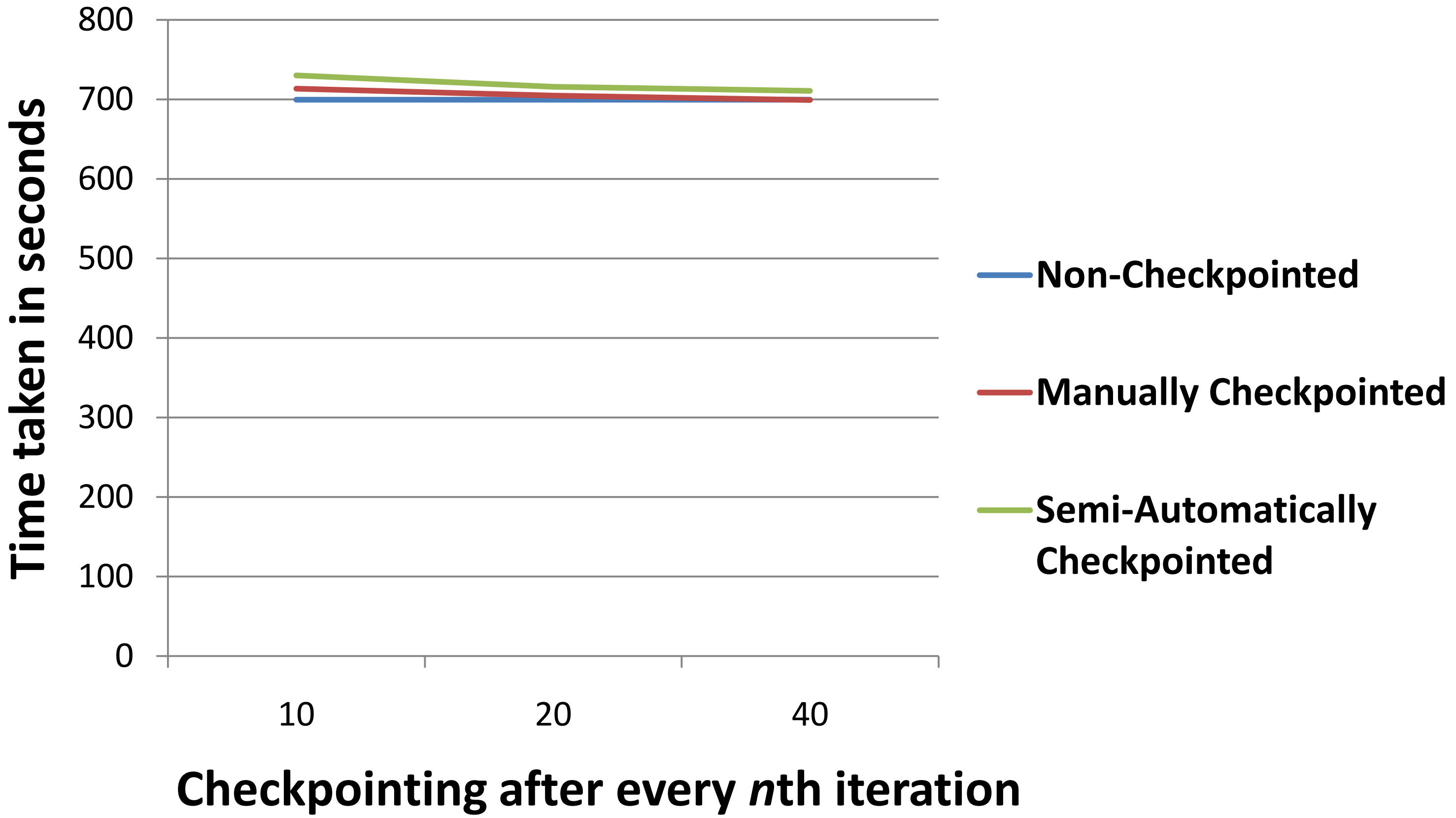
The GA was run for 100 generations on 82556 image segments

Sequential Poisson Solver



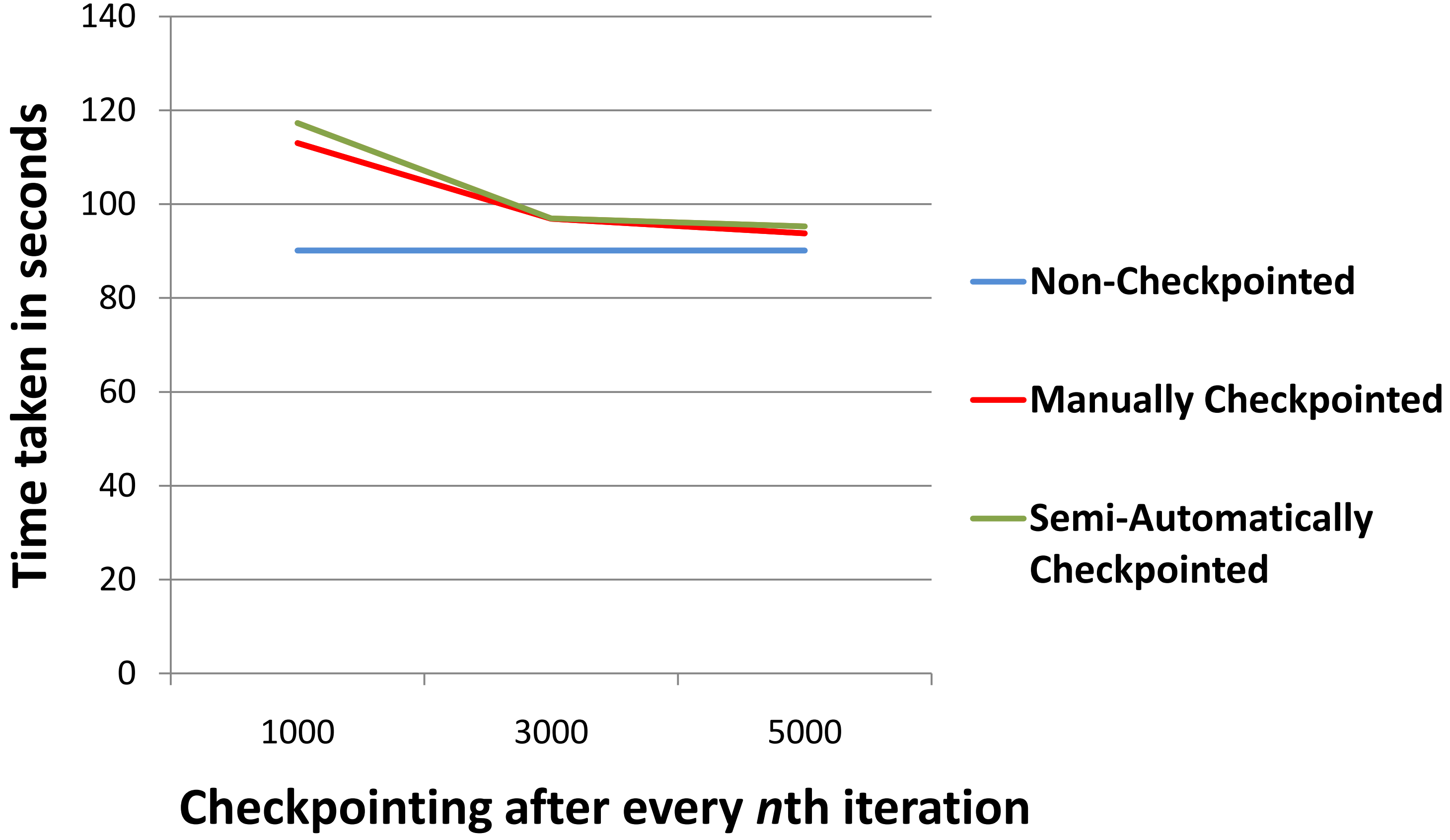
The matrices were of the size 10,000 X 10,000 and the program was run for 50,000 iterations. The solution converged after 41218 iterations.

Parallel Genetic Algorithm

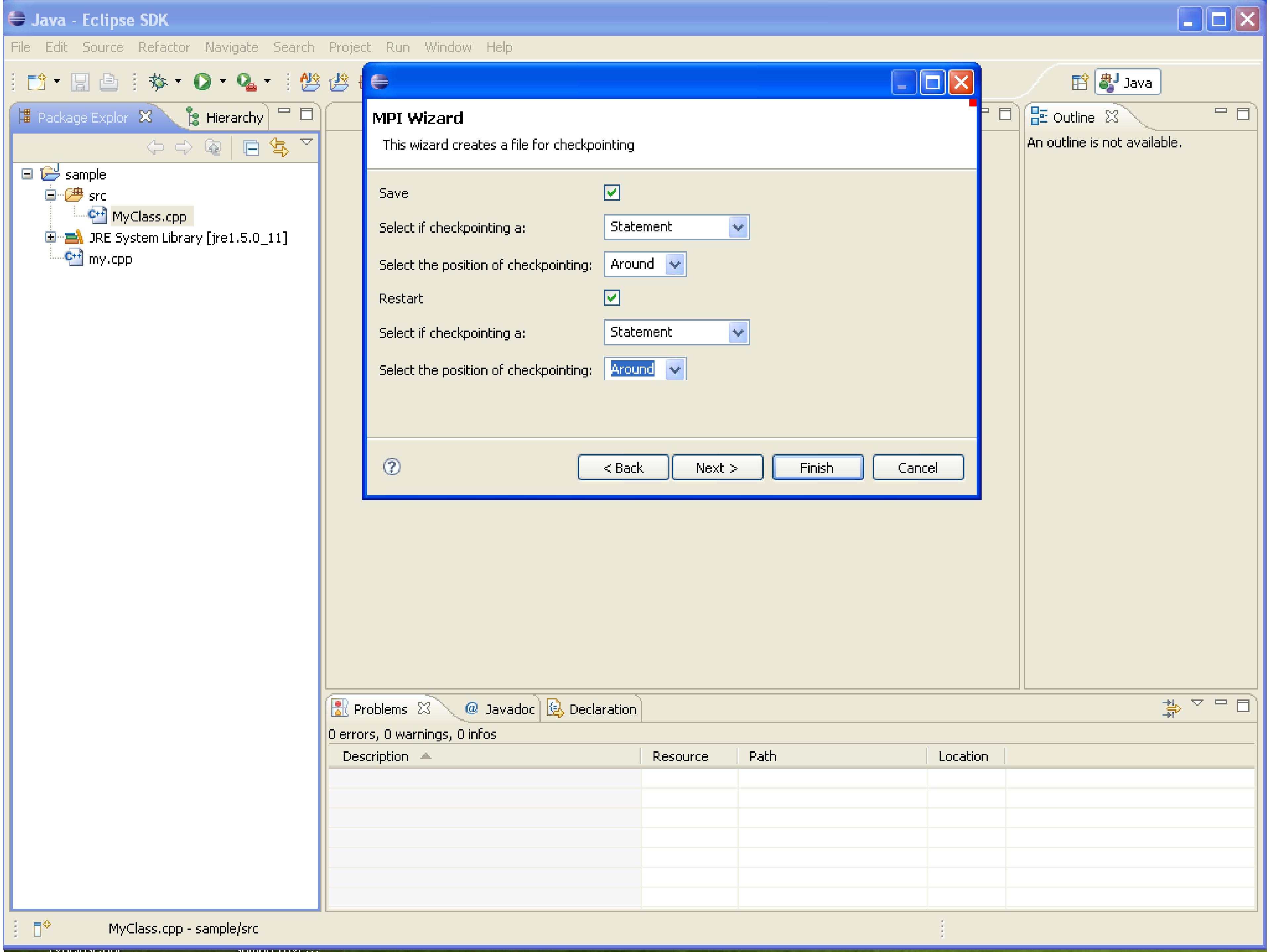


The PGA was run for 1000 generations on 82556 image segments and 50 processors

Parallel Poisson Solver



The matrices were of the size 10,000 X 10,000 and the program was run for 50,000 iterations on 40 processors. The solution converged after 41218 iterations.


Java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Package Explor
Hierarchy
sample
src
MyClass.cpp
JRE System Library [jre1.5.0_11]
my.cpp
MPI Wizard
This wizard creates a file for checkpointing
Save
Select if checkpointing a:
Statement
Select the position of checkpointing:
Around
Restart
Select if checkpointing a:
Statement
Select the position of checkpointing:
Around
< Back
Next >
Finish
Cancel
Outline
An outline is not available.
Problems
Javadoc
Declaration
0 errors, 0 warnings, 0 infos
Description
Resource
Path
Location
MyClass.cpp - sample/src

Thanks!

Questions ?

Email: {ritu, mernik, puri}@cis.uab.edu

http://www.cis.uab.edu/ccl/index.php/Domain-Specific_Language_for_Checkpointing

Acknowledgement

Dr. Purushotham Bangalore
Dr. Marjan Mernik
Dr. Suman Roychoudhury