Modeling Multigrid Algorithms for Variational Imaging

Isabel Dietrich, Harald Koestler, Reinhard German, Ulrich Ruede

Computer Networks and Communication Systems
University of Erlangen-Nürnberg, Germany

April 09, 2010
What do we do?

- Model algorithms in UML
- Transform to image processing application
- Case study: image denoising
Outline

1. Software Packages
   - Syntony
   - Image Processing Framework

2. Modeling Approach

3. Model Transformation

4. Results

5. Conclusion
UML-Based Simulation with Syntony

Testing Profile

MARTE Profile

Casual

Operation(param=trigger());
fork();
(Class2.create().sendTo(self().Port1));
"opaque";
join();

generate

generate

generate

Test Cases

Discrete-Event Simulation

Multigrid Algorithms

Isabel Dietrich
University of Erlangen

Modeling Multigrid Algorithms for Variational Imaging
Mathematical modeling

Different models for different image processing applications

Example: denoising with homogeneous diffusion

Minimize $\mathcal{E}[u] = \int_{\Omega} |u^0 - u|^2 + \alpha |\nabla u|^2 \, dx$

- $\mathcal{E}[u]$: energy functional
- $u^0$: observed image
- $u$: denoised image
- $\alpha$: regularization parameter
- $\Omega$: image domain

$\Rightarrow$ Solve Partial Differential Equation (PDE)
Multigrid Algorithms

- General approach to solve PDEs efficiently
- Two principles:
  - Smoothing property: use iterative methods → smooth high frequency error
  - Coarse Grid principle: use information from coarse grids → approximate low frequency error
- Combine principles in recursive algorithm
Multigrid Algorithms

- Basic steps:
  - Exact solution: □
  - Interpolation of solution: ▲
  - Smoothing: ●
  - Restriction of Residual: ◊
  - Interpolation of the error and Correction of the solution: ▼

Image Processing Framework

Isabel Dietrich
University of Erlangen
Modeling Multigrid Algorithms for Variational Imaging
Variational Image Processing Framework

- Many mathematical models for different applications
- Many implementations of efficient multigrid solvers
- Hardware-specific implementations (GPU, Cell processors,...)

Pros

- High computational performance

Cons

- Long compile times
- Compile times grow with number of applications
- Extraction of single applications difficult
- Extension difficult (good knowledge of code needed)
Improved Software Development Process

**UML Model**
- class and activity diagrams
- language/platform independent
- graphical interface
- high-level modelling

**Framework (C++)**
- optimized routines in modular blocks
- basic library: I/O, data types, communication
- low-level modelling

**Syntony (Code Generator)**
- reduced templates and ifdefs in code
- shorter compile times
- application and architecture specific

**GUI**
- control input parameters
- graphical output

Isabel Dietrich
University of Erlangen

Modeling Multigrid Algorithms for Variational Imaging
Modeling Approach

1. Represent entire framework in UML classes (can be done automatically by some UML tools)
2. Model high-level algorithms in activity diagrams
3. Specify modules/methods to copy from existing framework
Modeling: Classes

```
 Solver ( )
 solve ( sd : SolverData [1..*], sp : SolverParameters [1..*] ) : MYREALTYPE

 FMGSolver ( sd : SolverData [1..*], sp : SolverParameters [1..*], op : Operator [1..*] )
 Cycle ( sd : SolverData [1..*], sp : SolverParameters [1..*], lev : Integer )
 init ( sd : SolverData [1..*], sp : SolverParameters [1..*], image : Array_Myrealtype )
 FMGSolver ( sd : SolverData [1..*], sp : SolverParameters [1..*], op : Operator [1..*] )
 VCycle ( sd : SolverData [1..*], sp : SolverParameters [1..*], lev : Integer )
 FASVCycle ( sd : SolverData [1..*], sp : SolverParameters [1..*], lev : Integer )
 setZero ( lev : Integer )
 store_res_old ( )
 print_cycle_info ( )
```
Modeling: Inclusion of Existing Existing Modules

- **Stereotypes**: extend UML semantics
  - «NumericModule», «NumericOperation»:
    - Apply to classes/operations
    - Code is not generated, but copied from existing sources
    - Name and path of existing sources may be specified
  - «NumericLoop»:
    - Apply to actions
    - Action is executed in loop with specified number of repetitions
Modeling: Activities

Isabel Dietrich
University of Erlangen
Modeling Multigrid Algorithms for Variational Imaging
Model Transformation

- Controlled from Eclipse-based GUI
- Classes: straightforward C++ code
- Activities: token-based semantics (control/object flow)
- Details: in the paper
Results: Functionality

Original image – smoothed image
### Results: Code Generation Overhead

- Evaluate code generation + compilation times
- Compare to compilation time for original framework
- Two models:
  - Homogeneous diffusion (HD)
  - Complex diffusion (CD)

<table>
<thead>
<tr>
<th></th>
<th>Generation</th>
<th>Compilation</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>HD</td>
<td>5.1 s</td>
<td>9.9 s</td>
<td>15 s</td>
</tr>
<tr>
<td>CD</td>
<td>4.9 s</td>
<td>11.3 s</td>
<td>16.2 s</td>
</tr>
<tr>
<td>original framework</td>
<td>-</td>
<td>60 - 120 s</td>
<td>60 - 120 s</td>
</tr>
</tbody>
</table>
Results: Runtime Overhead

- Evaluate for different image sizes:
  - Absolute runtimes
  - Relative overhead compared to original framework
- Two models:
  - Homogeneous diffusion (HD)
  - Complex diffusion (CD)

<table>
<thead>
<tr>
<th>Model</th>
<th>256 × 256</th>
<th>512 × 512</th>
<th>1024 × 1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>HD</td>
<td>0.03 s (2 %)</td>
<td>0.13 s (1 %)</td>
<td>0.44 s (0.5 %)</td>
</tr>
<tr>
<td>CD</td>
<td>0.1 s (1 %)</td>
<td>0.38 s (0.5 %)</td>
<td>1.5 s (0.2 %)</td>
</tr>
</tbody>
</table>
Conclusion

What have we done?

- Model-driven software development process for numerical algorithms
  - High-level behavior: UML activity diagrams
  - Hardware-specific / computationally intensive modules: copy existing code

Future Work

- Model additional applications
- Code generation for different hardware platforms (GPUs)
Thank you for your attention!