

# Design techniques for high performance multi-physics simulations

Andrea Lani and Tiago Quintino \*  
Von Karman Institute for Fluid Dynamics

6th May 2009

## Contents

<b>1</b>	<b>Basics on Computer Science</b>	<b>5</b>
1.1	Object Oriented Programming . . . . .	5
1.1.1	Class . . . . .	5
1.1.2	Object . . . . .	6
1.1.3	Composition . . . . .	6
1.1.4	Inheritance . . . . .	7
1.1.5	Polymorphism . . . . .	7
1.2	Generic Programming . . . . .	8
1.2.1	Templates . . . . .	8
1.3	Design Patterns . . . . .	9
1.3.1	Example: Template Method . . . . .	9
1.4	Component Based Software Engineering . . . . .	10
1.4.1	Components and Frameworks . . . . .	10
<b>2</b>	<b>Computational domain representation</b>	<b>11</b>
2.1	One domain, multiple views . . . . .	11
2.2	One concept per view . . . . .	12
2.2.1	Many concepts, one access . . . . .	13
2.2.2	Topological Region Sets . . . . .	13
2.2.3	Geometric Entities . . . . .	15
2.2.4	Data Storage . . . . .	16
<b>3</b>	<b>Parallel Algorithms</b>	<b>19</b>
3.1	Parallel Mesh Reading . . . . .	19
3.1.1	Reading of the elements data . . . . .	19
3.1.2	Parallel Mesh Decomposition . . . . .	20
3.1.3	Construction of the Overlap Region . . . . .	21

---

\*Insert Acknowledgments

3.1.4	Reading of the Topological Region Sets . . . . .	23
3.1.5	Reading of the Nodes and States . . . . .	24
3.2	Parallel Mesh Writing . . . . .	25
3.2.1	Parallel writing by ranges . . . . .	25
3.3	Performance tests . . . . .	26
3.3.1	Performance of I/O algorithms . . . . .	26
3.3.2	Speedup and parallel efficiency . . . . .	27
<b>4</b>	<b>Method-Command-Strategy Pattern</b>	<b>31</b>
4.1	MCS: Description . . . . .	31
4.1.1	Example of SpaceMethod: FVM_Method . . . . .	33
4.1.2	Example of Collaboration: ConvergenceMethod and LinearSystem-Solver . . . . .	34
<b>5</b>	<b>Meta-Programming</b>	<b>37</b>
5.1	Typesafe and Size-Deducing FET . . . . .	37
5.1.1	Fast Expression Templates . . . . .	37
5.1.2	Typesafe Enumeration Policy . . . . .	38
5.1.3	Size-Deducing FET for Small Arrays . . . . .	39
5.1.4	Applications . . . . .	41
5.1.5	Performance Results . . . . .	45
5.2	Runtime instantiation of static components . . . . .	47
5.2.1	Problem statement . . . . .	47
5.2.2	Using static components . . . . .	48
5.2.3	Solution based on template algorithms . . . . .	50
5.2.4	Automatic instantiation . . . . .	51
5.2.5	Template class loader . . . . .	52
5.2.6	Source repository, Code builder and Compiler . . . . .	52
5.2.7	Template strategy pattern . . . . .	57
5.2.8	Benchmark . . . . .	58
<b>6</b>	<b>Multi-Physics Component Environment</b>	<b>61</b>
6.1	Handling multi-physics . . . . .	61
6.1.1	Application example . . . . .	61
6.1.2	Problem statement . . . . .	61
6.1.3	Goals . . . . .	61
6.2	Data Sockets . . . . .	62
6.2.1	Socket identification . . . . .	64
6.2.2	Application code . . . . .	65
6.2.3	Data Broker . . . . .	66
6.3	Context architecture . . . . .	67
6.3.1	Context Switching . . . . .	69
<b>7</b>	<b>Conclusion</b>	<b>73</b>