A Pattern Language for Parallel Programming

Tim Mattson
timothy.g.mattson@intel.com

Beverly Sanders
sanders@cise.ufl.edu

Berna Massingill
bmassing@cs.trinity.edu

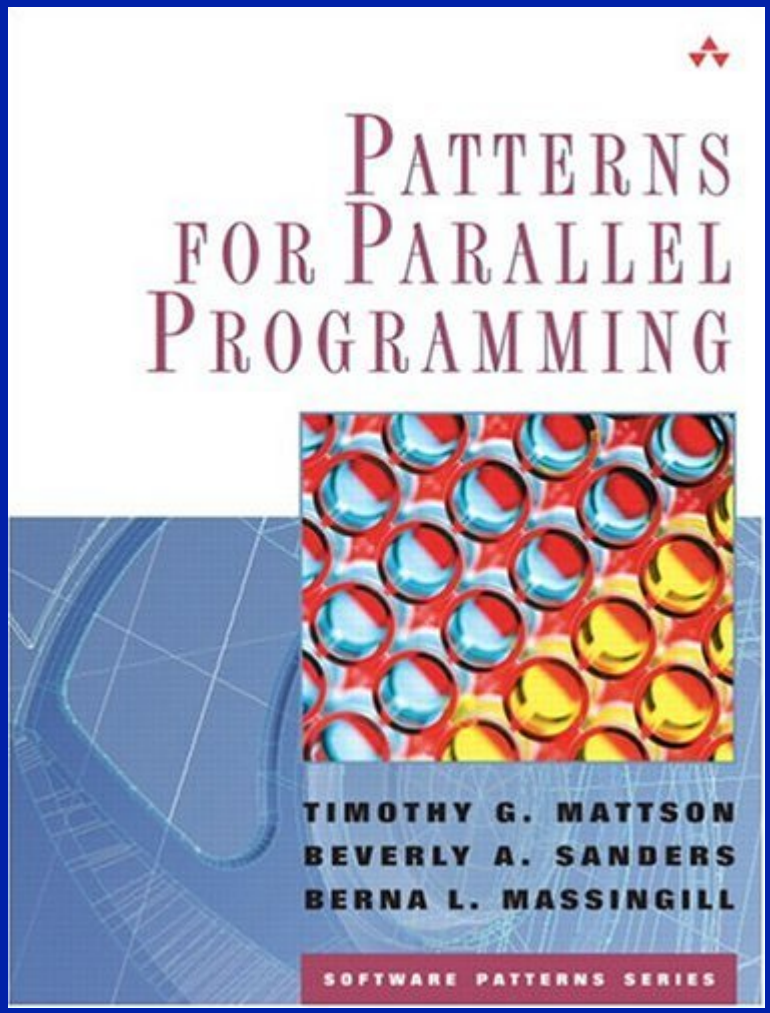
Motivation

- Hardware for parallel computing is “everywhere”: clusters, SMP workstations, NUMA “Big Iron”.
- Software to run on these systems is not.
- Knowledge needed to make effective use of hardware is mostly limited to high-end HPC community.
- How to disseminate this knowledge (to programmers, domain experts) so that “sequential software is rare”?

A Possible Long-Term Solution

- A layered solution stack focused on the algorithm designer, not the hardware:
  - A pattern language for parallel programming.
  - A component-based framework.
  - Low-level portable APIs for parallel computing.
  - Supporting middleware.
- We believe you must start at the top: Get the pattern language right first and you stand a better change of doing other layers right.

A Shameless Plug



A pattern language for parallel algorithm design with examples in MPI, OpenMP, and Java.

This is our hypothesis about how expert parallel programmers think about parallel programming.

Now available at a bookstore near you!

What's a Design Pattern?

- High-quality solution to frequently occurring problem in some domain.
- Written in consistent format to allow readers to quickly understand context and solution.
- Named so that pattern names provide a vocabulary for discussing solutions (as has happened in object-oriented programming).

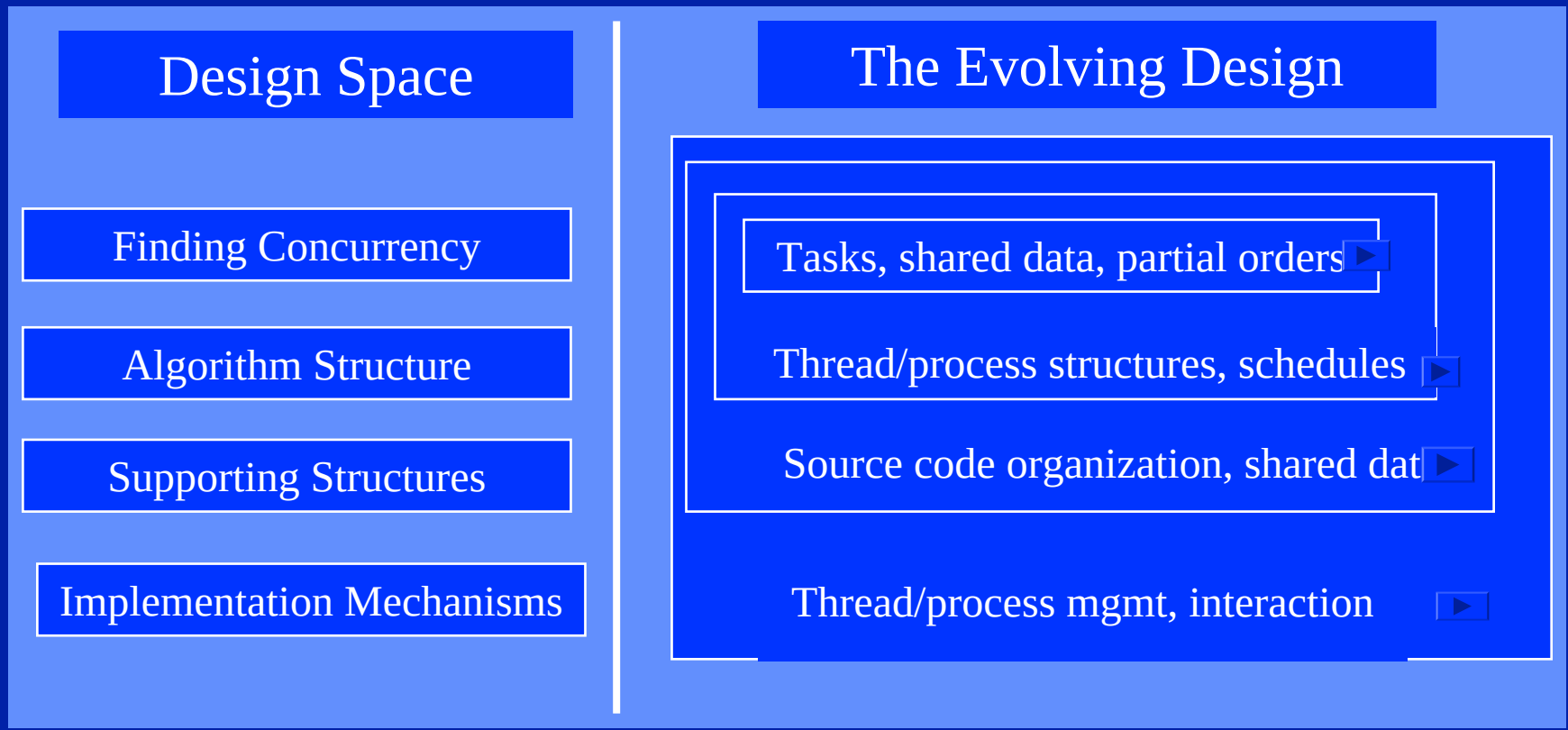
What's a Pattern Language?

- Carefully structured collection of patterns.
- Not a programming language.
- Can embody a design methodology, so user works through patterns to develop complex design using language's patterns.

The Pattern Language's Structure

A software design can be viewed as a series of refinements.

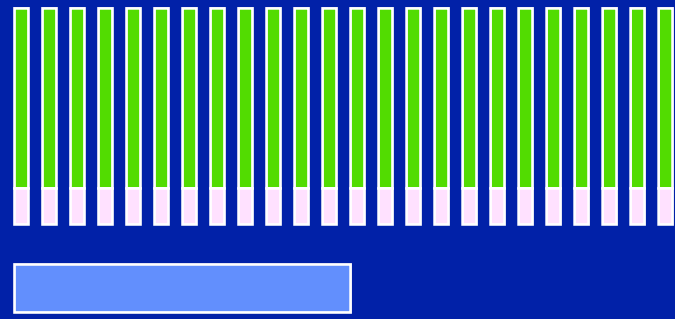
We consider the process in terms of 4 *design spaces* which add progressively lower-level elements to the design.



Parallel Software:

Where the design spaces fit in during software development

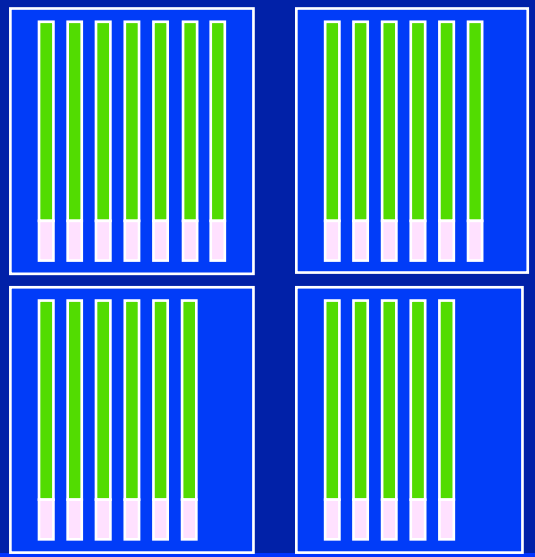
Finding Concurrency



Original Problem

Algorithm Structure

Tasks, shared and local data



Supporting struct. & impl. mech.

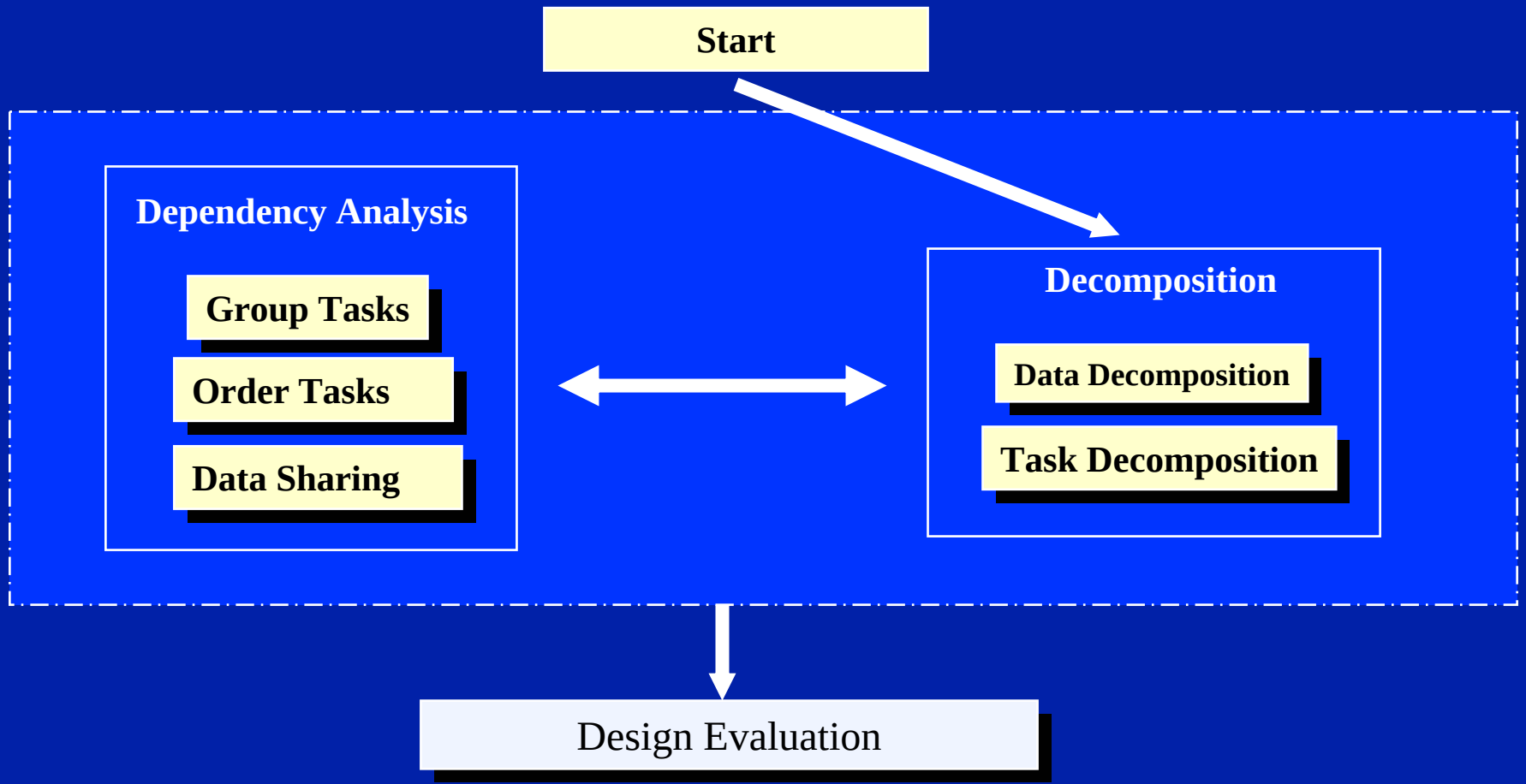
```
Program SPMD_Emb_Par ()
{
  Program SPMD_Emb_Par ()
  {
    Program SPMD_Emb_Par ()
    {
      Program SPMD_Emb_Par ()
      {
        TYPE *tmp, *func();
        global_array Data(TYPE);
        global_array Res(TYPE);
        int Num = get_num_procs();
        int id = get_proc_id();
        if (id==0) setup_problem(N, Data);
        for (int I= ID; I<N;I=I+Num){
          tmp = func(I, Data);
          Res.accumulate( tmp);
        }
      }
    }
  }
}
```

Units of execution + new shared data for extracted dependencies

Corresponding source code

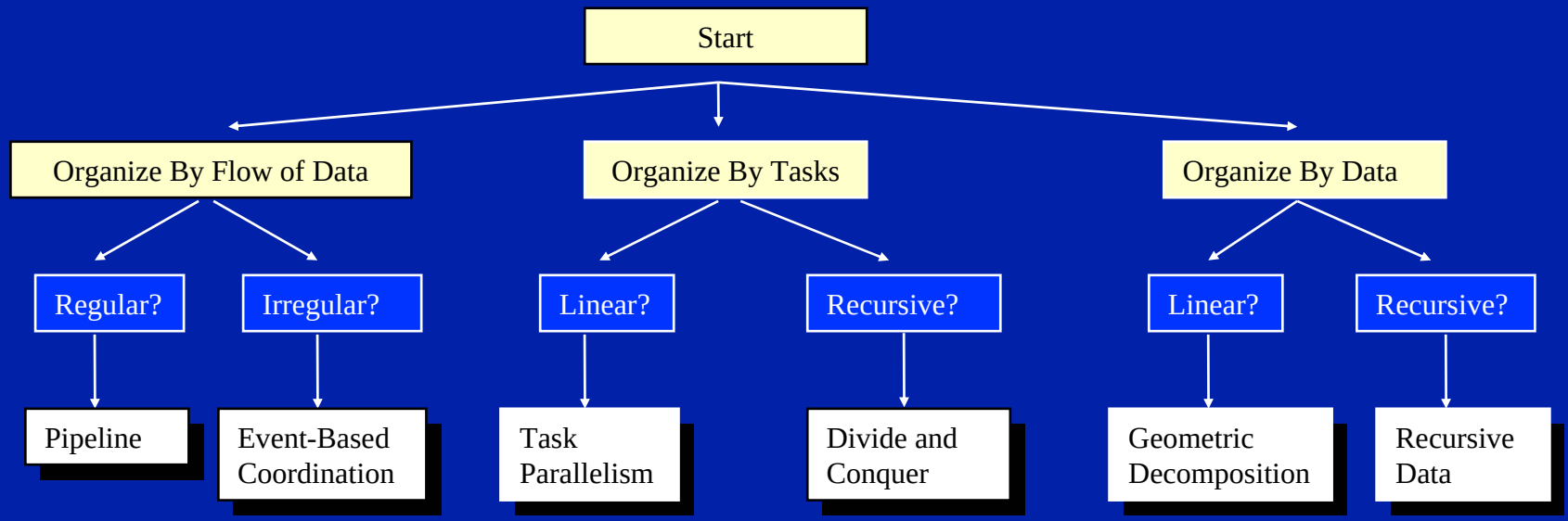
The Finding Concurrency Design Space

Start with a specification that solves the original problem, finish with a decomposition of the problem into tasks, plus an analysis of shared data and task dependencies (partial ordering).



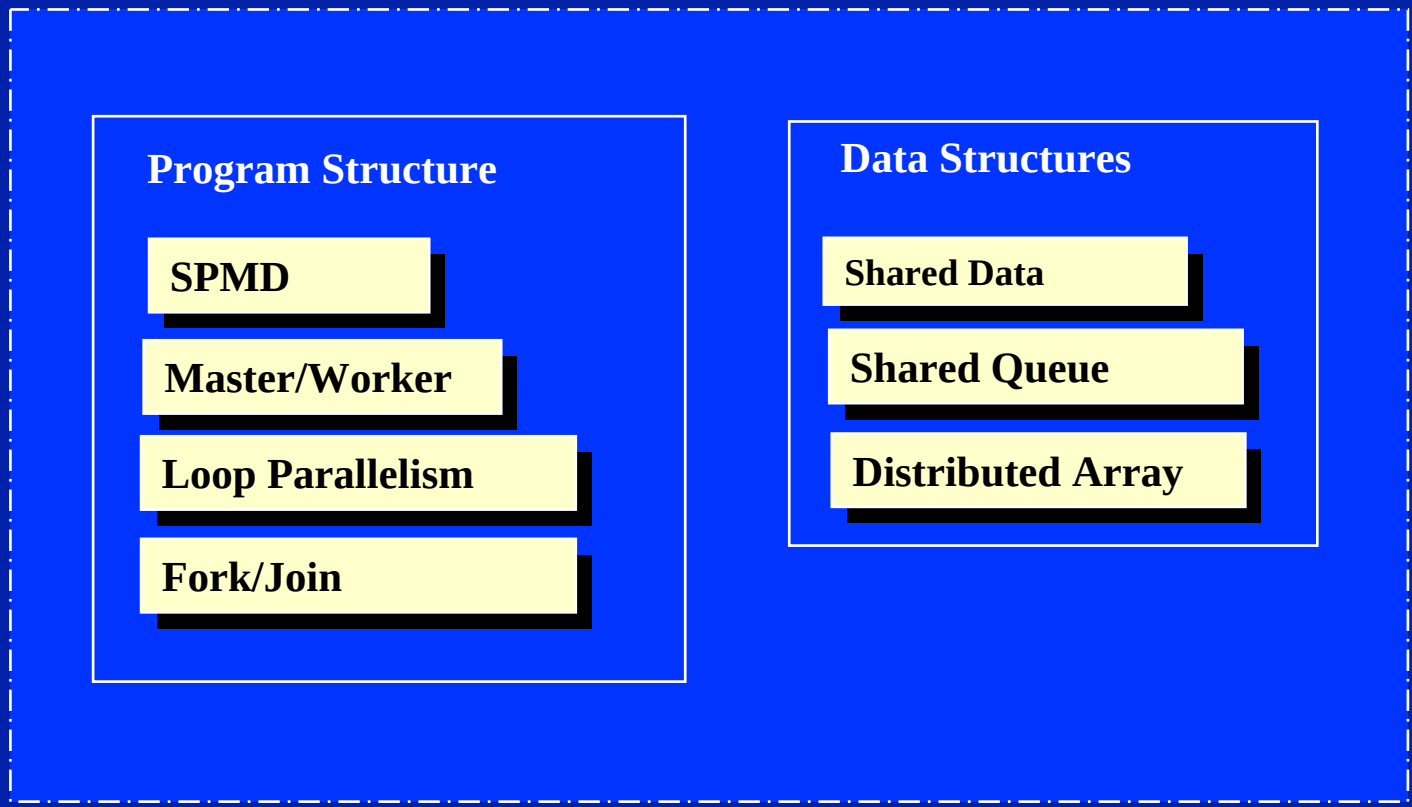
The Algorithm Structure Design Space

Select overall program organization to exploit concurrency identified in previous step.



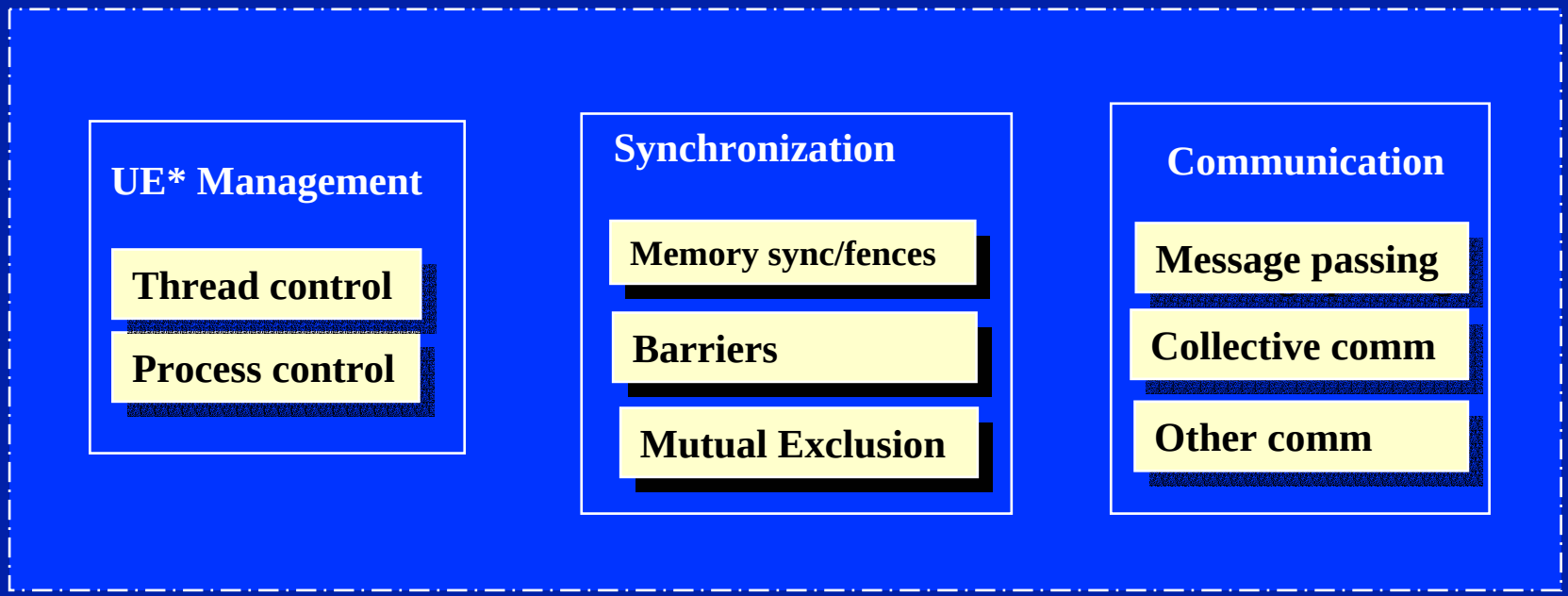
The Supporting Structures Design Space

High-level constructs affecting large-scale organization of the source code.



The Implementation Mechanisms Design Space

The “primitives” of parallel computing. Book’s examples are in Java, OpenMP, and MPI. This design space discusses key issues more generically. Not properly design patterns, included to make the pattern language self-contained.



* UE = Unit of execution

Open questions

- How close did we come to getting it right (identifying right/useful patterns)? We've heard from patterns people; now we need to hear from domain experts.
- Our patterns are modeled on 20 years of experience with HPC. Are they too narrow in scope?
- Our patterns come from an old-fashioned, procedural mindset and are not tied to modern object-oriented software design concepts. Should we be more GoF-like?
- Lowest-level design space was not represented as patterns. Does this suggest a need to be more abstract and a redesign of that space?