ProcessJ: A Possible Future of Process-Oriented Design

Jan Bækgaard Pedersen (UNLV)
Marc L. Smith (Vassar)

CPA-2013

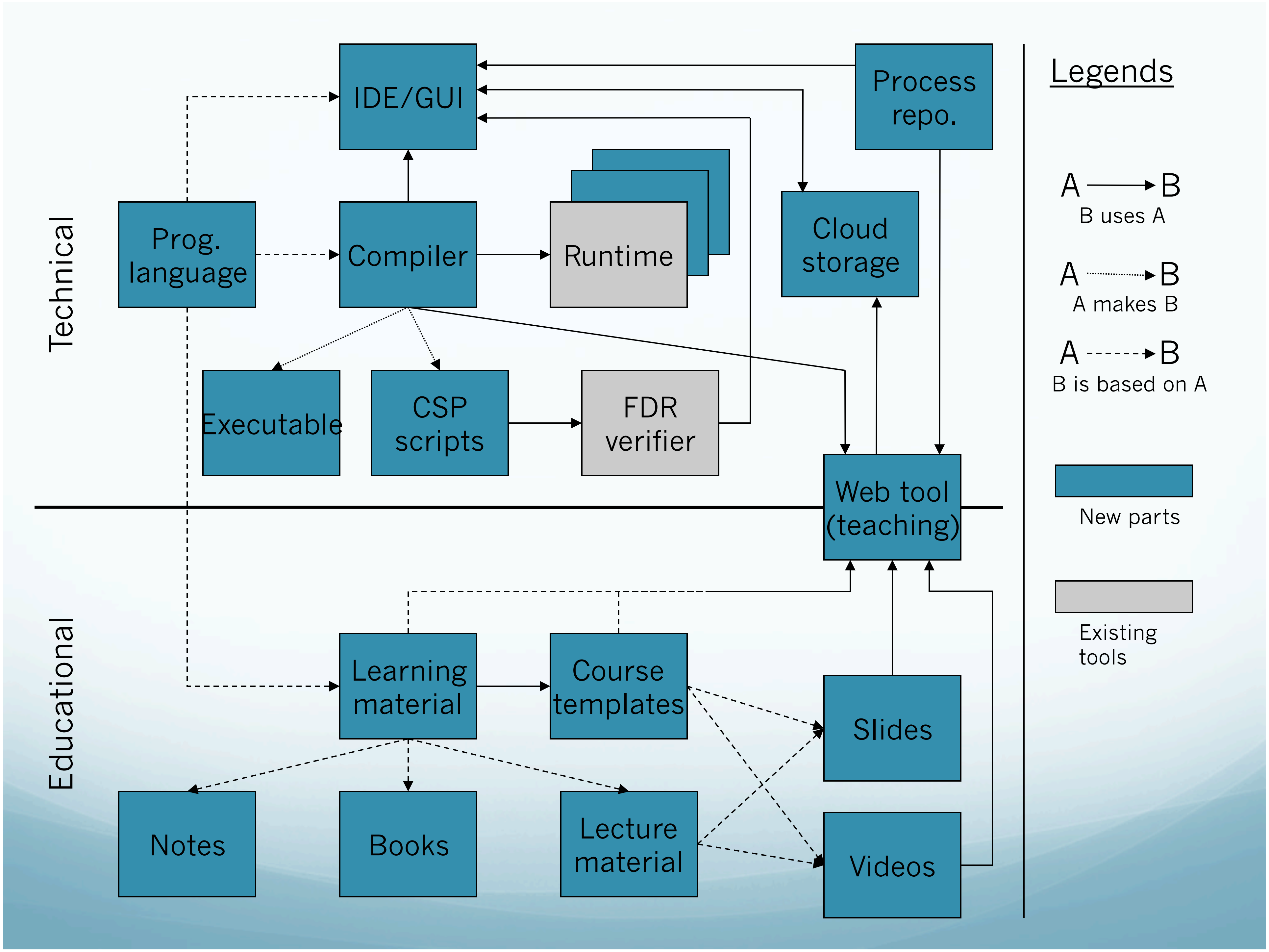
25-28 August 2013

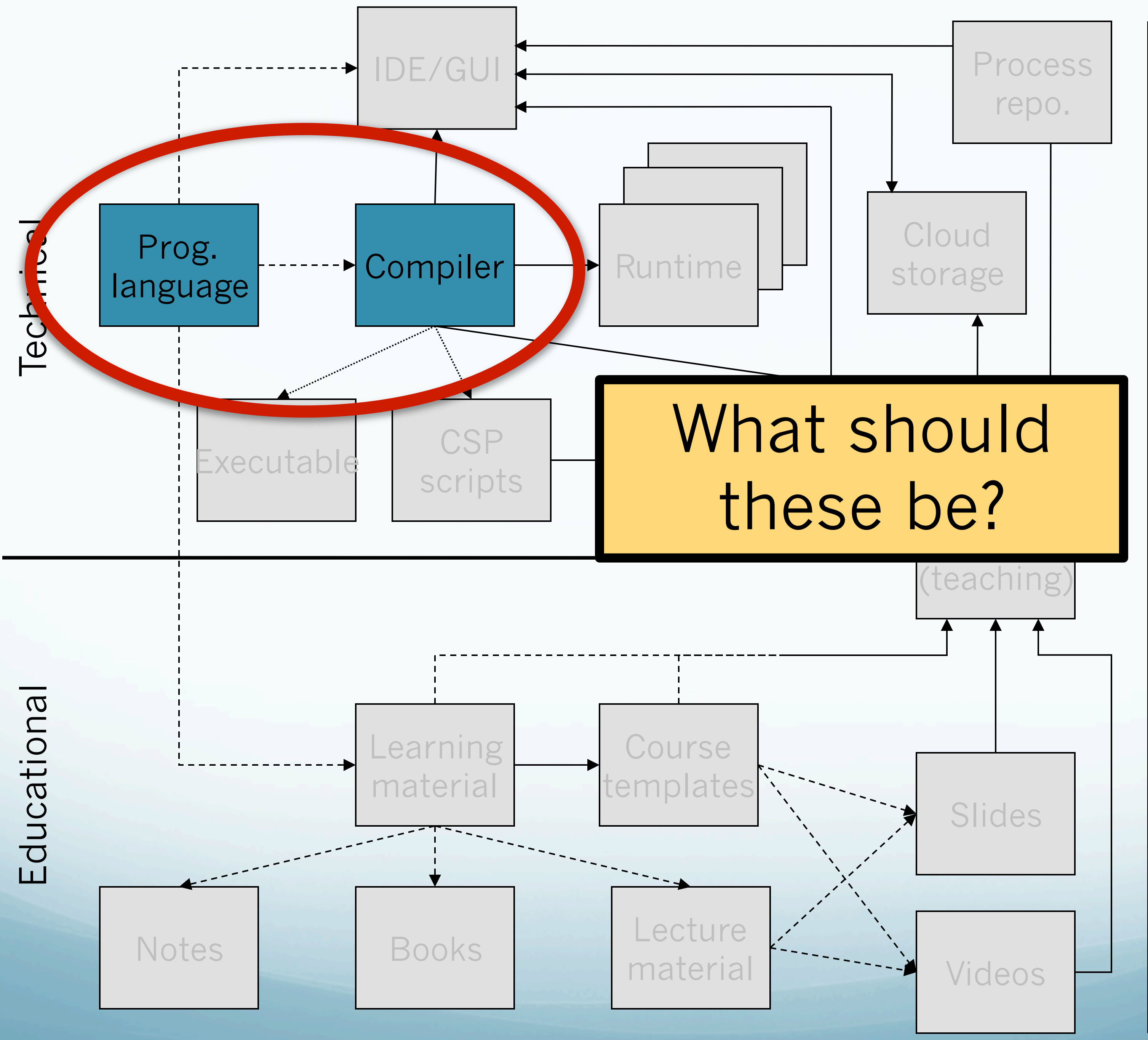
What do we have to do ….

- Question: What do we have to do to
  - spread the word about/use of process oriented design?
  - make the experience more easily accessible?
  - convince the distributed programming community that it actually is a good idea?
    - (This might include hostage taking)
  - Show that it can actually be used for HPC?

What do we have to do ....

- Answer:
  - Easy to learn language with the right semantics
    - Syntax of a familiar language (business as usual)
    - Safer semantics (but you can't do those bad things!)
  - Incorporation into online teaching tools
    - Accessible from anywhere
    - Easy to use in classrooms and teaching settings
  - Educational material
    - Finally someone will write "the book" (Peter?!)
    - Supporting course material (videos, ...)


Technical
Educational
IDE/GUI
Process repo.
Prog. language
Compiler
Runtime
Cloud storage
Executable
CSP scripts
FDR verifier
Web tool (teaching)
Learning material
Course templates
Slides
Notes
Books
Lecture material
Videos
Legends
A → B
B uses A
A ⋯→ B
A makes B
A - - → B
B is based on A
New parts
Existing tools



What should these be?

Legends

A → B
B uses A

A ⋯→ B
A makes B

A - - → B
B is based on A

🟦 New parts

⬜ Existing tools

New Language (again)

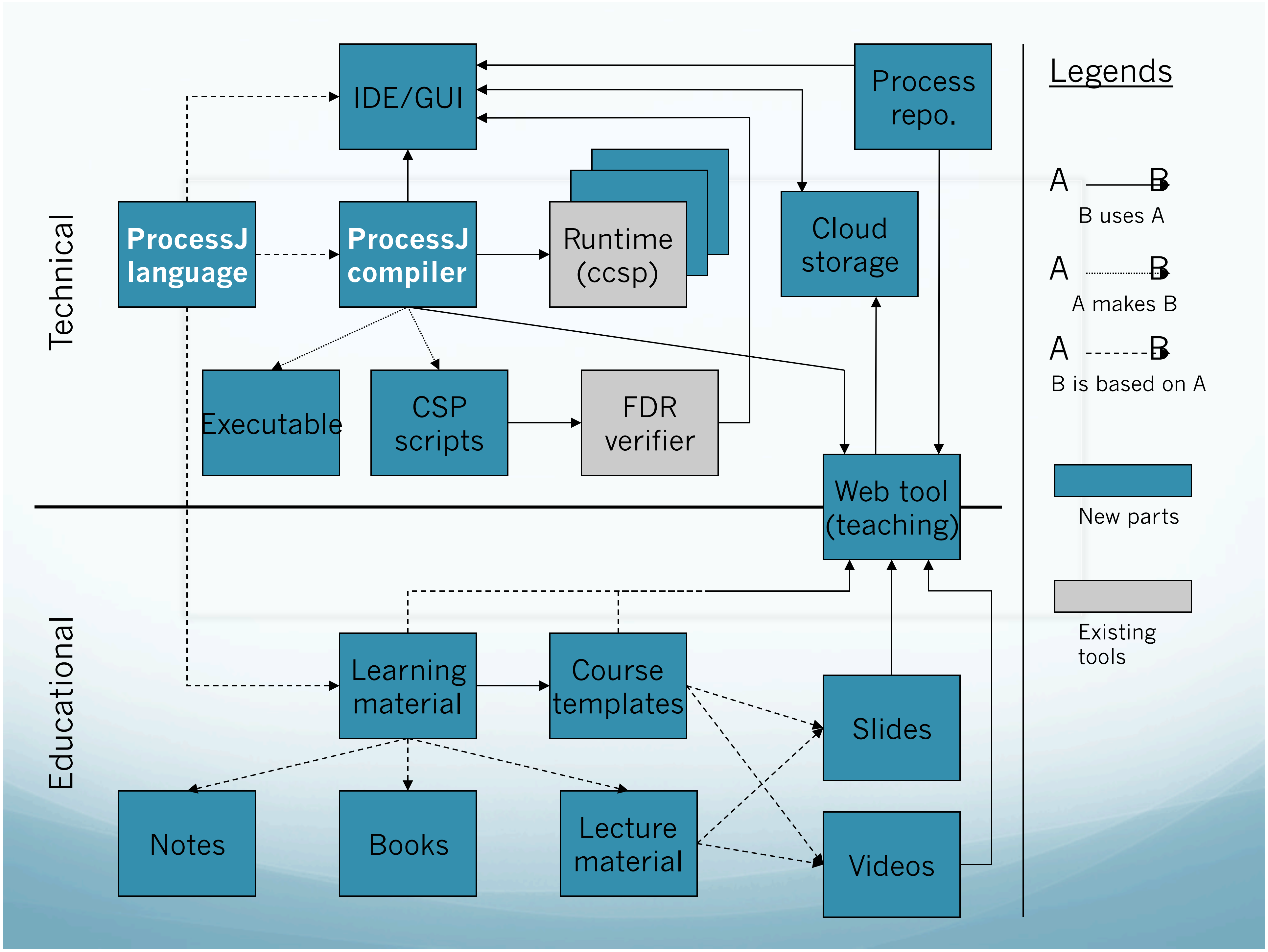
- Why not just provide a library for C/Java?
  - C: Too general – let you do bad things on purpose
    - Pointers
    - Shared memory
    - …
  - Java/JVM: Threading model of runtime too coarse
    - Speed?
    - Those pesky objects with wait and notify….
- If the programmer does not promise to behave 100% things will definitely go wrong.
- Programmer must stick to the “CSP Model”

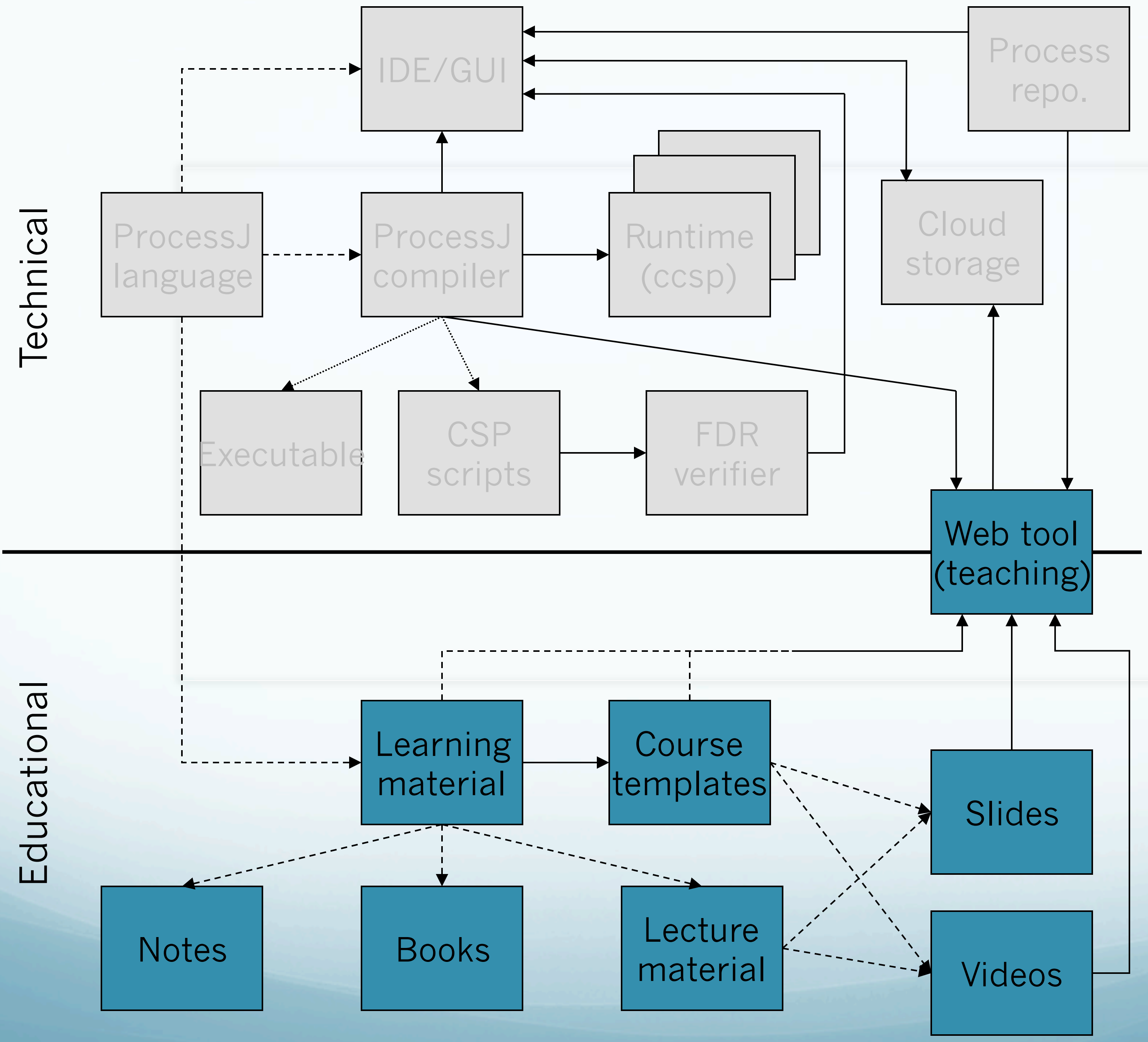
New Language (again)

- Perhaps a new syntax with a pre-processor and a source to source translator
  - A possibility – with some checking
- New language might more easily achieve and incorporate what we want
  - Integration of CSP generation
    - Might require a separate parser anyway

New Language (again)

- New language with the whole package (and more)
    - “known” syntax (like Java-ish)
    - Has communication primitives with CSP semantics
        - Synchronous channel communication
        - Barriers
        - Alternations
        - Parallel composition
        - Mobility (pi-calculus)
    - Has C/Java semantics for all other sequential language constructs
    - Has no shared memory abilities/pointers
    - Fast runtime system with multi-core scheduler


Technical
Educational
IDE/GUI
Process repo.
ProcessJ language
ProcessJ compiler
Runtime (ccsp)
Cloud storage
Executable
CSP scripts
FDR verifier
Web tool (teaching)
Learning material
Course templates
Slides
Notes
Books
Lecture material
Videos
Legends
A B
B uses A
A B
A makes B
A B
B is based on A
New parts
Existing tools


Technical
Educational
IDE/GUI
Process repo.
ProcessJ language
ProcessJ compiler
Runtime (ccsp)
Cloud storage
Executable
CSP scripts
FDR verifier
Web tool (teaching)
Learning material
Course templates
Slides
Notes
Books
Lecture material
Videos

Education is Everything

- Process-Oriented Design
  - We must teach it...
  - ...if we want students to learn it
- The longer we wait
  - The harder it is to teach
  - The harder it is to learn
- The longer we wait
  - The less natural it feels
  - The lower our chance of success

Language is Everything

- Forget about ProcessJ for a moment...
  and programming languages
- Programming languages ⟺ Natural languages
- The language we speak shapes our thoughts
- Teach foreign languages too late

  ⟹ children unlikely to become multilingual

Programming Language is Everything

- The language we learn to program in shapes how we solve problems
- Teach concurrent languages too late $\Rightarrow$ students unlikely to become parallel programmers
- We may say we care more about concurrency and process-oriented design more than any language…
- …but without an accessible process-oriented language, concurrency remains inaccessible

☹

The Wrong Debate(s)!

- The CS Education community has been debating the wrong things:
  - FP vs OOP ?
  - Objects first vs Objects early vs Objects later vs Objects late ?
- What about: Sequential vs Concurrent ?
  - The debate that hasn’t happened (yet!)
  - Institutional bias toward sequential!
  - We teach it like we learned it

Race[condition]ism!

- Conventional wisdom:
  - Concurrency is hard!
    - Hard to write (from an ingrained sequential mindset)
    - Hard to test (too many possible interleavings!)
    - Hard to debug (Heisenbugs)
  - In short: Hard to reason about!
- Conventional Wisdom is wrong!
- CW applies to particular models of concurrency
    - Threads and Locks with shared memory
    - Asynchronous message passing

~~Special~~ Sequential Interests

- Compositional blind spot!
- Sequential composition (in Java or your favorite OOL)
  - Data composes (i.e., data structures)
  - Code composes (i.e., code blocks)
  - Data and Methods compose (i.e., classes)
  - Classes compose (i.e., packages/libraries)
  - Packages compose (i.e., frameworks)
- Concurrent Composition?
  - Threads don't compose $\Rightarrow$ group of threads
  - Asynchronous message passing $\Rightarrow$ larger collection of objects and message buffers

Process-Oriented Abstraction is Everything

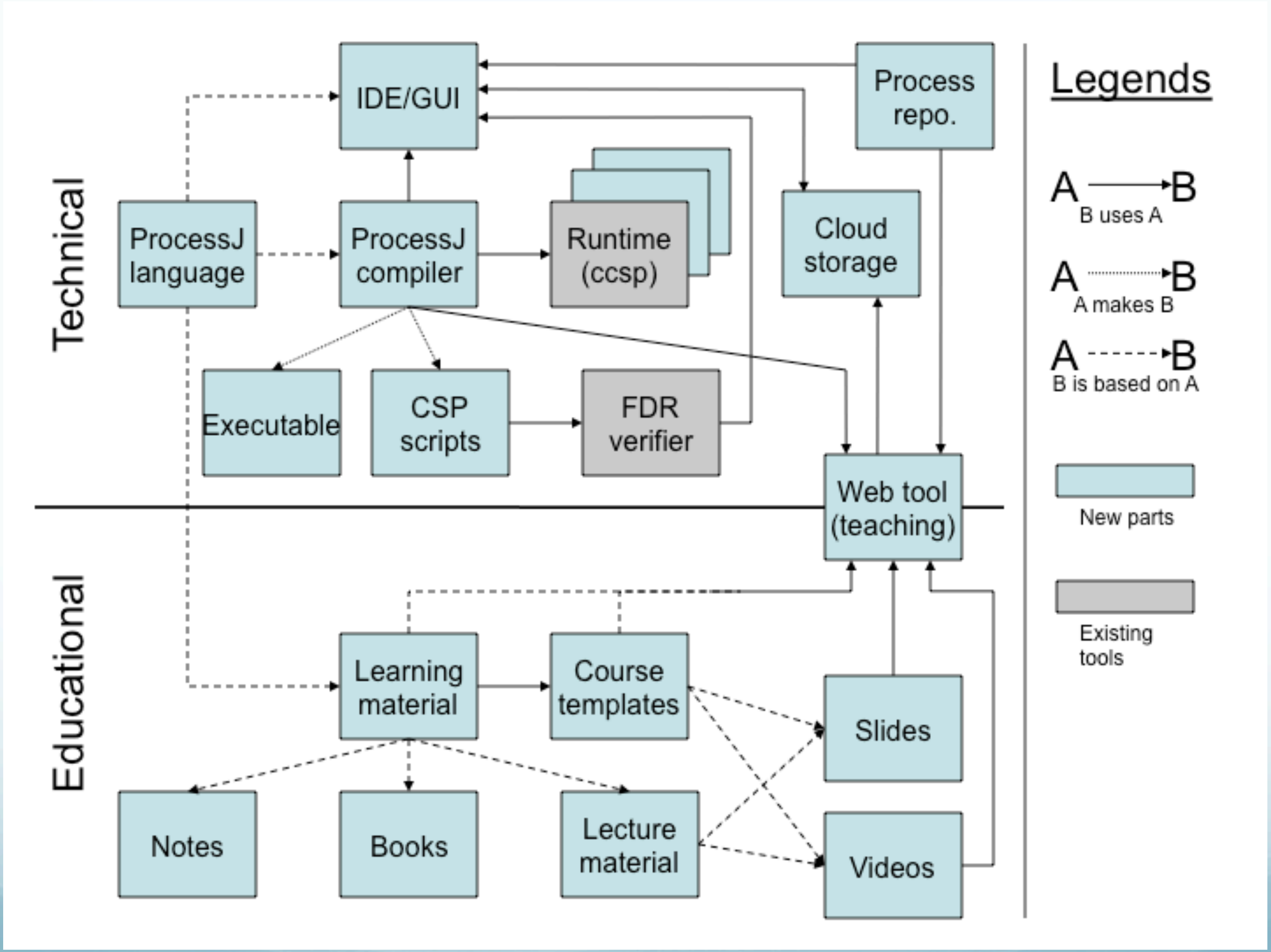
- Compositional / Concurrency Abstraction
- Processes compose explicitly:
  - Sequentially
  - Parallel
  - Choice (alternation)
- The composition of two or more processes is a process! (which can in turn be further composed)
- Reason about processes at an appropriate level of abstraction (internal hiding)

Process-Oriented Design Makes All the Difference

- Lowest level processes are sequential
  CSP, after all, stands for
  Communicating Sequential Processes
- Parallel composition is just another choice for composition, alongside sequential (;) composition!
- With the right language and syntax, Process-Oriented design could be taught:
  First?
  Early?
  Later?
  Late?

The earlier the better—let's not debate this!

ProcessJ is Everything



Timing is Everything

- ACM/IEEE-CS Joint Task Force for Computing Curricula. Computer Science Curricula 2013, Ironman Draft
  - includes for the first time Parallel Programming in the Core [undergraduate] Curriculum
  - new Knowledge Area (KA): Parallel and Distributed Computing
  - Concurrency is no longer an elective topic!

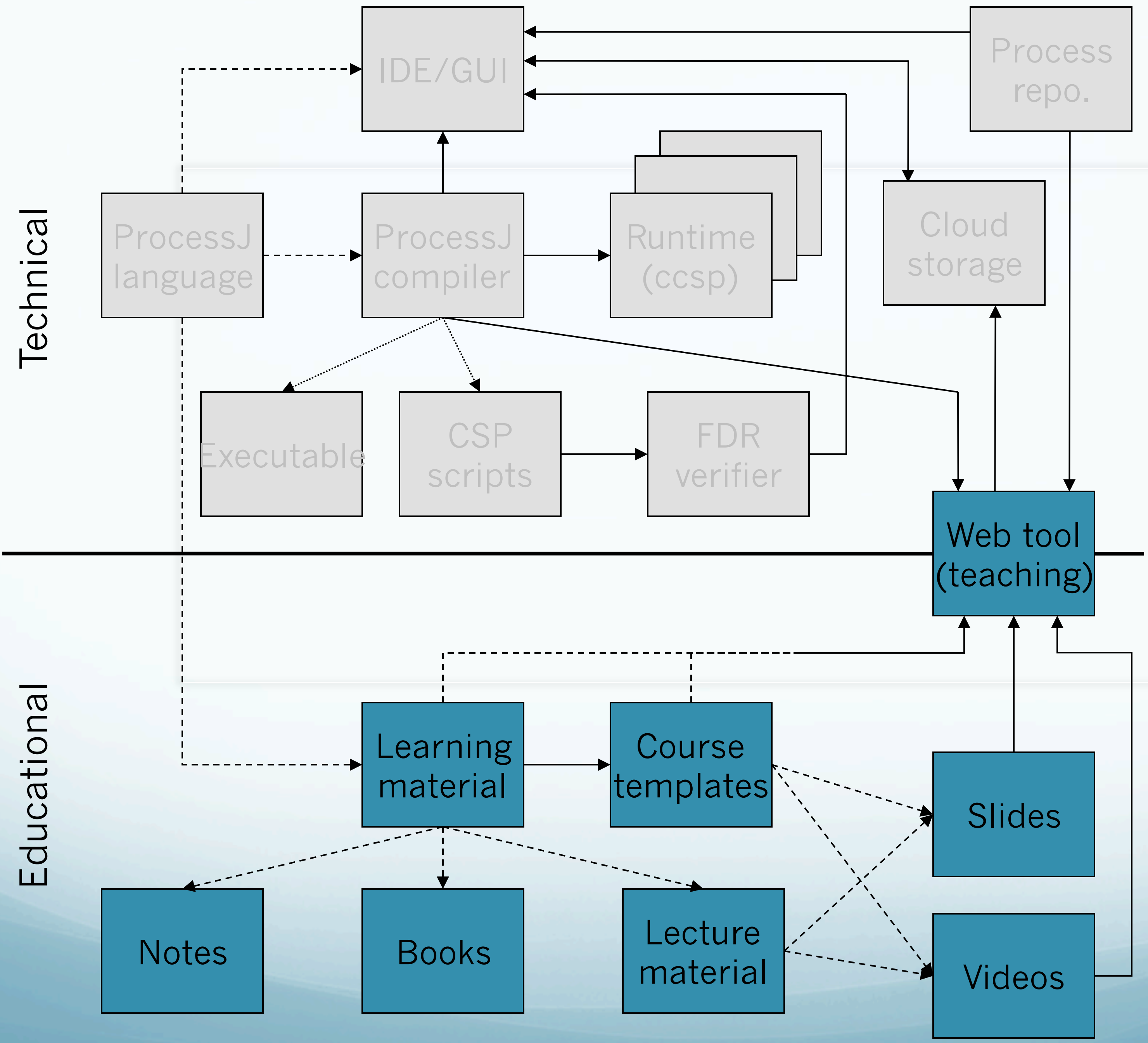
Context is Everything (from the Ironman draft)

"*Parallel computing:* Among the many changes to the Body of Knowledge compared to previous reports is a new Knowledge Area in Parallel and Distributed Computing. An alternative structure for the Body of Knowledge would place relevant topics in other Knowledge Areas: parallel algorithms with algorithms, programming constructs in software-development focused areas, multi-core design with computer architecture, and so forth. We chose instead to provide guidance on the essential parallelism topics in one place. Some, but not all, curricula will likely have courses dedicated to parallelism, at least in the near term."

From a Research Perspective

NSF Call for Proposals:

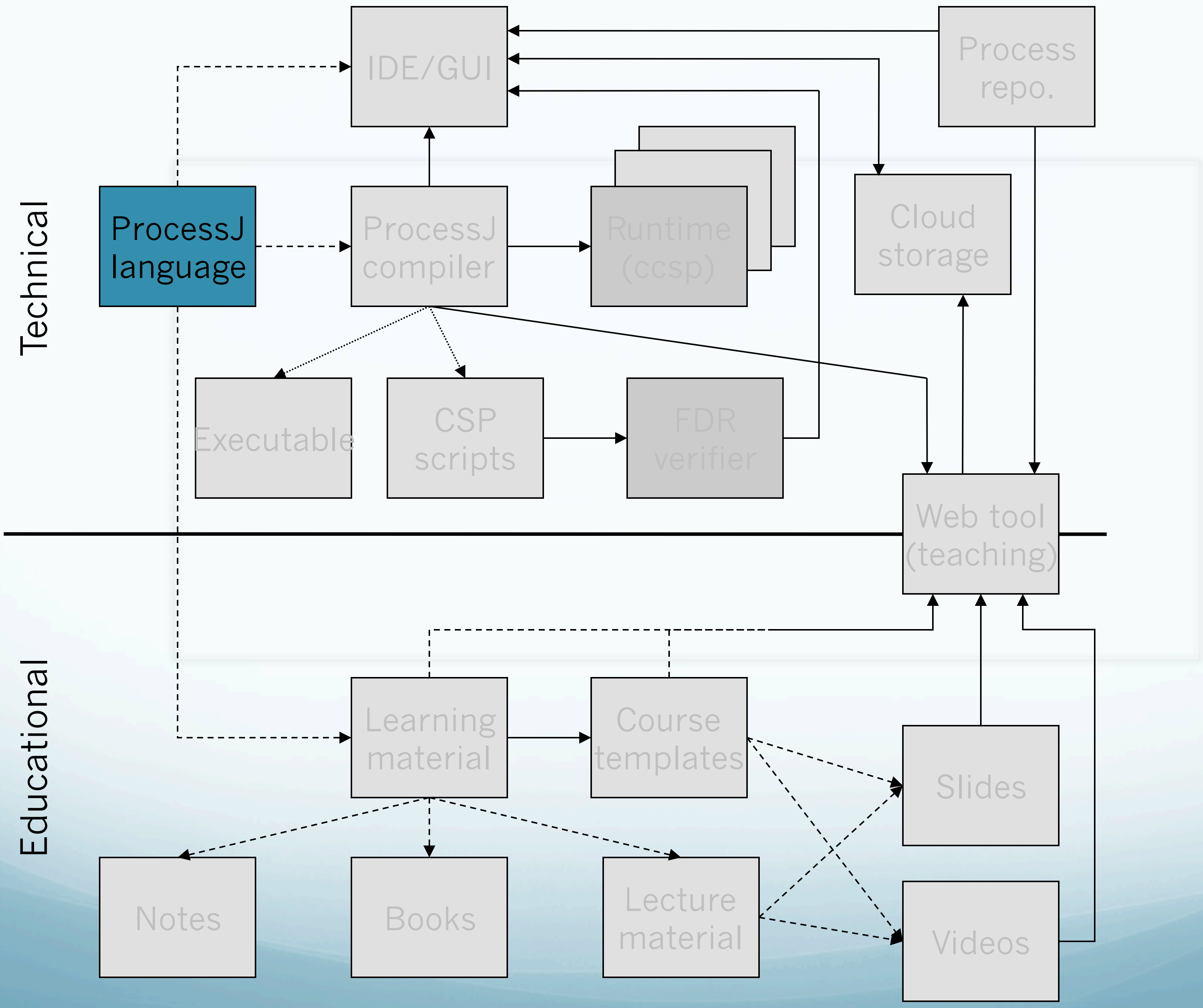
"New programming languages and language mechanisms that support new computational models, raise the level of abstraction, and lower the barrier of entry for parallel and concurrent programming. Parallel and concurrent languages that have programmability, verifiability, and scalable performance as design goals. Of particular interest are languages that abstract away from the traditional imperative programming model found in most sequential programming languages."



Technical
Educational
IDE/GUI
Process repo.
ProcessJ language
ProcessJ compiler
Runtime (ccsp)
Cloud storage
Executable
CSP scripts
FDR verifier
Web tool (teaching)
Learning material
Course templates
Slides
Notes
Books
Lecture material
Videos

Support is Everything

- We must help make it easier for faculty to consider adopting Process-Oriented Design in their courses
- The Web Tool will be the portal to supporting materials, including
  - Learning material: notes, book(s), lectures
  - Course templates
  - Slides and videos


Technical
Educational
IDE/GUI
Process repo.
ProcessJ language
ProcessJ compiler
Runtime (ccsp)
Cloud storage
Executable
CSP scripts
FDR verifier
Web tool (teaching)
Learning material
Course templates
Slides
Notes
Books
Lecture material
Videos

ProcessJ Language

- ProcessJ has Java-like syntax
  - Without objects
  - With records and arrays
  - CSP primitives:
    - Synchronous channels
    - Alternation
    - Barriers
    - Protocols
    - ....
    - All the good stuff we know from occam

ProcessJ Language

- Java like statements for while, do, for etc.
  - with Java/C semantics
- Channel communication, alternation etc.
  - with CSP semantics
- Process Mobility
  - pi-calculus
  - Polymorphic resumption interfaces

ProcessJ Example

```
proc void Producer (chan<int>.write out) {
  int x = 42;
  while (true) {
     while (x < 1000) {
     out.write (x);
     x++;
  }
  while (x > 0) {
    out.write (x);
    x--;
  }
}
```

ProcessJ Example

```
proc void Producer (chan<int>.write out) {
  int x = 42;
  while (true) {
     while (x < 1000) {
     out.write (x);
     x++;
  }
  while (x > 0) {
    out.write (x);
    x--;
  }
}
```

Writing end of a channel carrying integers; named out.

ProcessJ Example

```
proc void Producer (chan<int>.write out) {
  int x = 42;
  while (true) {
     while (x < 1000) {
     out.write (x);
     x++;
  }
  while (x > 0) {
    out.write (x);
    x--;
  }
}
```

Write the value of x to the channel out.

ProcessJ Example

```
proc void Producer (chan<int>.write out) {
  int x = 42;
  while (true) {
     while (x < 1000) {
     out.write (x);
     x++;
  }
  while (x > 0) {
    out.write (x);
    x--;
  }
}
```

ProcessJ Example

```
proc void Monitor (chan<int>.read in) {
  int last = in.read ();
  while (true) {
    int x;
    x = in.read ();
    if  (x == last) {
      ... system failure detected
    }
    last = x;
  }
}
```

ProcessJ Example

```
proc void Monitor (chan<int>.read in) {
 int last = in.read ();
 while (true) {
   int x;
   x = in.read ();
   if (x == last) {
     ... system failure detected
   }
   last = x;
 }
}
```

Reading end of a channel carrying integers; named in.

ProcessJ Example

```
proc void Monitor (chan<int>.read in) {
  int last = in.read ();
  while (true) {
    int x;
    x = in.read ();
    if (x == last) {
      ... system failure detected
    }
    last = x;
  }
}
```

Reading from the channel named in

ProcessJ Example

```
proc void Monitor (chan<int>.read in) {
  int last = in.read ();
  while (true) {
    int x;
    x = in.read ();
    if  (x == last) {
      ... system failure detected
    }
    last = x;
  }
}
```

ProcessJ Example

```
proc void main() {
  chan<int> c;
  par {
    Producer (c.write);
    Monitor (c.read);
  }
}
```

ProcessJ Example

```
proc void main() {
  chan<int> c;
  par {
    Producer (c.write);
    Monitor (c.read);
  }
}
```

Declare a channel carrying integers

ProcessJ Example

```
proc void main() {
  chan<int> c;
  par {
    Producer (c.write);
    Monitor (c.read);
  }
}
```

In parallel run Producer and Monitor

ProcessJ Example

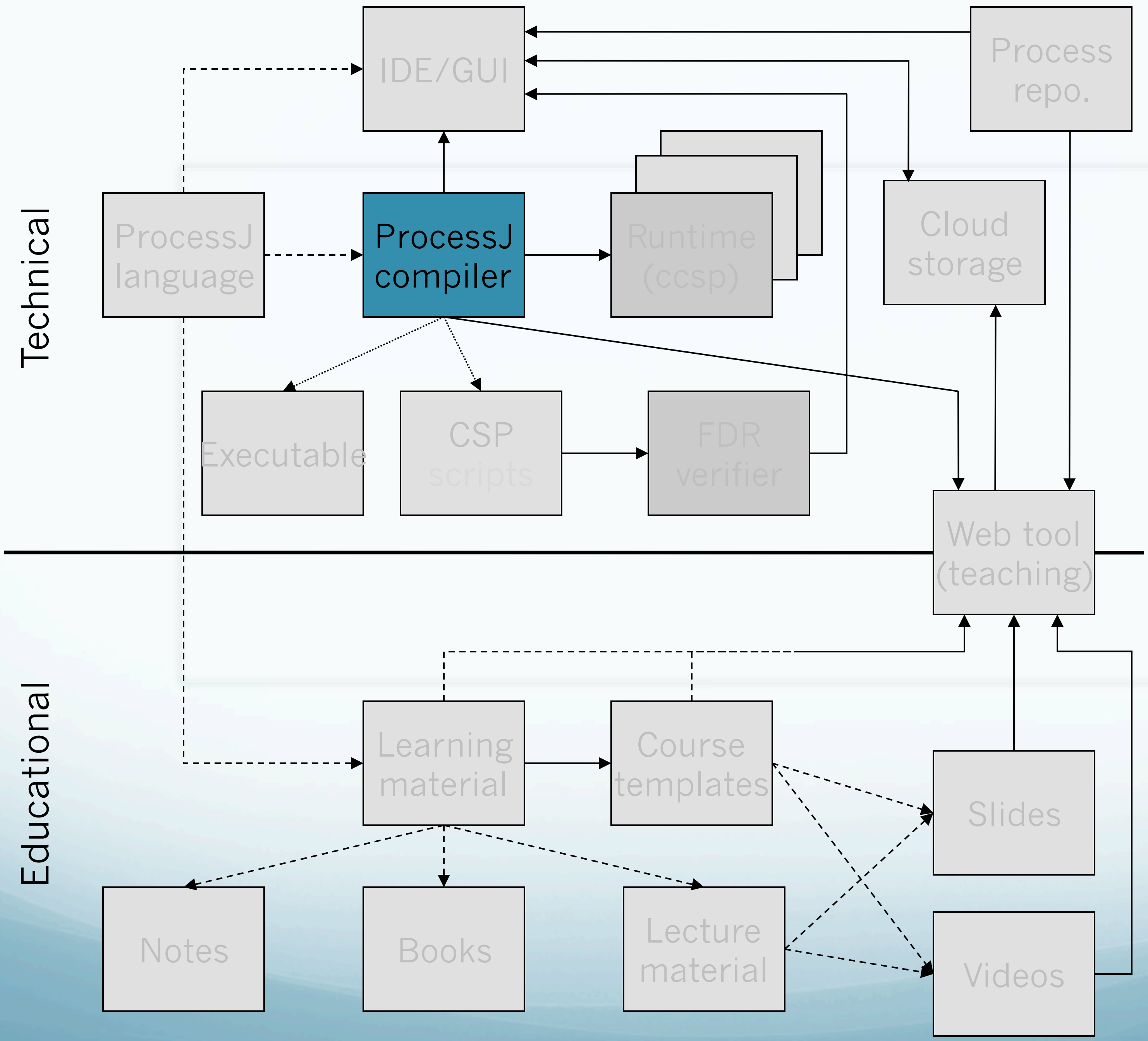
```
proc void main() {
  chan<int> c;
  par {
    Producer (c.write);
    Monitor (c.read);
  }
}
```

ProcessJ Example

```
proc void Producer (chan<int>.write out) {
  int x = 42;
  while (true) {
     while (x < 1000) {
     out.write (x);
     x++;
  }
  while (x > 0) {
    out.write (x);
    x--;
  }
}
```

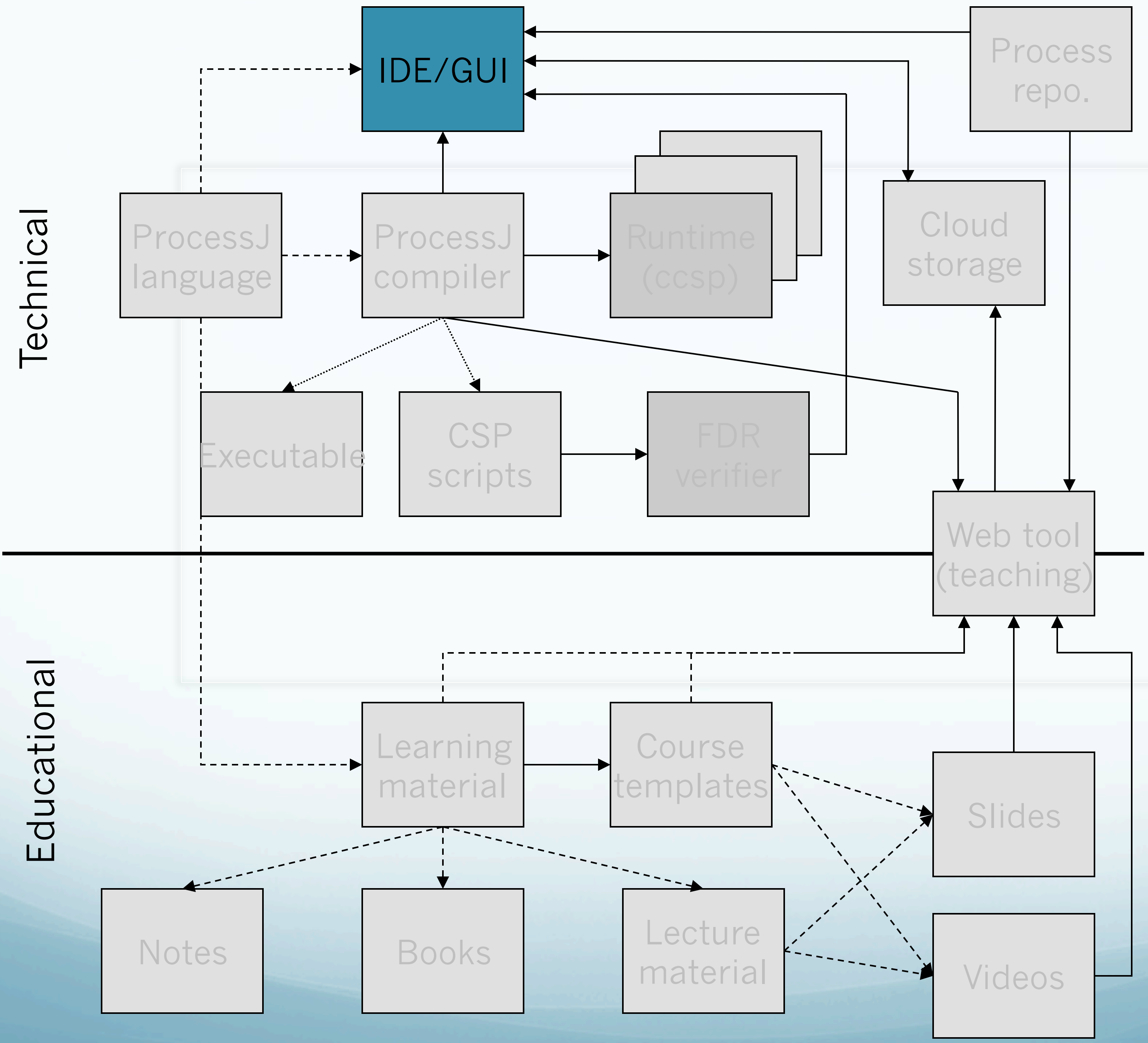
```
proc void Monitor (chan<int>.read in) {
  int last = in.read ();
  while (true) {
    int x;
    x = in.read ();
    if  (x == last) {
      ... system failure detected
    }
    last = x;
  }
}
```

```
proc void main() {
  chan<int> c;
  par {
    Producer (c.write);
    Monitor (c.read);
  }
}
```


Technical
Educational
IDE/GUI
Process repo.
ProcessJ language
ProcessJ compiler
Runtime (ccsp)
Cloud storage
Executable
CSP scripts
FDR verifier
Web tool (teaching)
Learning material
Course templates
Slides
Notes
Books
Lecture material
Videos

ProcessJ Compiler

- New compiler written in C/C++ using the CCSP runtime.
- Other back ends:
  - JavaScript for online teaching tool
  - MPI
- Generates CSP-M for FDR 3 (;-)) checking


Technical
Educational
IDE/GUI
Process repo.
ProcessJ language
ProcessJ compiler
Runtime (ccsp)
Cloud storage
Executable
CSP scripts
FDR verifier
Web tool (teaching)
Learning material
Course templates
Slides
Notes
Books
Lecture material
Videos

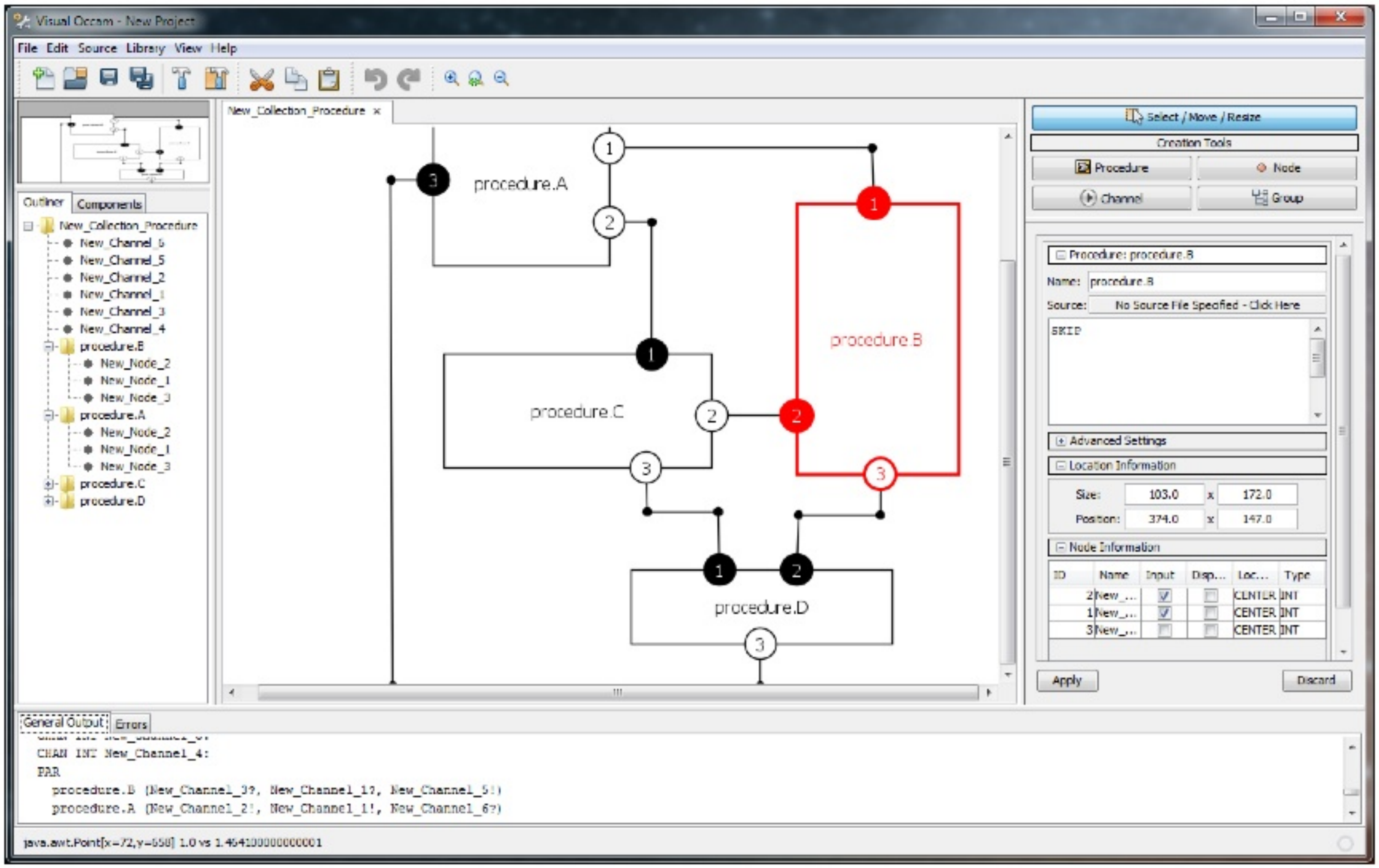
GUI for ProcessJ

- Process oriented programming is well suited for graphical programming
  - Processes are nodes
  - Channels are arcs
- Processes can consist of other processes which can consist of other processes etc.
  - Processes are building blocks for other processes
  - We have a “Lego” catalogue of well known processes
  - Code reuse is as easy as drag and drop (and connect external channels)

GUI for ProcessJ

- GUI should integrate into
  - Process repository
  - Online Cloud Storage
- GUI should produce CSP based on process layout
  - And perhaps also documentation
- Visual occam is a graphical programming interface (M.Sc. Thesis project) – proof of concept.
  - Works for occam
  - Written in Java

GUI – Visual occam



Procedure: procedure.B

Name: procedure.B

Source: No Source File Specified - Click Here

```
SKIP
```

Advanced Settings

Location Information

Size: 103.0 x 172.0

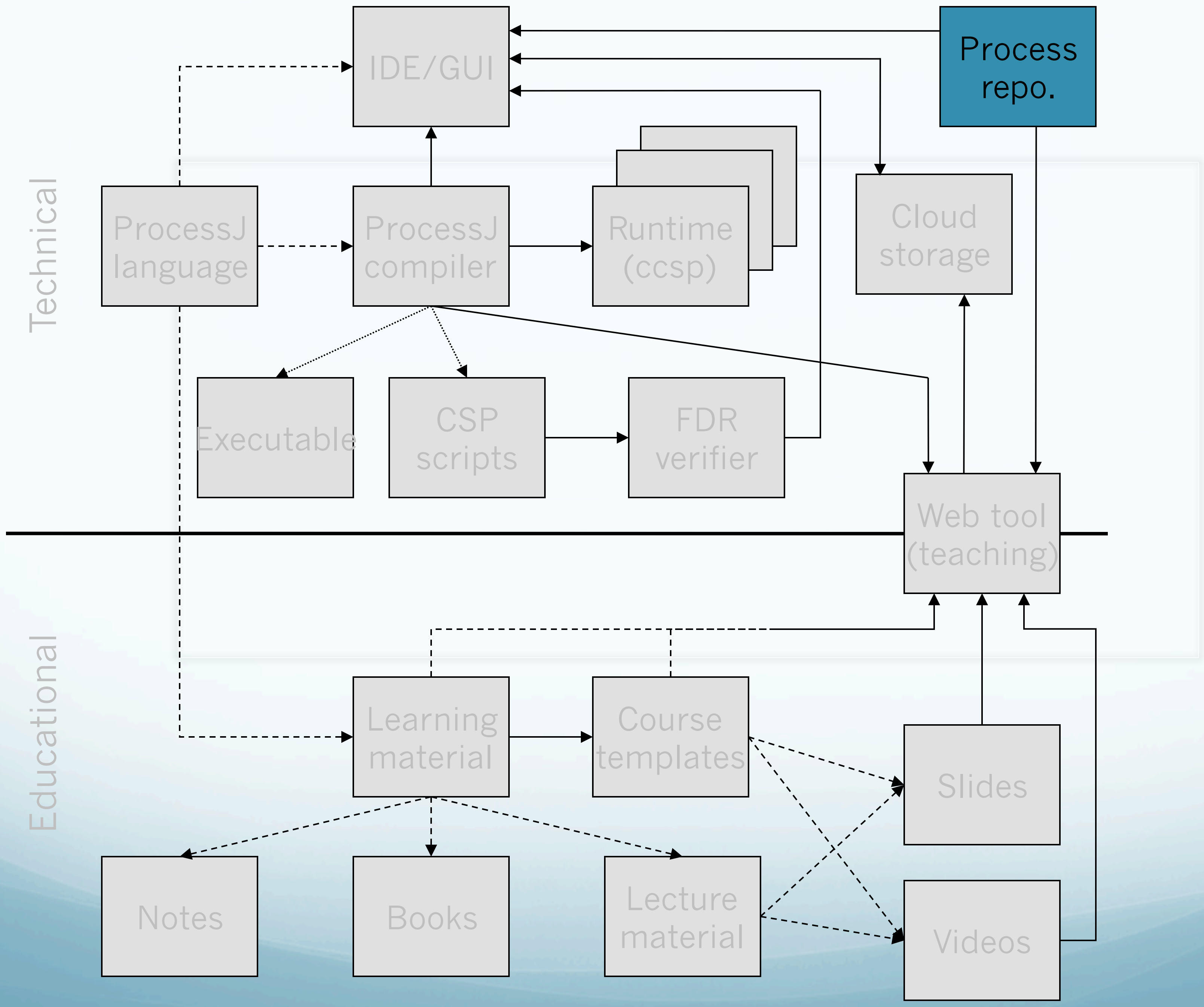
Position: 374.0 x 147.0

Node Information

ID	Name	Input	Disp...	Loc...	Type
2	New_...	✓		CENTER	INT
1	New_...	✓		CENTER	INT
3	New_...			CENTER	INT

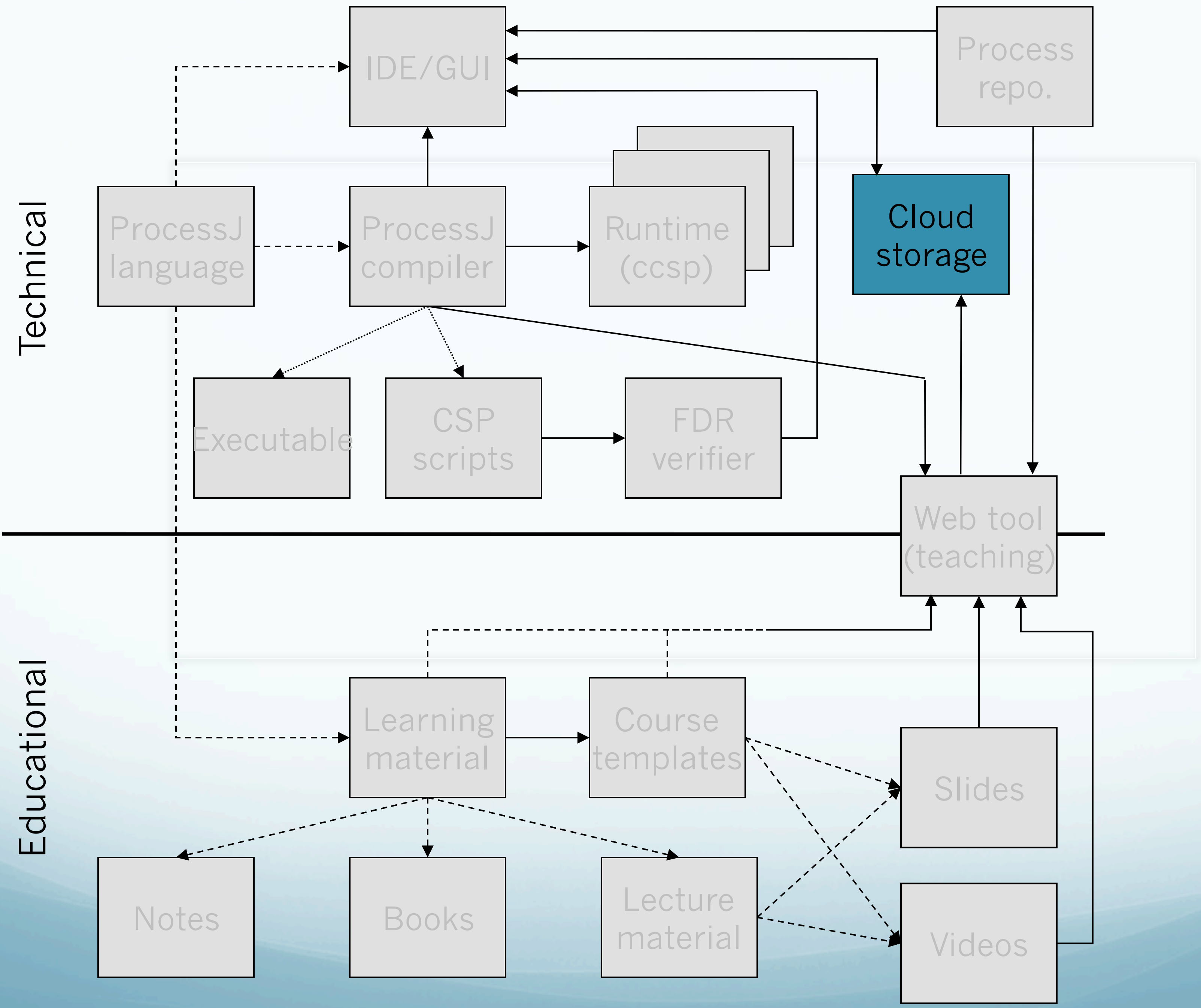
```
  CHAN INT New_Channel_4:
  PAR
    procedure.B (New_Channel_3?, New_Channel_1?, New_Channel_5!)
    procedure.A (New_Channel_2!, New_Channel_1!, New_Channel_6?)
```

```
java.awt.Point[x=72,y=658] 1.0 vs 1.4641000000000001
```


Technical
Educational
IDE/GUI
Process repo.
ProcessJ language
ProcessJ compiler
Runtime (ccsp)
Cloud storage
Executable
CSP scripts
FDR verifier
Web tool (teaching)
Learning material
Course templates
Slides
Notes
Books
Lecture material
Videos

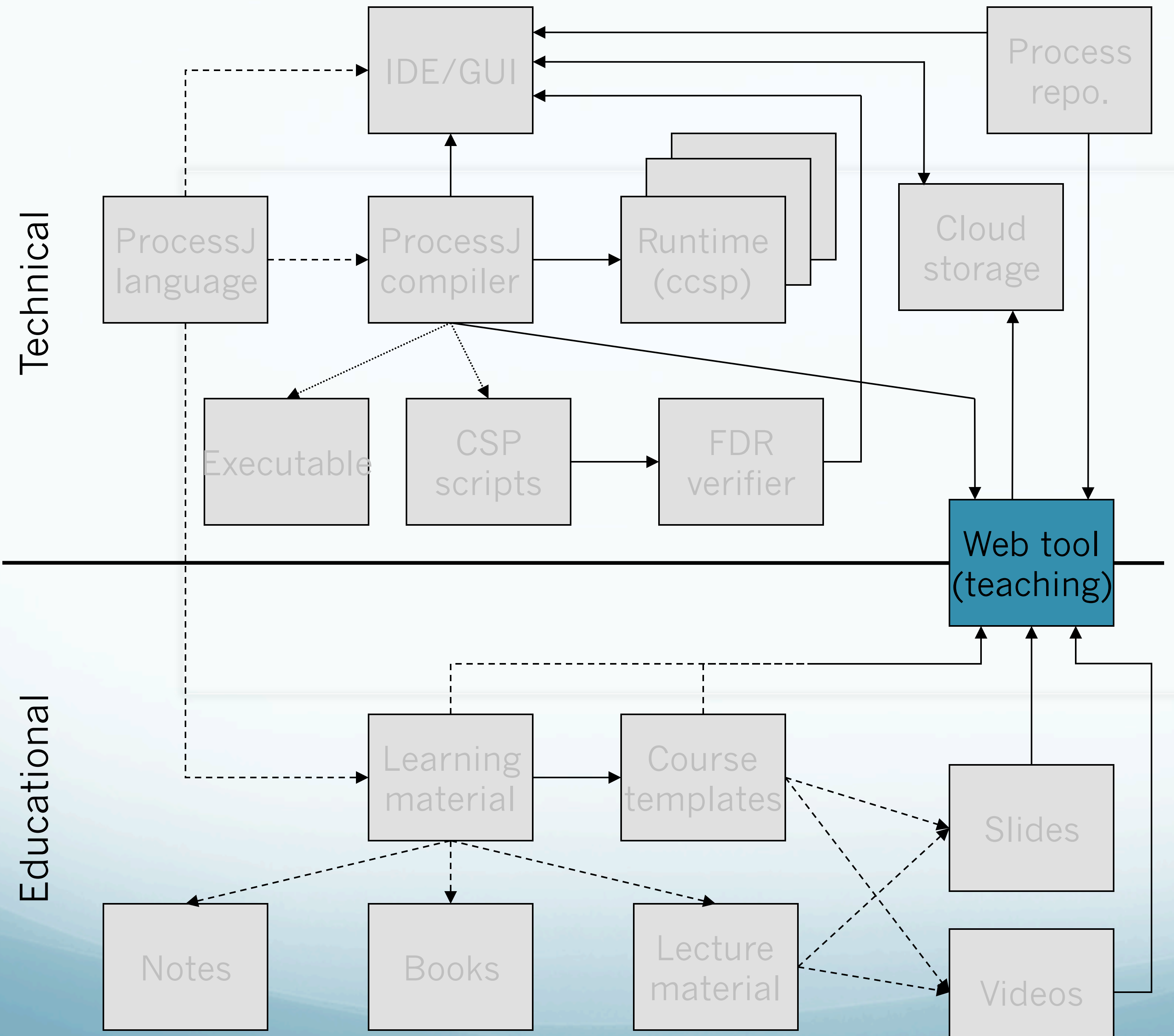
ProcessJ Process Repository

- Like the Apple App Store, but with source
  - Free ;-)
  - For developers/programmers
- ProcessJ code can be shared
  - Annotated with CSP assertions certificate
    - Deadlock free, livelock free, ..
    - Programmer generated assertions
  - Example code for use can be associated


Technical
Educational
IDE/GUI
Process repo.
ProcessJ language
ProcessJ compiler
Runtime (ccsp)
Cloud storage
Executable
CSP scripts
FDR verifier
Web tool (teaching)
Learning material
Course templates
Slides
Videos
Notes
Books
Lecture material

Cloud Storage

- Online storage of ProcessJ code / CSP
  - Accessible through
    - Web interface (ProcessJ website)
    - GUI
    - Online Teaching Tool
  - Integrating well known storage sites like
    - GitHub
    - Google Drive
    - Dropbox

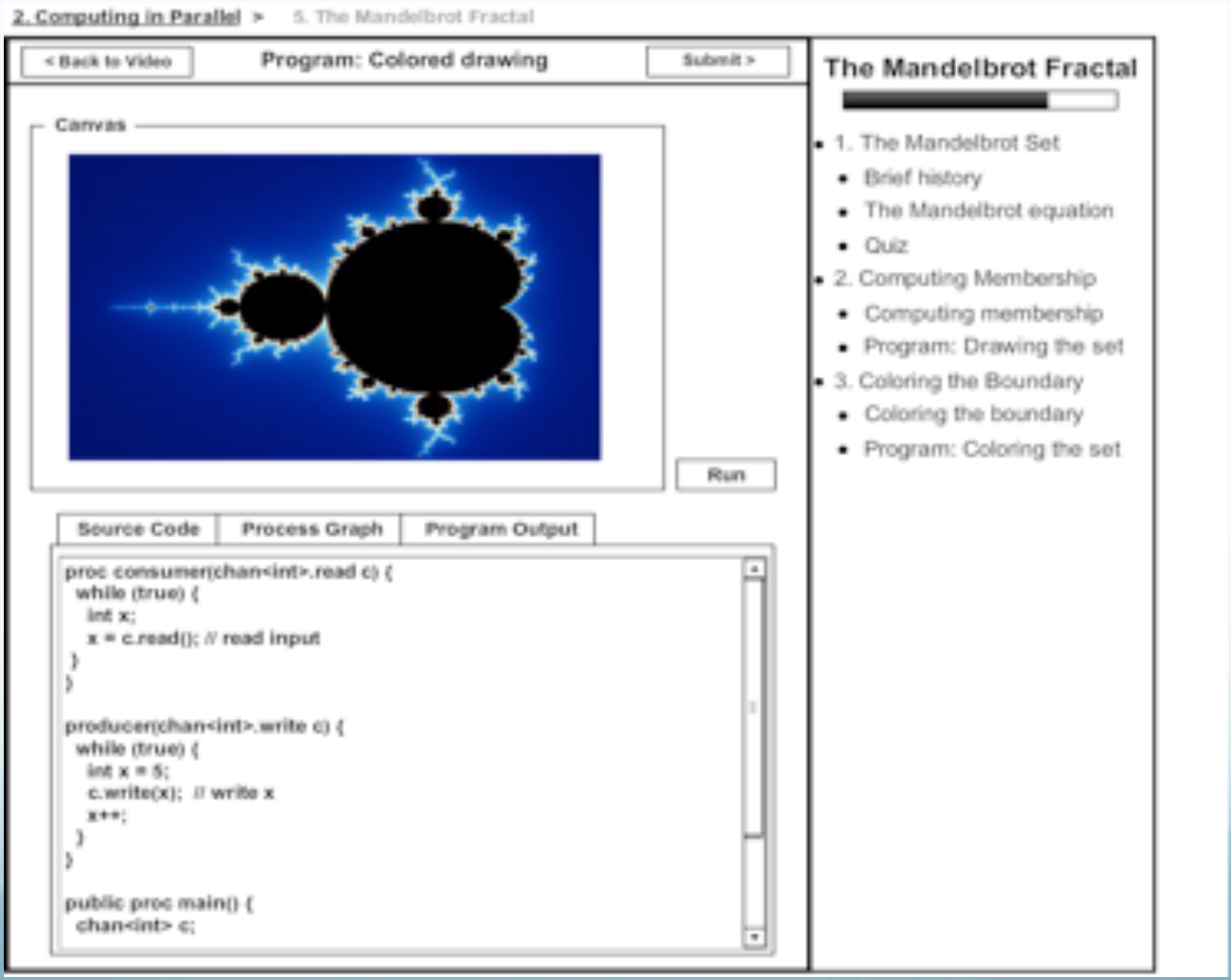


Technical
Educational
IDE/GUI
Process repo.
ProcessJ language
ProcessJ compiler
Runtime (ccsp)
Cloud storage
Executable
CSP scripts
FDR verifier
Web tool (teaching)
Learning material
Course templates
Slides
Videos
Notes
Books
Lecture material

ProcessJ Online Teaching Tool

- Web tool (massive open online course)
  - Written in HTML 5
  - Contains
    - ProcessJ editor
    - Cloud Storage access
    - Process Repository access
  - Compilation done remotely – execution done locally
    - Remote compiler returns JavaScript and a JavaScript CSP runtime.
  - Integrates teaching materials
    - Notes, videos, etc (like codingbat etc.)

ProcessJ Online Teaching Tool



2. Computing in Parallel > 5. The Mandelbrot Fractal

```
proc consumer(chan<int>.read c) {
  while (true) {
    int x;
    x = c.read(); // read input
  }
}

producer(chan<int>.write c) {
  while (true) {
    int x = 5;
    c.write(x);  // write x
    x++;
  }
}

public proc main() {
  chan<int> c;
```

The Mandelbrot Fractal

- 1. The Mandelbrot Set
  - Brief history
  - The Mandelbrot equation
  - Quiz
- 2. Computing Membership
  - Computing membership
  - Program: Drawing the set
- 3. Coloring the Boundary
  - Coloring the boundary
  - Program: Coloring the set

Aliasing Issues

- Data structures for creating graphs etc are important
  - If you can't do what you normally do in the language no one wants it!
- Parallel usage checking might be hard (impossible) to do at compile time
  - Maybe offer runtime checking (expensive, but if you want all the bells and whistles, it will cost you!)
- ????

Us and *Them* (The Environment)

- Paper arose from an NSF proposal (that did not get funded) seeking

  "New programming languages and language mechanisms that support new computational models, raise the level of abstraction, and lower the barrier of entry for parallel and concurrent programming. Parallel and concurrent languages that have programmability, verifiability, and scalable performance as design goals. Of particular interest are languages that abstract away from the traditional imperative programming model found in most sequential programming languages."

- Some of the comments might give an insight into what 'they' think
  - Shows us what the computing community thinks about process oriented design
  - Might help us devise different strategies

Us and *Them* (The Environment)

- "At times, the proposal is talking about CSP as a panacea. Unfortunately, this (in my opinion) reveals ignorance of the richness of both the problem space (types of applications) and the solution space (known models for parallel or concurrent computation), and thus undermines the credibility of success. For example, try writing a well-performing parallel matrix multiplication in CSP."

Us and *Them* (The Environment)

- Matrix multiplication:

```
for (i=0 ; i<n; i++) {
  for (j=0; j<n; j++) {
    C[i][j] = 0;
    for (k=0; k<n; k++) {
      C[i][j] += A[i][j]*B[j][k];
    }
  }
}
```

Us and *Them* (The Environment)

- ProcessJ parallel matrix multiplication:

```
par for (i=0 ; i<n; i++) {
  par for (j=0; j<n; j++) {
    C[i][j] = 0;
    for (k=0; k<n; k++) {
      C[i][j] += A[i][j]*B[j][k];
    }
  }
}
```

- Or a systolic array of n*n processes

Us and *Them* (The Environment)

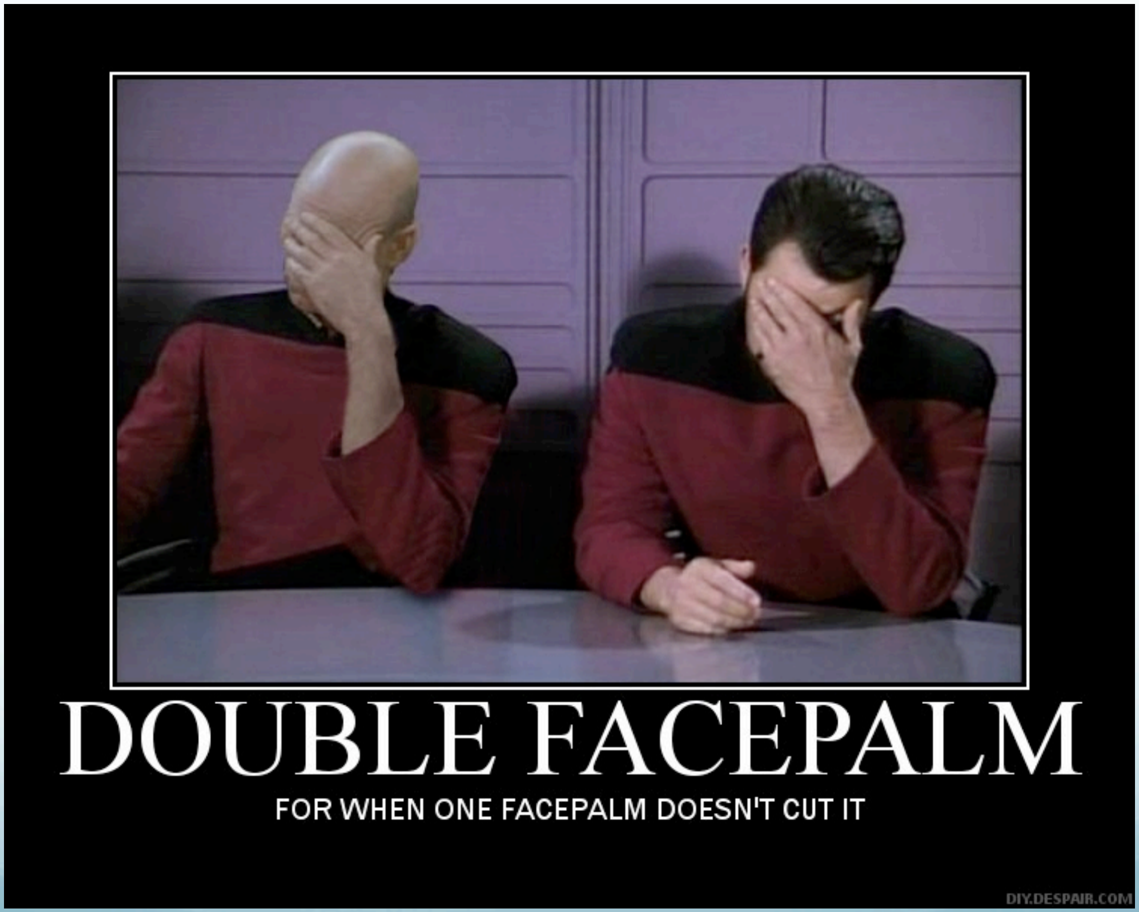
- “The inclusion of a graphical interface / IDE is a nice idea, but orthogonal as a research problem.
- “Graphical IDEs have been around forever, but have never made it beyond the fringe. They tend to survive only if supported by a monopolistic proprietary owner.”
  - This shows a lack of understanding of the power of programming-by-picture and the importance of composability.
  - Learning curve cannot be too high (for the GUI)
  - Used in VLSI for decades!

Us and *Them* (The Environment)

- "Can't you just use Google's Go?"

Us and *Them* (The Environment)

- "Can't you just use Google's Go?"



- Talk amongst yourselves; I'll give you a topic:

"The future of Process Oriented Programming"

- Discuss!!!