Modeling for Everyone

developing high-end production print systems

Hristina Moneva & Eugen Schindler
Océ product families

Océ ColorWave 700 printer
- 500 Posters per day

Océ Arizona 6170 XTS printer
- 200 Signs per day

Océ VarioPrint i300 inkjet press
- More than 3000 m² Billboards/Banners per day
- 25,000 brochures per day

Océ ColorStream 6000 Chroma printer
- Up to 6,000 Books per day

Océ ImageStream 3500 printer
- Up to 125,000 Personalized magazines per day
Printing: more than paper
Breaking down a print system

i300

Controller
Engine
Breaking down a print system

i300

Controller features

- UI
- Job management
- Workflows, planning
- Fleet management
- Color management
- Media management
Breaking down a print system

i300

Engine features

- Brings engine alive with behavior (control, measurement)
- Systems level multi-disciplinary aspects:
  - Productivity
  - Data trends/analysis
  - Variability
  - Error handling
Variability

Same engine, different configuration (input and finishing/output)

Comparable configuration, different engines

Variability within the engine:
1. Paper input module
2. Paper transport module
3. Print module
4. Fixation module
5. Finishing equipment
Technologies

Controller (cut-sheet):
• Single configurable code base
  • C, C++, C#, Java, TypeScript, XML, ...
  • PowerShell, Python, Lua, ... (for build & test)
• Co-developed at five different R&D sites
• Supports ±8 product families
• Supports Océ and Canon engines

Engines:
• Code bases per product family, with a shared reuse architecture and infrastructure
  • C, C++, VHDL, Matlab, RSA-RTE
  • Python and Lua for testing & data science
• Multiple sites, dedicated development of specific engine types

Both:
• Mature development/build/test environment:
  • Mature engineering tools and version control (VS, RSA-RTE, TFS)
  • Automatic deployment of development and test environment
  • Nightly build and test (unit tests, module tests, system test framework)
• MDE for complex control using state machines
  • Synthesis – custom MPS-based tooling (components, generates C++ & Boost), RSA-RTE
  • Analysis – OIL (components & interfaces, proof-of-concept tooling)
Some MBD examples

- Interface specifications
- System capabilities
- Features

Integral productivity
(job, SW, HW, operator behavior, floor plan)

Performance

State Machines
(Component & Interface)

Impositions
(Finishing vs Applications)
Some MBD examples

- Software-in-the-loop simulation
- Physics simulations
- Multi-disciplinary media handling
- Service Training
- Model-based control
The Modeling Manifesto

We are uncovering better ways of developing products by doing it and helping others do it. Through this work we have come to value:

- **Virtual prototypes** over physical models
- **Simple, easy-to-change models** over complex detailed static ones
- **Demonstrable models** over submarine ones
- **Modeling in teams** over lone wolf modeling
- **Connected models** over stand-alone models
- **Self-documented models** over classic documentation

That is, while there is value in the items on the right, we value the items on the left more.
Product Development Vision

Sensible modeling
Not every atom needs modeling

80% -- Risk reduction & architecture verification

20%
There are already a lot of MBD success stories and it is not enough!

“faster, better, cheaper”

But we need more...
Next Steps in Modeling

- Basic modeling established
- Challenge to scale up modeling, make it “boring”:
  - Accessibility of models & tools for “everyone”
  - Proper interfacing and lifecycle management of models
- For many domain models, MPS can help:
Many times a bit = a lot!

- Instead of being too specialistic...
  (e.g. solving complex problems in an optimal way that affect only a few people on the floor)
- Focus on issues that affect everyone.

Everyone understands that

supports the current way of working

Too far away
Make the modeling users into developers

- Instead of being too narrow...
  (e.g. just a selected group of people is able and willing to do it)
- They come with improvements and new ideas

Supports the current way of working

- model simulation
- test generation
- support other languages
- model interfaces
- capabilities

“I want more!”
Connecting models leads to exponential added value

- Instead of being too isolated...
  (e.g. models are developed in isolation)

- impact analysis
- less errors
- better quality
- consistency
- correctness by specification
- faster development
- separation of concerns
- verification
- better communication

“faster, better, cheaper”
Current state: modeling

- SM name
- namespaces
- events
- actions
- states
- initial
- transitions
- event [guard] / action

- boolean parameter
- discarded events
- nested states
- internal behavior “do”
- errors & warnings
- comments

- “auto completion”
- no final state
- no extended states (with variables)
- no orthogonal regions
- no internal behavior entry/exit
- no events deferral
- no external transitions
- no deep history
Current state: modeling
Current state: modeling

```
namespace(s) pressEnterAndStartTyping... :: nsName

events  
etvent1 ( check ) --
etvent2 ( -- ) --
etvent3 ( -- ) --

actions  
action ( -- ) --

//view only, not editable:

<table>
<thead>
<tr>
<th></th>
<th>event1 ( check ) --</th>
<th>event2 ( -- ) --</th>
<th>event3 ( -- ) --</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>-&gt; AnotherState</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>: [ ! check ] / --</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-&gt; NewState</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>: [ check ] / action ()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AnotherState</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-&gt; &lt;no target&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>: [ -- ] / action ()</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-&gt; NewState</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>: [ -- ] / --</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bla (Initial)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-&gt; NewState</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>: [ -- ] / action ()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NewState</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

TABULAR EDITOR

WORK IN PROGRESS
Current state: code generation

*right-click

*generated files

*.h

*.cpp

Reflects the design pattern
Includes custom logging
Can be unit tested
Controller structure

- Reverse engineer our controller architecture from source
Controller structure

- Reverse engineer our controller architecture from source
Controller structure

- Reverse engineer our controller architecture from source
- Query the usage of interfaces and make “views”
Controller structure
Controller structure

- Reverse engineer our controller architecture from source
- Query the usage of interfaces and make “views”
- Analyze the architecture
Controller structure

- Reverse engineer our controller architecture from source
- Query the usage of interfaces and make “views”
- Analyze the architecture
- Generate up-to-date documentation
Interface definitions

- Import/export current IPC/IDL format

```xml
<xml version="1.0" encoding="utf-8"?>
<module xmlns:xa="http://www.w3.org/2001/XMLSchema-instance" name="PrintI"  >
  <metadata>
    <author name="Niels Aan de Brug"/>
    <version major="4" minor="0" patch="0"/>
    <date value="2012-12-12"/>
  </metadata>
  <types>
    <type name="engine_settings"/>
    <type name="devicecontrol"/>
  </types>
  <enum name="failed_result">
    <item name="unspecified"/>
    <item name="connection_lost"/>
    <item name="not_separated"/>
    <item name="separated"/>
  </enum>
  <import_types name="engine_settings"/>
  <import_types name="devicecontrol"/>
</module>
```

```xml
File Name: PrintI.xml
Module Name: PrintI

DOC
This interface is used by clients to print work to an abstract print engine, a print device proxy. The server side handles device specific behavior and protocols.

pre: <no pre>
post: <no post>

METADATA
author: Niels Aan de Brug
version: major 1, minor 0, patch 0
date: 2012-12-12
<no checksum>

IMPORT TYPES
name: engine_settings, subsystem: <no subsystem>
  <no doc>
name: devicecontrol, subsystem: <no subsystem>
  <no doc>
Imported Types
Imported Types are extracted from ID XML and will not be included in the code Generator.
  enginesettings.engine_job_parameters
  devicecontrol.media_tray_info

TYPE
Enum
enum failed_result
  <no doc>
  enum item: unspecified
  <no doc>
  enum item: connection_lost
  <no doc>
  enum item: not_separated
  <no doc>
  enum item: separated
  <no doc>
```
Interface definitions

- Import/export current IPC/IDL format
- Edit textually and graphically
WIP @ controller

• Requirements
• Data modeling
• Workflow modeling
• …
Modular Océ Reusable PlatForm

Electronics as easy configurable as software

Flexibility, TTM, availability

Scalable processor power

Plug-in tiles

Base board
Electronics/Software Interface
Electronics/Software Interface

Project/Board: DEMO_BSI
BLDCmA5 File: PINCH_MOTOR
   slot: 1
tile_name: bldcma5
tile_version: 2.0
motor of DCMotor: PINCH_MOTOR
   pm: PAM
      name: PINCH_MOTOR_PAM
      frequency: 20000
      enable: PINCH_MOTOR_ENA_N
      direction: PINCH_MOTOR_DIR
      inverted: no

enable: SimpleOutput
      name: PINCH_MOTOR_ENA_N
      activeState: ACTIVE_LOW

direction: SimpleOutput
      name: PINCH_MOTOR_DIR
      activeState: ACTIVE_HIGH

hall: HallDecoder
      name: PINCH_MOTOR_HALL

encoder: QuadratureDecoder
      name: PINCH_MOTOR_ENC

fault: SimpleInput
      name: PINCH_MOTOR_FAULT_N
      activeState: ACTIVE_LOW

brake: SimpleOutput
      name: PINCH_MOTOR_BRAKE
      activeState: ACTIVE_HIGH

spi: SPI
      name: PINCH_MOTOR_SPI
      chip select: name: PINCH_MOTOR_EEPROM_CS

VHDL
SW APIs
TSM
Transporting cut-sheets in i300
Mechanics/Software Interface

Mechanical paperpath layout
Mechanics/Software Interface

Mechanical paperpath layout

Mechanical parts
Mech./Software Interface (+Electronics)

Mechanical paperpath layout

Mech. parts

Elec. parts

Embedded Software Interface

Drive Map
“Free-form” 

Read, (mis)interpret 

Discuss
Design

Realization

Verify

Specify

Observe

Develop

Discuss

Timing Designer

Control SW Developer
Timing
Scheduling
C++ / Python / Java / …

Timing Designer =
Domain Expert & Engineer

Rare!

Bottleneck

Programs timing

Thinks about timing

Researcher

Engineer

Domain Expert
Timing Domain Expert

Language Engineer

Timing/ Scheduling model

Simulation (e.g. SIL)

Control (e.g. ESW)

Analysis (e.g. Productivity)

Timing DSLs

Modeling Tool
<table>
<thead>
<tr>
<th>Time</th>
<th>Pos</th>
<th>Vel</th>
<th>Acc</th>
<th>Finch</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20520</td>
<td>23379</td>
<td>1313</td>
<td>105</td>
</tr>
<tr>
<td>2</td>
<td>17764</td>
<td>20027</td>
<td>1274</td>
<td>102</td>
</tr>
<tr>
<td>3</td>
<td>17364</td>
<td>19527</td>
<td>1250</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>16964</td>
<td>19027</td>
<td>1226</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>16564</td>
<td>18527</td>
<td>1200</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>16164</td>
<td>17982</td>
<td>1150</td>
<td>104</td>
</tr>
<tr>
<td>7</td>
<td>15764</td>
<td>17362</td>
<td>1500</td>
<td>124</td>
</tr>
<tr>
<td>8</td>
<td>15365</td>
<td>16745</td>
<td>1536</td>
<td>123</td>
</tr>
<tr>
<td>9</td>
<td>14965</td>
<td>16121</td>
<td>1700</td>
<td>136</td>
</tr>
<tr>
<td>10</td>
<td>14565</td>
<td>15506</td>
<td>1700</td>
<td>136</td>
</tr>
<tr>
<td>11</td>
<td>14166</td>
<td>14871</td>
<td>1231</td>
<td>98</td>
</tr>
<tr>
<td>12</td>
<td>13766</td>
<td>14141</td>
<td>861</td>
<td>69</td>
</tr>
<tr>
<td>13</td>
<td>13366</td>
<td>14163</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>14</td>
<td>12967</td>
<td>13993</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>15</td>
<td>12567</td>
<td>13623</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>16</td>
<td>12167</td>
<td>13353</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>17</td>
<td>11767</td>
<td>13084</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>18</td>
<td>11368</td>
<td>12814</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>19</td>
<td>10968</td>
<td>12544</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>20</td>
<td>10568</td>
<td>12274</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>21</td>
<td>10168</td>
<td>12004</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>22</td>
<td>9768</td>
<td>11735</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>23</td>
<td>9369</td>
<td>11465</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>24</td>
<td>8969</td>
<td>11195</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>25</td>
<td>8569</td>
<td>10925</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>26</td>
<td>8169</td>
<td>10655</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>27</td>
<td>7769</td>
<td>10385</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>28</td>
<td>7369</td>
<td>10115</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>29</td>
<td>6969</td>
<td>9845</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>30</td>
<td>6569</td>
<td>9575</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>31</td>
<td>6169</td>
<td>9305</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>32</td>
<td>5769</td>
<td>8935</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>33</td>
<td>5369</td>
<td>8565</td>
<td>675</td>
<td>54</td>
</tr>
<tr>
<td>34</td>
<td>4969</td>
<td>8195</td>
<td>675</td>
<td>54</td>
</tr>
</tbody>
</table>

Niagara LM2-layout Rev 8.7A MIXED  Time: 111.993  Speed: 56%
Routes

Routelist RouteSpec for ExamplePaperpath

Routes

Route Simplex = ( [input - process - output] )

Sectors

- Sector input : c1, c2
- Sector process : c2 (c3_c2_1), c3, c4
- Sector return : c4, c6, c11, c7, c11, c8, c2
- Sector output : c4, c10

Entries

- entry c1 connector : c1

Destinations

- destination c10 connector : c10

First turn point:

Turn point tolerance (amount of extra segment after a turn point)

0 mm
Job Specifications, Media Catalog, Timing Behavior
Media Actions & Timing, using mbeddr.platform MarginCell/Review...
...and mbeddr C with math notation

```c
exported double AccelerationDistance(double/mmps2/ a_Acceleration, double/mmps/ a_Velocity1, double/m

return \[ \frac{a_{Velocity1}^2 - a_{Velocity2}^2}{2 \times a_{Acceleration}} \];
```

```c
} AccelerationDistance (function)

\[ a_{A} = \frac{-B}{C} \]  

if (l_X < 0) {  
  l_X = 0;
} if

else {

  l_Sq = \sqrt{Q};

  l_X1 = \frac{-a_{B} - l_Sq}{2 \times a_{A}};

  l_X2 = \frac{-a_{B} + l_Sq}{2 \times a_{A}};

  if (l_X1 >= 0) {
```
Timing/ Scheduling model

Simulation (e.g. SIL)

Control (e.g. ESW)

Analysis (e.g. Productivity)

Timing DSLs

Modeling Tool

Timing Domain Expert

Language Engineer

Timing Domain Expert

Language Engineer
Timing/ Scheduling model

Simulation (e.g. SIL)

Timing DSLs

Modeling Tool

Control (e.g. ESW)

Analysis (e.g. Productivity)
Timing/ Scheduling model

Timing DSLs

Modeling Tool

...
Job Specification Niagara Schedule for DL908490Mitsubishi x 701

- Orient sheet with LongEdgeFirst
- Follow route Simplex
- Adhere to timing behavior NiagaraTimingBehavior
- Use scheduler MONOTONIC with period 0.9 s

```java
double decelerationDistance(double vNow, double vOld, double a)
{
    return \( \frac{v_{\text{Now}}^2 - v_{\text{Old}}^2}{2 \cdot a} \);
}
```

Action: Turn (speed vInitial, speed vMax) using <choose optional parameter calculation model>

Entered with speed vInitial
Speed is vMax with LeadingEdge starting at InterfacePoint1107_InterfacePoint1115_10_POWERERKFLAP
Stopped with TrailingEdge at InterfacePoint1225_InterfacePoint1365_14_POWERERKFLAP < 15 mm and with
Flip Edges
Accelerated to speed vMax
Speed is vInitial with LeadingEdge starting at InterfacePoint1107_InterfacePoint1115_10_POWERERKFLAP
```

Route list: Niagara Routes for NiagaraPaperpath

Sectors Sector input: InterfacePoint2473, TM001, InterfacePoint2598, InterfacePoint579
Sector process: InterfacePoint579, InterfacePoint2636, InterfacePoint1
Sector fixation: InterfacePoint1, FIXIERKERRARFLAP, InterfacePoint156
Sector bypass: InterfacePoint1, InterfacePoint156
Sector toOutputRackReturn: InterfacePoint156, 0000LOCKERARFLAP, InterfacePoint156
Sector output: InterfacePoint1588, InterfacePoint1775, InterfacePoint1590, InterfacePoint1297, InterfacePoint1115, InterfacePoint1238, TM001, InterfacePoint2598, InterfacePoint97

Entries Or Entry Connector: InterfacePoint2473
Destinations Exit Connector: InterfacePoint1224
First turn point:
Turn point tolerance (amount of extra segment after a turn point)
0 mm
```

Diagram: Example Paperpath
Variant Management

variability PrinterVariability

imports

Engine

- number ProcessSpeed
- number InterSheetDistance

PM


Feature model

- Fluid
  - Cyan
  - Magenta
  - Yellow

Configuration BlackWhite for Printer

- 3 Attributes
  - M PaperOutput
  - M Engine

- 2 Attributes
  - M OM
  - M FM2
    - 6 Subfeatures
  - M FM1
    - 6 Subfeatures
  - M PM
    - 5 Subfeatures
  - M ITM
  - M Controller
    - M UI
    - M DataHandling
    - M PaperInput
Variants & Modular Design

Layout Module Mod1
Join point <no join>

Layout Module Mod2
Join point <no join>

Layout Module Mod3
Join point <no join>
Variants & Modular Design

Configuration

P2: physical build of type Mod2
Associated with feature FM present in current configuration

P11: physical build of type Mod1
Associated with feature FM1 present in current configuration

P10: physical build of type Mod1
Associated with feature FM2 absent in current configuration

P3: physical build of type Mod3
Associated with feature OCM present in current configuration
Variants & Modular Design
Variants & Functional Decomposition
# Variants & Cost price

## Part List

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Code Number</th>
<th>Cost</th>
<th>Currency</th>
<th>Minimum Quantity</th>
<th>Nr of Circuits</th>
<th>Nr of Wires</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optical sensor</td>
<td>Sensor</td>
<td>133773311</td>
<td>10</td>
<td>USD</td>
<td>250</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Temperature sensor</td>
<td>Sensor</td>
<td>133773312</td>
<td>5</td>
<td>USD</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Pressure sensor</td>
<td>Sensor</td>
<td>133773313</td>
<td>100</td>
<td>USD</td>
<td>28</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Distance sensor</td>
<td>Sensor</td>
<td>133773314</td>
<td>80</td>
<td>USD</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Solenoid</td>
<td>Actuator</td>
<td>133773315</td>
<td>5</td>
<td>USD</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Stepper motor</td>
<td>Actuator</td>
<td>133773316</td>
<td>10</td>
<td>USD</td>
<td>3480</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>BLDC motor A</td>
<td>Actuator</td>
<td>133773317</td>
<td>25</td>
<td>USD</td>
<td>500</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>BLDC motor B</td>
<td>Actuator</td>
<td>133773318</td>
<td>20</td>
<td>USD</td>
<td>5</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>BLDC motor C</td>
<td>Actuator</td>
<td>133773319</td>
<td>50</td>
<td>USD</td>
<td>10</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Heater</td>
<td>Actuator</td>
<td>133773310</td>
<td>15</td>
<td>USD</td>
<td>0</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>
WIP @ embedded

- Formal methods analysis for scheduling
- Functional domain modeling
- Product Line Architecture
MBD landscape of opportunities

**CUSTOMER ENVIRONMENT**
- new opportunities: business, product, technologies

**DOMAIN**
- total aspects
- multi-disciplinary consistency
- product families consistency and variability

**PRINT SYSTEM**
- requirements / features

**USAGE**
- variability
- integral productivity

**CONTROLLER & ENGINE SW**
- design analysis
- impact analysis
- simulation performance

**TEST F.W.**
- specifies features
- generates interfaces

**SOURCE CODE**
- extracts / documents
- simulators
- generates

**COMPONENT**
- behavior spec (components & interfaces)
- data spec
- data constraints
- usage spec

**COMPONENT**
- data + interface

**INTERFACES code**
- State machines code
- Unit tests code
- mocks

**Physics & Chemistry & Mechanical & Electro**
- Design space exploration
- CAD
- CAE
- FEM analysis

**Virtual experiments**

**Engineer**
- Architecting
- Design
- Engineering

**Multi-disciplinary**

**Mono-disciplinary**
Conclusions/lessons so far (1)

• MPS = tool, MPS language design = way of thinking
• Attitude: “not suitable” vs. no “fundamental restriction”
• Small things can block → it’s not a failure, just a lesson
• Understanding instead of selling: go broad (endless possibilities = need to see opportunities)
• A million buts → don’t act desperate/frustrated, be patient → factory vs. garden
• “What’s in it for me?”
• Teach → Support → Fly
• Without mbeddr.platform & MPS, most of it becomes impossible
Conclusions/lessons so far (2)

- JetBrains MPS team open & responsive: good communication on YouTrack
- Forums can be faster, but if you know the right person to ask, that also goes quite well
- Stackoverflow not yet so active
- Overall: courses by JetBrains are recommended as they bring a gentle, but solid introduction to MPS after which potential language engineers realize what MPS can do for them; courses by Itemis really good for more advanced stuff (mbeddr.platform, IETS3, variant management, etc.)
- Need better support for graphical positioning in 2D (e.g. for debuggers); need “Convection in MPS” (web IDE helps!)
It’s an exciting journey!
Questions?

Judge a man by his questions, rather than by his answers

-Voltaire
A CANON COMPANY