Challenges of Trustworthy Human/AI Teaming in Long-running Relationships

NIST ENGINEERING LABORATORY

Peter Denno
Kyoung-Yun Kim
Nicolas Lelouche

Outline

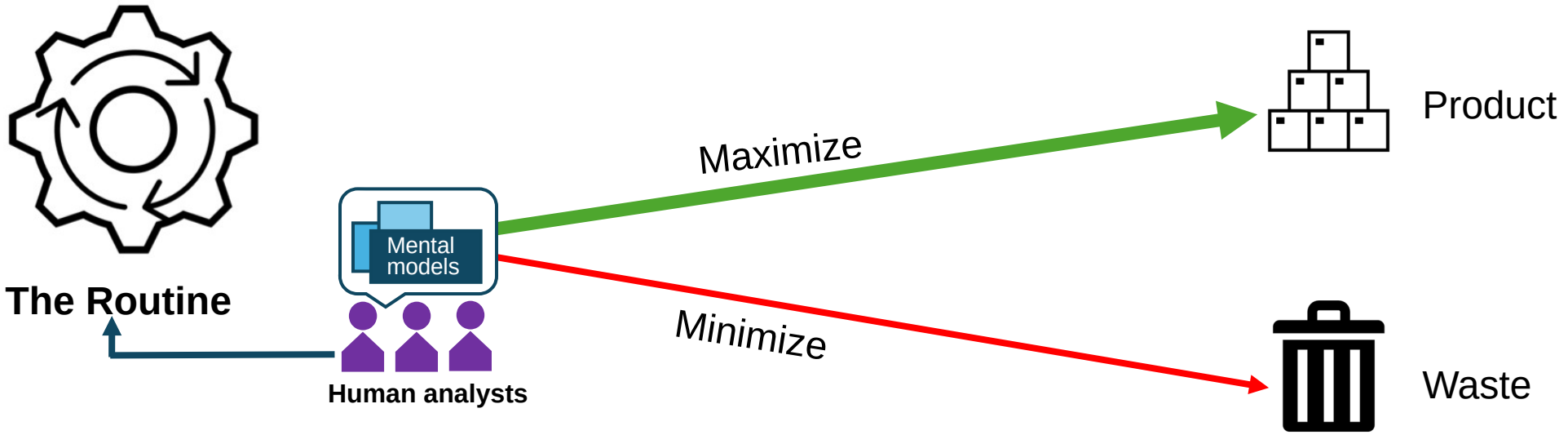
1) Scope
2) Teaming example – schedulingTBD
3) Design for trustworthy teaming
4) Summary and future plans

Teaming is working together

- Teaming on *physical tasks* requires situation awareness.
- Teaming on *cognitive tasks* requires shared understanding.
- Teaming on *long-running cognitive tasks* requires remembering what you committed to and why.

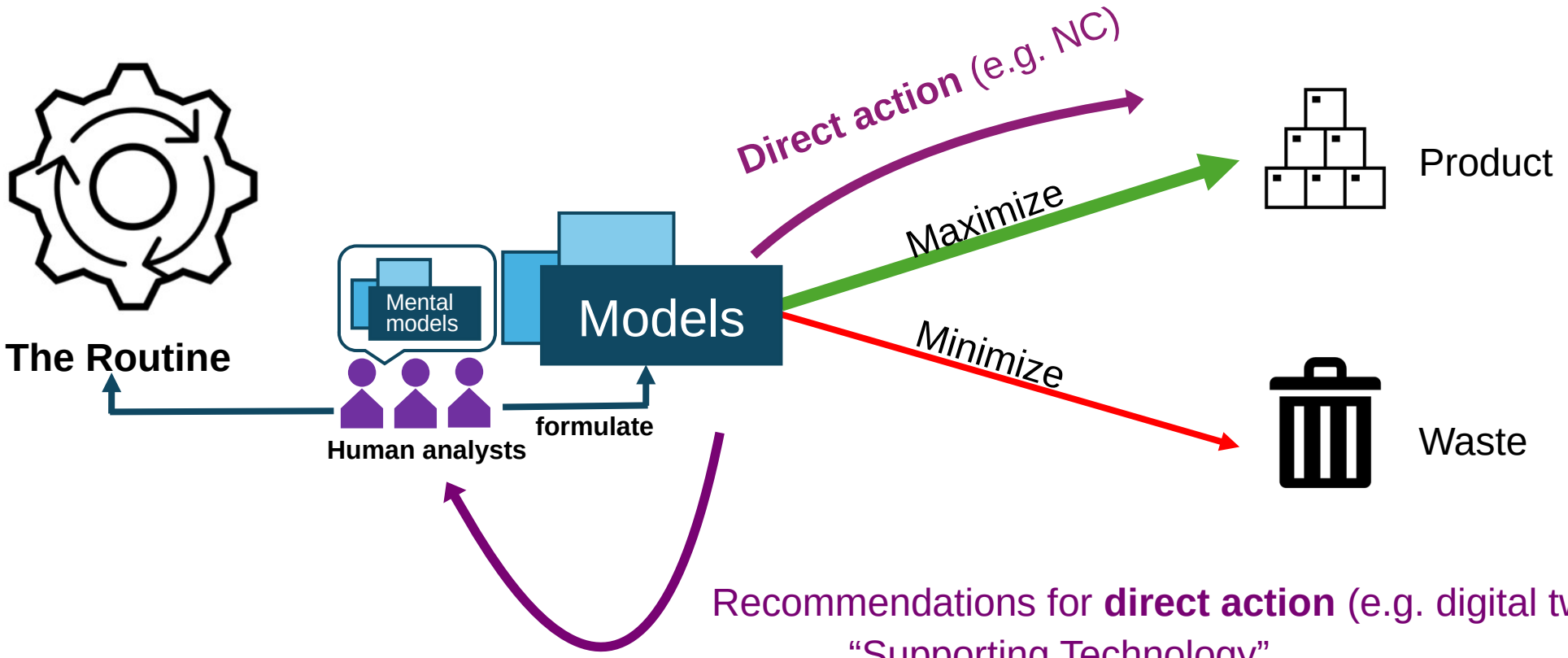
Manufacturing

- **Manufacturing involves a long-running relationship to a routine.**



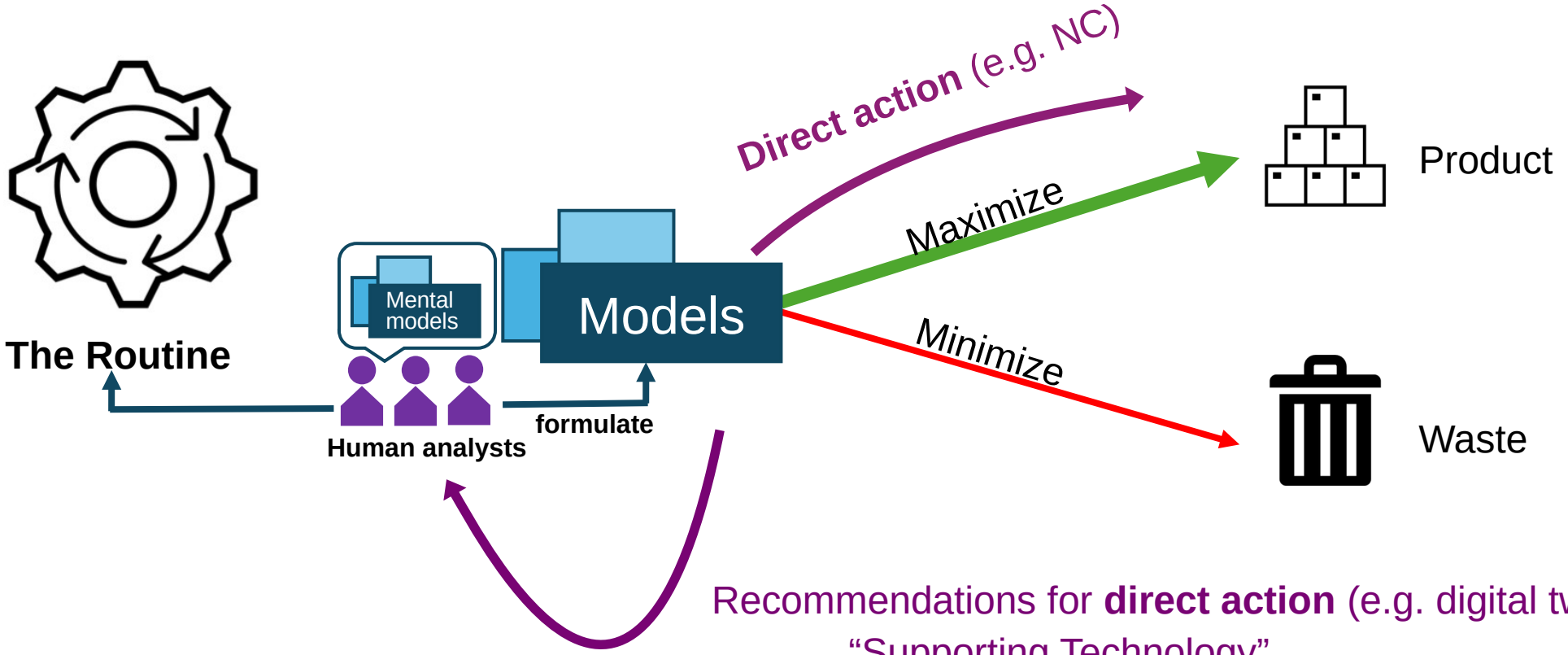
Manufacturing

- **Manufacturing involves a long-running relationship to a routine.**
- **Not all models are mental models.**



Research question

Can HMT help manufacturing experts learn to use manufacturing's *supporting technologies* to improve their productivity and competitiveness?



Recommendations for **direct action** (e.g. digital twins)
"Supporting Technology"

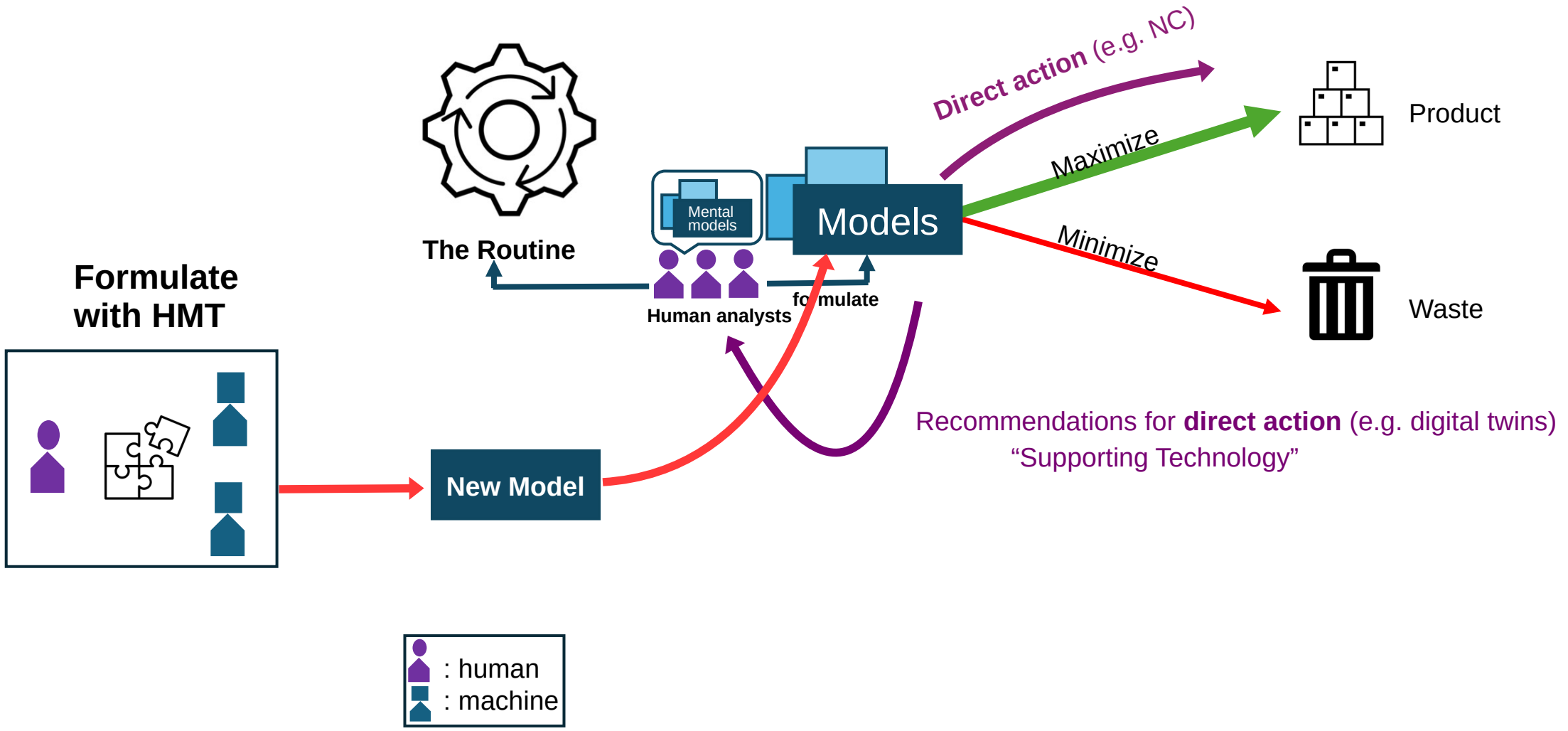
HMT = Human / Machine Teaming

Research question

Can HMT help manufacturing experts *learn* to use manufacturing's *supporting technologies* to improve their productivity and competitiveness?

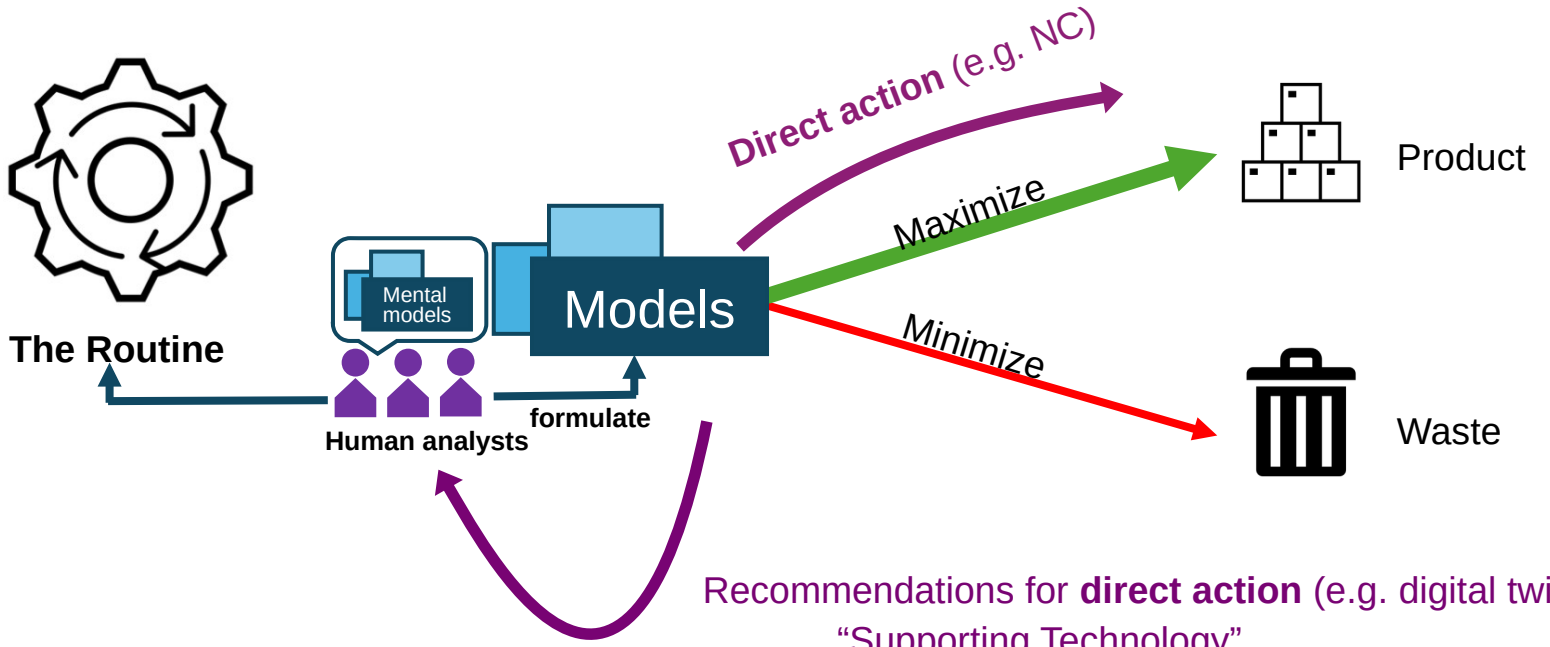
The teaming involves **mentoring human participants** in the use of the supporting technology.

Enhancing Operations with New Supporting Technology using HMT



Possible Applications

- Integrating a digital twin into your production operations
- Integrating interactions with supply chain partners
- Systems engineering of entire systems
- Formulate a solution for some supporting technology using a domain-specific language.
  - Examples: designing digital twins, production scheduling systems,…



Recommendations for **direct action** (e.g. digital twins)
"Supporting Technology"

Outline

1) Scope
2) Teaming example – schedulingTBD
3) Design for trustworthy teaming
4) Summary and future plans

schedulingTBD

Sur Fountain Pens

Process

Resources

Data

Interviewer, Mon Aug 12 2024

I suppose processing times for each of the steps you just mentioned might vary from product to product. But generally speaking, how long does each step take? Please produce a list just like the one you did for process steps, one process per line, but append to each line the typical processing time in parentheses.

Surrogate Expert

1. Material Preparation (2 hours)
2. Nib Fabrication (1 hour)
3. Barrel Fabrication (1.5 hours)
4. Cap Fabrication (1 hour)
5. Ink Converter Fabrication (0.5 hours)
6. Assembly (2 hours)
7. Quality Control (1 hour)
8. Packaging (0.5 hours)
9. Shipping (1 hour)

Interviewer, Mon Aug 12 2024

Okay, we now know enough to get started on a MiniZinc solution. In the code pane (upper right of the app) we created a simplistic scheduling system.It only illustrates the idea of running one job through each of the tasks you mentioned(excepting any tasks that weren't part of making the product, those we'll deal with later.

Interviewer, Mon Aug 12 2024

Earlier, you listed the process steps typically used in making product (e.g. 1. Material Preparation, 2. Nib Fabrication, etc.) and for each you specified typical durations.Now we'd like you to tell us what raw materials and

Type message here....

```
int : nProducts = 1;
set of int: Product = 1..nProducts;
enum Task = {materialPrep, nibFabrication, barrelFabrication, capFabrication, inkConverterF
% Durations are in units of hours and may be rounded.
array [Product, Task] of float: taskDuration = [|2.000, 1.000, 1.500, 1.000, 0.500, 2.000,

float : minTime = 0;
% It shouldn't take longer than doing only one thing at a time.
float : maxTime = sum (p in Product, t in Task) (taskDuration[p, t]);

% We assume a task uses one major resource; resources are synonymous with tasks.
array [Product, Task] of var minTime..maxTime: taskStarts; % 'var' means it is a decision v
array [Product, Task] of var minTime..maxTime: taskEnds;

% This is used in the constraint below.
predicate no_overlap(var float:s1, float:d1, var float:s2, float:d2) = s1 + d1 <= s2 \/ s2

% Ensure that no two Products aren't using the same resource (task) at the same time.
constraint forall (t in Task)
            (forall  (i,k in Product where i < k)
               (no_overlap (taskStarts[i,t], taskDuration[i,t], taskStarts[k,t], taskDurat

% We assume no tasks are running now, so the first task of some product can start now.
constraint exists (p in Product) (taskStarts[p, materialPrep] = 0);

% Starting some time in the past is not allowed.
constraint forall (p in Product where p > 1) (taskStarts[p, nibManufacturing] > 0);

% Every other task can start after the previous task ends.
constraint forall (p in Product) (taskEnds[p, materialPrep] == taskStarts[p, nibFabricatior
constraint forall (p in Product) (taskEnds[p, nibFabrication] == taskStarts[p, barrelFabric
constraint forall (p in Product) (taskEnds[p, barrelFabrication] == taskStarts[p, capFabric
constraint forall (p in Product) (taskEnds[p, capFabrication] == taskStarts[p, inkConverter
constraint forall (p in Product) (taskEnds[p, inkConverterFabrication] == taskStarts[p, ass
constraint forall (p in Product) (taskEnds[p, assembly] == taskStarts[p, qualityControl]);
constraint forall (p in Product) (taskEnds[p, qualityControl] == taskStarts[p, packaging]),

% A task ends taskDuration time units after it starts.
constraint forall (p in Product, t in Task) (taskEnds[p, t] == taskStarts[p, t] + taskDurat

% Minimize time elapsed when the final task ends (maxspan).
solve minimize max (p in Product) (taskEnds[p, packaging]);
```

schedulingTBD

Sur Fountain Pens

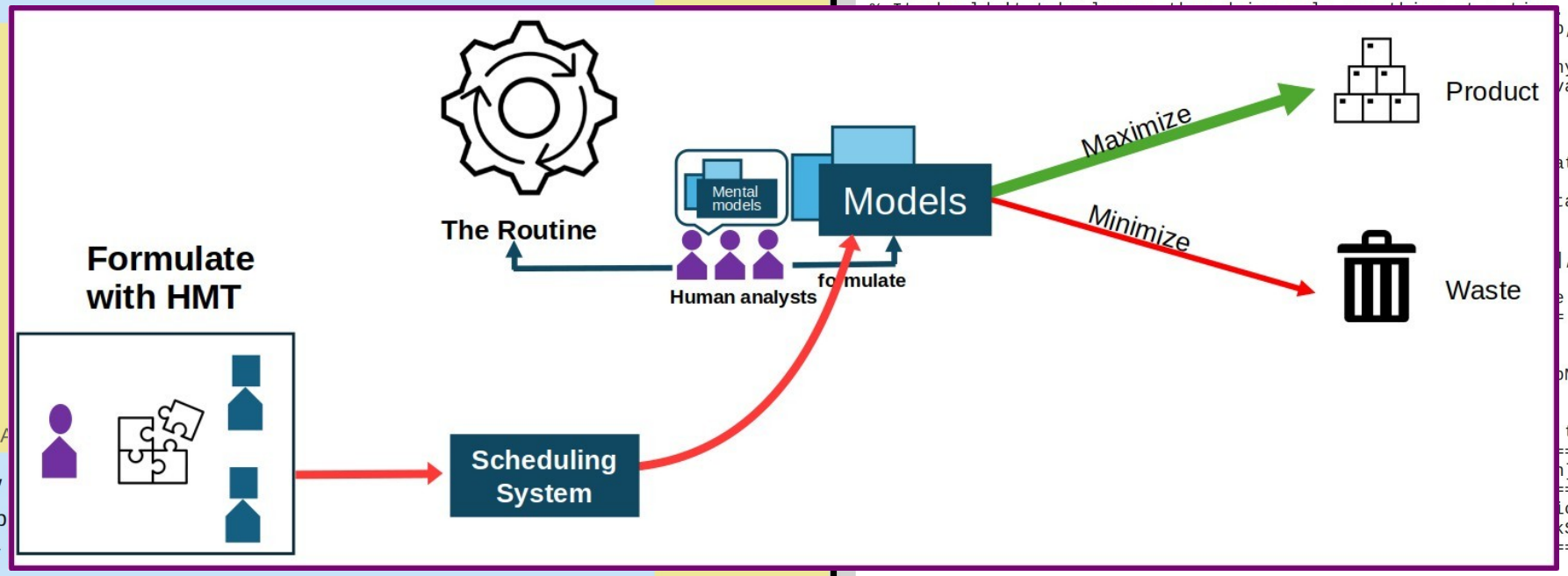
Process

Resources

Data

Interviewer, Mon Aug 12 2024

I suppose processing times for each of the steps you just mentioned might vary from product to product. But generally speaking, how long does each step take? Please produce a list just like the one you did for process steps, one process per line, but append to each line the typical processing time in parentheses.



Interviewer, Mon A

Okay, we now
code pane (up
system.It only
tasks you mentioned(excepting any tasks that weren't part of making the product, those we'll deal with later.

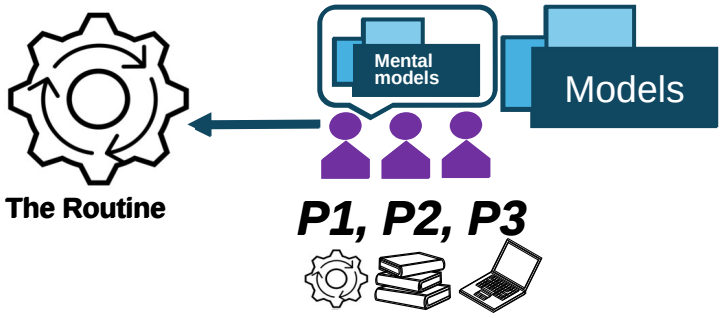
Interviewer, Mon Aug 12 2024

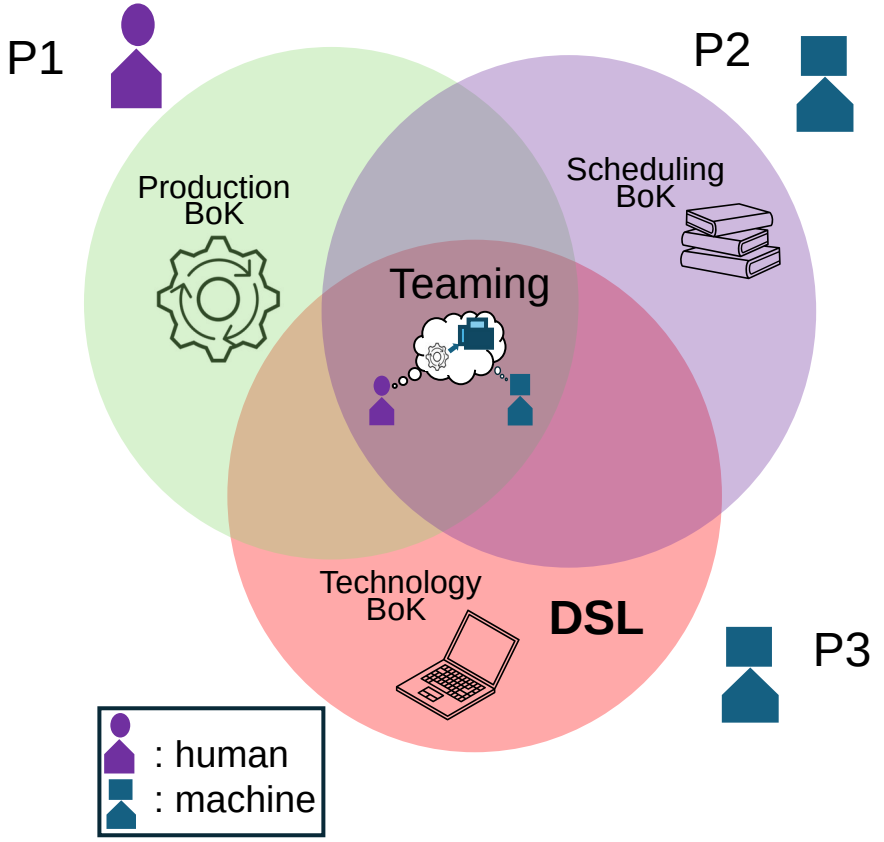
Earlier, you listed the process steps typically used in making product (e.g. 1. Material Preparation, 2. Nib Fabrication, etc.) and for each you specified typical durations.Now we'd like you to tell us what raw materials and

Type message here....

```
int : nProducts = 1;
set of int: Product = 1..nProducts;
enum Task = {materialPrep, nibFabrication, barrelFabrication, capFabrication, inkConverterF
% Durations are in units of hours and may be rounded.
array [Product, Task] of float: taskDuration = [|2.000, 1.000, 1.500, 1.000, 0.500, 2.000,

float : minTime = 0;
                                                                                , t]);
                                                                                ymous with tasks.
                                                                                var' means it is a decision v
                                                                                t:d2) = s1 + d1 <= s2 \/ s2
                                                                                task) at the same time.
                                                                                , taskStarts[k,t], taskDurat
                                                                                e product can start now.
                                                                                = 0);
                                                                                bManufacturing] > 0);
                                                                                taskStarts[p, nibFabricatio
                                                                                == taskStarts[p, barrelFabric
                                                                                ] == taskStarts[p, capFabric
                                                                                == taskStarts[p, inkConverte
                                                                                ication] == taskStarts[p, ass
                                                                                kStarts[p, qualityControl]);
                                                                                == taskStarts[p, packaging]),
% A task ends taskDuration time units after it starts.
constraint forall (p in Product, t in Task) (taskEnds[p, t] == taskStarts[p, t] + taskDurat

% Minimize time elapsed when the final task ends (maxspan).
solve minimize max (p in Product) (taskEnds[p, packaging]);
```

Factoring the problem of creating a scheduling system





- **Imagine 3 personae working together:**

  P1 knows the company's production processes well.

  P2 knows a technical/scholarly body of knowledge (BoK) useful to helping ***P1*** with a task.

  P3 knows how to express the BoK in some technology.

schedulingTBD Sur Fountain Pens

Process

Resources

Data

Interviewer, Mon Aug 12 2024

I suppose processing times for each of the steps you just mentioned might vary from product to product. But generally speaking, how long does each step take? Please produce a list just like the one you did for process steps, one process per line, but append to each line the typical processing time in parentheses.

Surrogate Expert

1. Material Preparation (2 hours)
2. Nib Fabrication (1 hour)
3. Barrel Fabrication (1.5 hours)
4. Cap Fabrication (1 hour)
5. Ink Converter Fabrication (0.5 hours)
6. Assembly (2 hours)
7. Quality Control (1 hour)
8. Packaging (0.5 hours)
9. Shipping (1 hour)

Interviewer, Mon Aug 12 2024

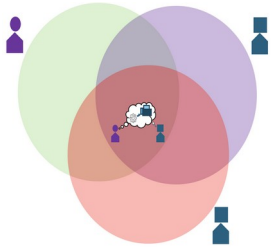
Okay, we now know enough to get started on a MiniZinc solution. In the code pane (upper right of the app) we created a simplistic scheduling system.It only illustrates the idea of running one job through each of the tasks you mentioned(excepting any tasks that weren't part of making the product, those we'll deal with later.

Interviewer, Mon Aug 12 2024

Earlier, you listed the process steps typically used in making product (e.g. 1. Material Preparation, 2. Nib Fabrication, etc.) and for each you specified typical durations.Now we'd like you to tell us what raw materials and

Type message here....

```
int : nProducts = 1;
set of int: Product = 1..nProducts;
enum Task = {materialPrep, nibFabrication, barrelFabrication, capFabrication, inkConverterF
% Durations are in units of hours and may be rounded.
array [Product, Task] of float: taskDuration = [|2.000, 1.000, 1.500, 1.000, 0.500, 2.000,

float : minTime = 0;
% It shouldn't take longer than doing only one thing at a time.
float : maxTime = sum (p in Product, t in Task) (taskDuration[p, t]);

% We assume a task uses one major resource; resources are synonymous with tasks.
array [Product, Task] of var minTime..maxTime: taskStarts; % 'var' means it is a decision v
array [Product, Task] of var minTime..maxTime: taskEnds;

% This is used in the constraint below.
predicate no_overlap(var float:s1, float:d1, var float:s2, float:d2) = s1 + d1 <= s2 \/ s2

% Ensure that no two Products aren't using the same resource (task) at the same time.
constraint forall (t in Task)
            (forall  (i,k in Product where i < k)
               (no_overlap (taskStarts[i,t], taskDuration[i,t], taskStarts[k,t], taskDurat

% We assume no tasks are running now, so the first task of some product can start now.
constraint exists (p in Product) (taskStarts[p, materialPrep] = 0);

% Starting some time in the past is not allowed.
constraint forall (p in Product where p > 1) (taskStarts[p, nibManufacturing] > 0);

% Every other task can start after the previous task ends.
constraint forall (p in Product) (taskEnds[p, materialPrep] == taskStarts[p, nibFabricatio
constraint forall (p in Product) (taskEnds[p, nibFabrication] == taskStarts[p, barrelFabric
constraint forall (p in Product) (taskEnds[p, barrelFabrication] == taskStarts[p, capFabric
constraint forall (p in Product) (taskEnds[p, capFabrication] == taskStarts[p, inkConverte
constraint forall (p in Product) (taskEnds[p, inkConverterFabrication] == taskStarts[p, as
constraint forall (p in Product) (taskEnds[p, assembly] == taskStarts[p, qualityControl]);
constraint forall (p in Product) (taskEnds[p, qualityControl] == taskStarts[p, packaging])

% A task ends taskDuration time units after it starts.
constraint forall (p in Product, t in Task) (taskEnds[p, t] == taskStarts[p, t] + taskDurat

% Minimize time elapsed when the final task ends (maxspan).
solve minimize max (p in Product) (taskEnds[p, packaging]);
```



■ : Production process body of knowledge

■ : Scheduling body of knowledge

■ : Software tools body of knowledge

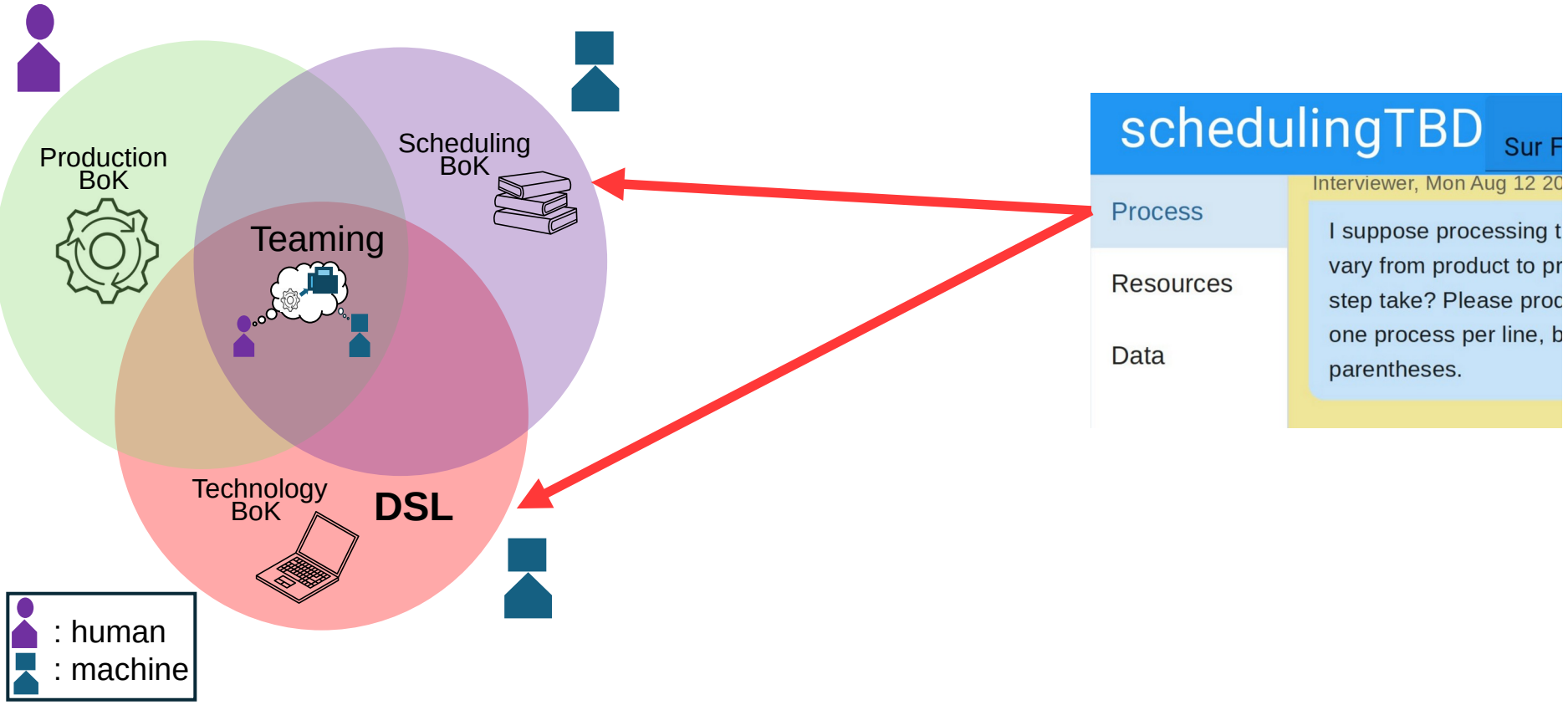
Outline

1) **Scope**
2) **Teaming example – schedulingTBD**
3) **Design for trustworthy teaming**
4) **Summary and future plans**

Outline

1) **Scope**
2) **Teaming example – schedulingTBD**
3) **Design for trustworthy teaming**
   - **The interview follows an expert's plan.**
   - **Use a DSL.**
   - **Use hidden conversations.**
   - **Use intermediate models.**
   - **Test with surrogate users.**
4) **Summary and future plans**

Interviewers are Experts at Something



The Interview is a Dynamic Planning Problem

- Currently using traditional hierarchical task network (HTN) planner.
- Planning domain encodes expert knowledge.
  - What question to ask next
- Dynamic (on-line) planning
- One (Chain-of-Thought) agent type for each Q/A pair.
- Result of Q/A analysis is facts influencing later planning.

- Idea for future: RAG-based orchestration agent might be more robust.

Elements of HTN Planning Domain

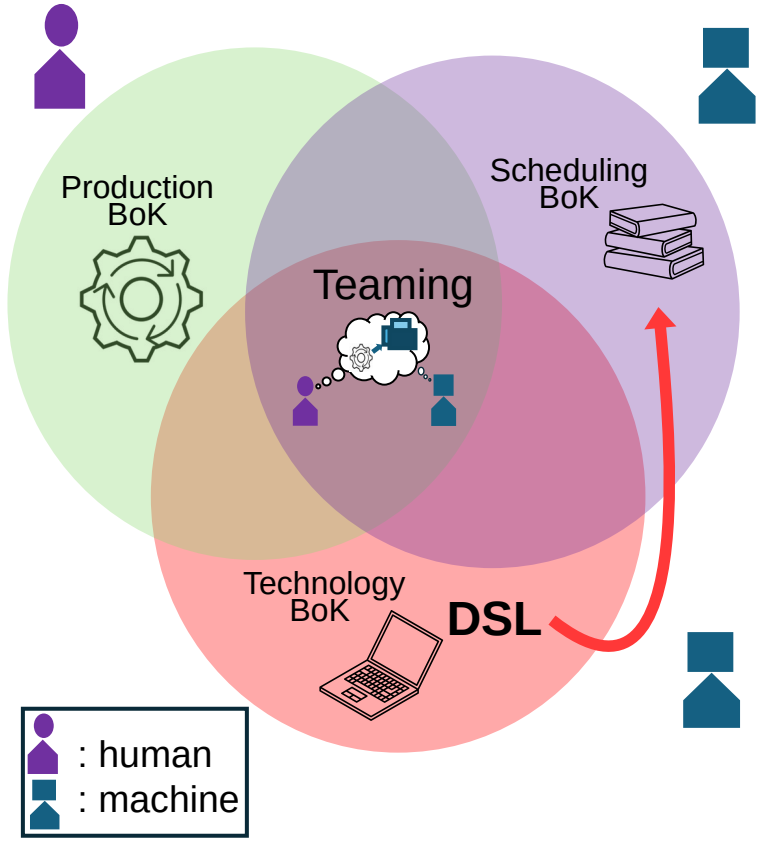
```
{:method/head (characterize-process ?p)                              ; Top-level goal.
 :method/rhsides [{:method/preconds [(proj-id ?p)]                   ; Needs only a project-id to run.
                   :method/task-list [(describe-challenge ?p)        ; Methods run sequentially.
                                      (classify-production-type ?p)
                                      (analyze-process ?p)
                                      (verify-process ?p)]}]}

{:method/head (describe-challenge ?p)
 :method/rhsides [{:method/preconds [(proj-id ?p)]
                   :method/task-list [(!describe-challenge ?p)                ; Call to an operator.
                                      (remark-raw-material-challenge? ?p)]}]} ; Call to a method.

{:method/head (remark-raw-material-challenge? ?p)
 :method/rhsides [{:method/case "cited"
                   :method/preconds [(cites-raw-material-challenge ?p)]
                   :method/task-list [(!remark-raw-material-challenge ?p)]}
                  {:method/case "not-cited"
                   :method/preconds [(not (cites-raw-material-challenge ?p))]
                   :method/task-list [(!do-nothing)]}]}

{:operator/head (!describe-challenge ?p)}                ; Plan leaves are operators, first Q/A pair.
{:operator/head (!remark-raw-material-challenge ?p)}
```

Outline

1) Scope
2) Teaming example – schedulingTBD
3) Design for trustworthy teaming
   - The interview follows an expert's plan.
   - Use a DSL.
   - Use hidden conversations.
   - Use intermediate models.
   - Test with surrogate users.
4) Summary and future plans

Why use a domain-specific language?

- Constructionist learning (learn by making)
- DSLs are
  - small languages
  - application-focused
  - ideally declarative (about what not now)
- In mentoring, use the DSL
  - to anchor concepts
  - as an on-ramp to a community of practice.



Peter Denno, *Cognitive Work in Future Manufacturing Systems: Human-centered AI for Joint Work with Models*, Journal of Integrated Design & Process Science, 2023

Outline

1) **Scope**
2) **Teaming example – schedulingTBD**
3) **Design for trustworthy teaming**
   - **The interview follows an expert's plan.**
   - **Use a DSL.**
   - **Use hidden conversations.**
   - **Use intermediate models.**
   - **Test with surrogate users.**
4) **Summary and future plans**

“Hidden” conversations

Human interviewee

AI interviewer of human

AI surrogate expert interviewee

AI interviewer of surrogate

AI models manager

“What is your scheduling challenge?”

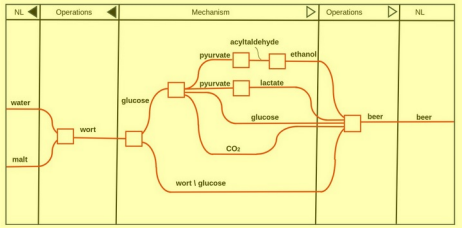
“We make X...”

You are an expert at making X.

“naïve” Q/A discussion about making X

“Make a model of this."

“Make a model of that."

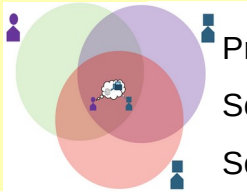


“Here is a MiniZinc model of the problem”

“Is this [what human said] consistent with our models?”

“No. Ask for clarification.”

[clarifying question]



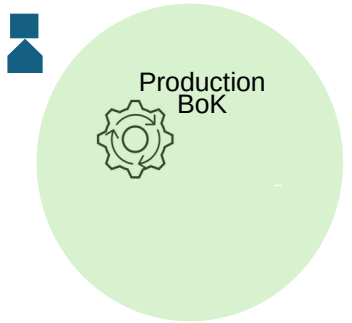
Production process body of knowledge

Scheduling body of knowledge

Software tools body of knowledge

Hidden Conversations: Motivation

- Manufacturing spans a diverse body of knowledge.
- Make an LLM-based agent play the role of expert.
- Collect *default knowledge* through this *parallel conversation*.
- Use the default knowledge to eliminate excessive questions.



Interviewer, Mon Aug 12 2024

A FLOW-SHOP is a production system designed so that all jobs follows the same sequence of steps through production resources. A JOB-SHOP is a production system where each job might follow its own route, depending on its unique requirements. Is the process you described more like a flow-shop or a job-shop? Respond respectively with either the single term FLOW-SHOP or JOB-SHOP.

Surrogate Expert
FLOW-SHOP

Outline

1) **Scope**
2) **Teaming example – schedulingTBD**
3) **Design for trustworthy teaming**
   - The interview follows an expert's plan.
   - Use a DSL.
   - Use hidden conversations.
   - Use intermediate models.
   - Test with surrogate users.
4) **Summary and future plans**

Intermediate Models:
The Problem Being Addressed

- Conventional SE V&V benefits from methods that are *previewable*.
  - Previewable = Possible to *dissect and inspect* the method before use.
- LLM-based agents (e.g. assistants) easily delegated to tasks.
  - But delegation is the antithesis of previewability.
  - Explainable AI (XAI) explains w.r.t algorithms and training data, not a belief about the rationality of the delegated agent.
- Consequently, a shortcoming in V&V.

Intermediate Models: A Way Forward?

- Apply Chain-of-Thought (CoT) reasoning on unstructured text from conversation, transforming and refining information to graphs and hypergraphs.
  - CoT isn’t quite previewable, but it is close.
- Progressively transform what is learned in conversation to graphical forms viewable (and possibly executable) by humans for V&V.
  - BPMN
  - SysML
  - Modelica
  - ...

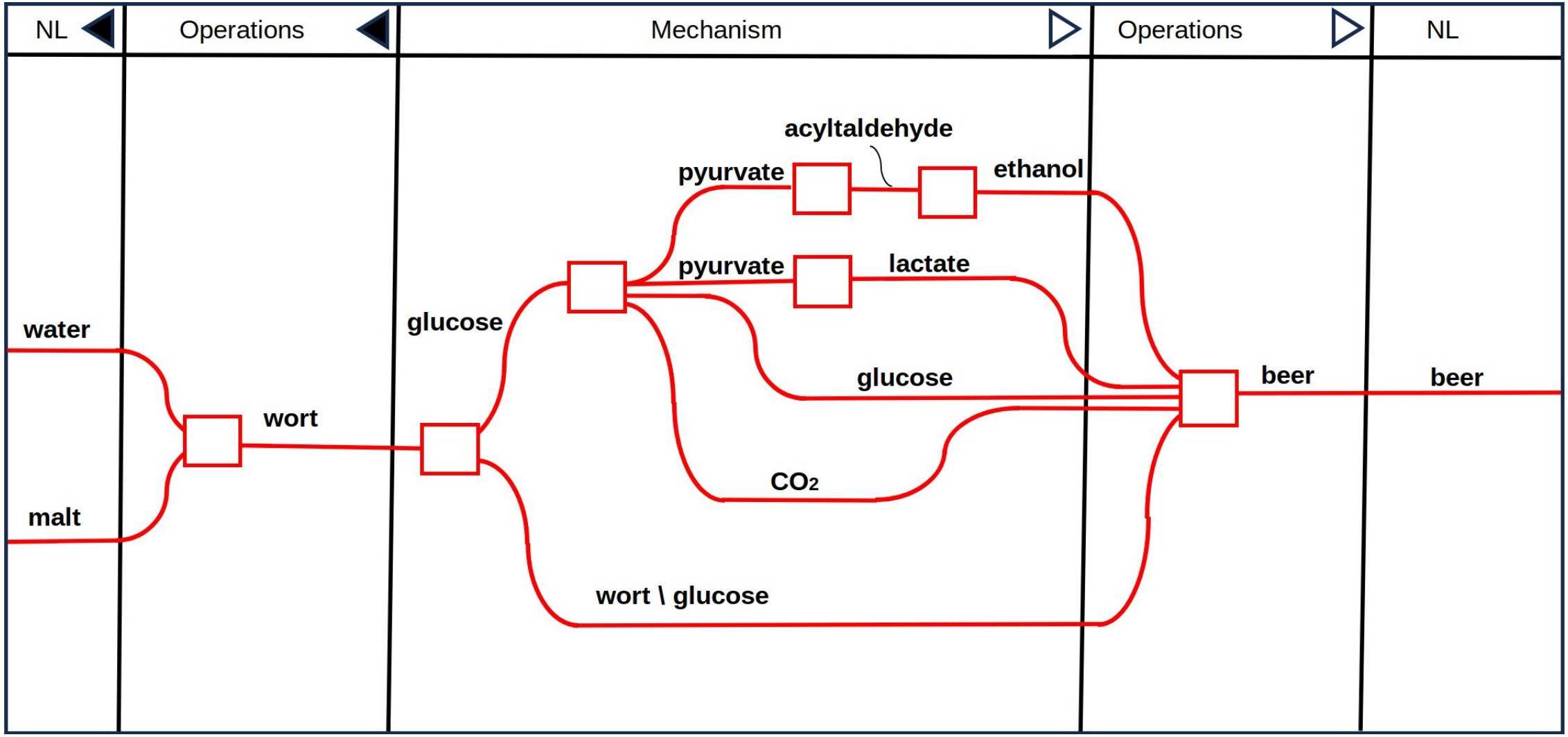
Intermediate Models:
Text to Knowledge Graph by Chain-of-Thought Transformation

```
You are a helpful assistant. You read user data and output an interpretation of it as a vector
of Clojure maps conforming to requirements expressed below. The requirements are enacted in
phases named 'revisions', (REV-1, REV-2..., REV-5). We will ask for interpretation to revision
REV-i data using input that is in the form of REV-i-1 output. This allows us to check your work
at each revision before proceeding to the next. We may ask for the same revision twice
consecutively; this is especially true if the first answer you provided did not meet
specifications. The data we are working with typically concerns manufacturing processes. Here
is example input to REV-1, which is the only input that is not a vector of Clojure maps:

1. Raw Material Acquisition - 1 day to 3 months
2. Material Preparation - 2 days
3. Component Machining - 5 days
4. Assembly - 3 days
5. Quality Control (QC) - 1 hour (but could go longer)
6. Packaging - 1 day
7. Shipping - Varies depending on where it is going

In REV-1 you separate each line of text into three parts: process step, process name, and
duration; you create a Clojure map of this information. The keys you should use in the map
are respectively :PROCESS-STEP :PROCESS and :DURATION (upper-case Clojure keys). Specifically,
the value of map key :PROCESS-STEP is the number starting the string. The value of map key
:PROCESS is a string describing the process step. The value of map key :DURATION is a string
of the remaining information in the line, typically describing the duration:

The REV-1 output that you should produce from the REV-1 input above is:

[{:PROCESS-STEP 1 :PROCESS "Raw Material Acquisition" :DURATION "1 day to 3 months"},
 {:PROCESS-STEP 2 :PROCESS "Material Preparation" :DURATION "2 days"},
 {:PROCESS-STEP 3 :PROCESS "Component Machining" :DURATION "5 days"},
 {:PROCESS-STEP 4 :PROCESS "Assembly" :DURATION "3 days"},
 {:PROCESS-STEP 5 :PROCESS "Quality Control (QC)" :DURATION "1 hour (but could go longer)"},
 {:PROCESS-STEP 6 :PROCESS "Packaging" :DURATION "1 hour"},
 {:PROCESS-STEP 7 :PROCESS "Shipping" :DURATION "Varies depending on where it is going"}]

REV-2, REV-3, and REV-4 are not shown. Respectively, they isolate commentary, interpret the
duration strings, and flag processes that concern supply chain, rather than work done
in-house. The results of these transformations are shown in REV-5 results.

In REV-5, we add a camelCase string suitable for use as a program variable that describe the
:PROCESS. The variable should be concise, no more than 15 characters long or so and should
start with a lowercase letter, a-z. In the map, we use the keyword :VAR to identify this
camelCase string. Suitable output for REV-5 is shown below:
```

```
[{:PROCESS-STEP 1 :PROCESS "Raw Material Acquisition"
                  :DURATION {:QUANTITY-LOW  {:AMOUNT-STRING "1" :UNITS :days}
                             :QUANTITY-HIGH {:AMOUNT-STRING "3" :UNITS :months}},
                  :SUPPLY-CHAIN? true, :VAR "getRawMaterial"},
                  :VAR "machining"},
 {:PROCESS-STEP 4 :PROCESS "Assembly" :DURATION {:AMOUNT-STRING "3" :UNITS :days},
                  :VAR "assembly"},
 {:PROCESS-STEP 5 :PROCESS "Quality Control (QC)" :DURATION {:AMOUNT-STRING "1" :UNITS :hours}
                  :COMMENT "(but could go longer)", :VAR "qualityCtrl"},
 {:PROCESS-STEP 6 :PROCESS "Packaging" :DURATION {:AMOUNT-STRING "1" :UNITS :days},
                  :VAR "packaging"},
 {:PROCESS-STEP 7 :PROCESS "Shipping" :DURATION "varies"
                  :COMMENT "Varies depending on where it is going", :SUPPLY-CHAIN? true,
                  :VAR "shipping"}]
```

See https://github.com/pdenno/schedulingTBD/blob/main/data/instructions/process-dur-agent.txt

Interrelating Conversation
(a goal currently unrealized)



Lobski & Zanasi, String Diagrams for Layered Explanation, 2024.

“Hidden” conversations

Human interviewee

AI interviewer of human

AI surrogate expert interviewee

AI interviewer of surrogate

AI models manager

“What is your scheduling challenge?”

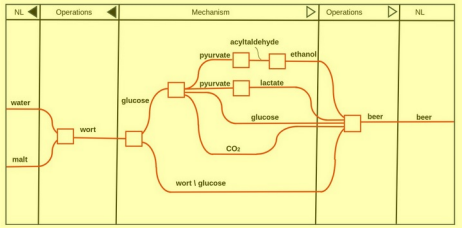
“We make X...”

You are an expert at making X.

“naïve” Q/A discussion about making X

“Make a model of this."

“Make a model of that."

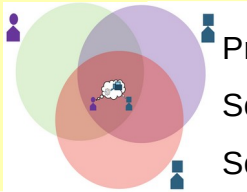


“Here is a MiniZinc model of the problem”

“Is this [what human said] consistent with our models?”

“No. Ask for clarification.”

[clarifying question]



Production process body of knowledge

Scheduling body of knowledge

Software tools body of knowledge

Outline

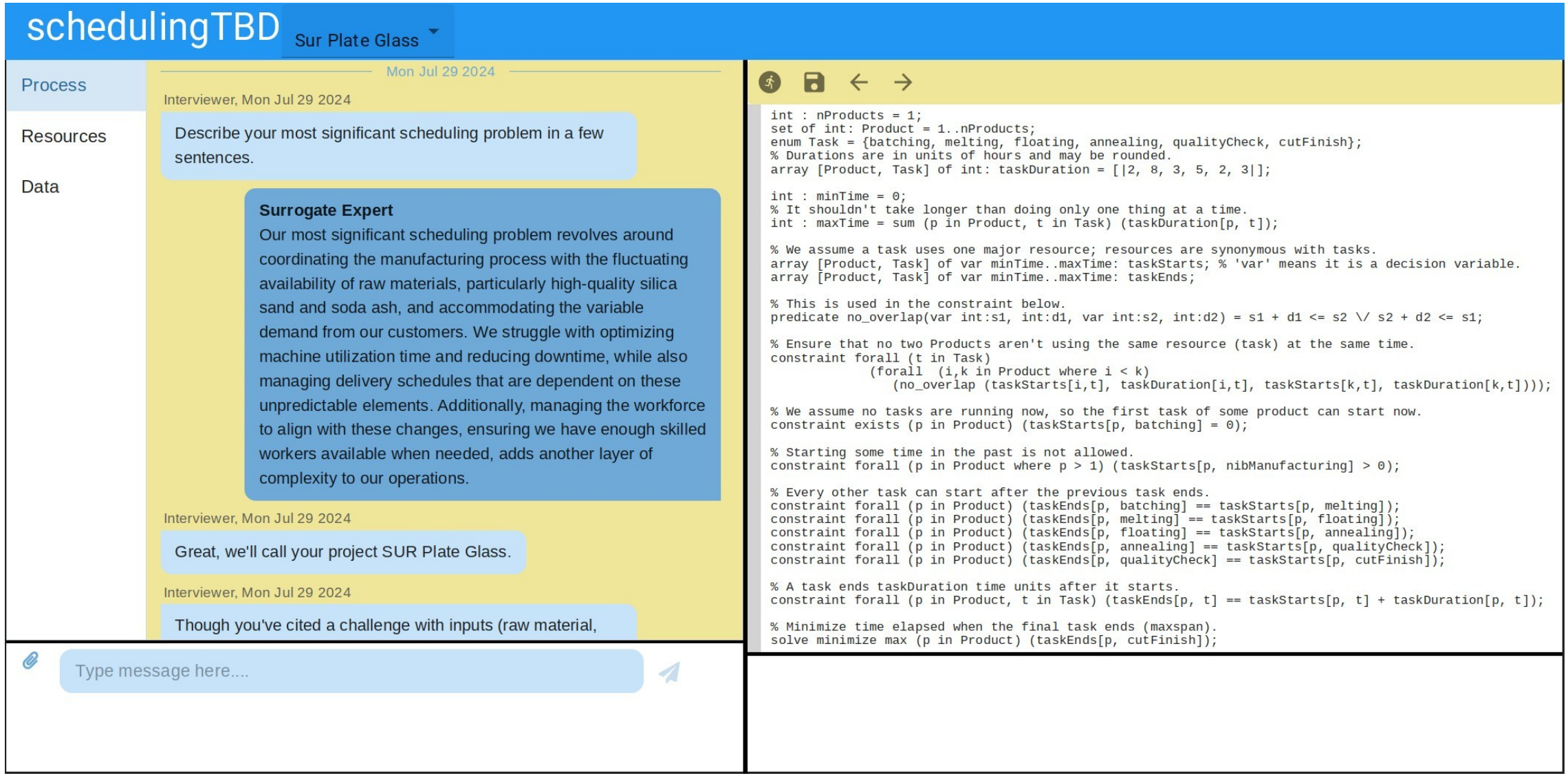
1) **Scope**
2) **Teaming example – schedulingTBD**
3) **Design for trustworthy teaming**
   - **The interview follows an expert's plan.**
   - **Use a DSL.**
   - **Use hidden conversations.**
   - **Use intermediate models.**
   - **Test with surrogate users.**
4) **Summary and future plans**

Testing: Science Channel's *How It's Made*

- **Where do you find enough experts?**
- **Canadian documentary series showing how every-day items are manufactured**
- **1600 episode segments**
  - We're using them as test cases (i.e. the $Xs).

```
"You manage a company that makes $X.  You are an expert in production and manage your
company's supply chains.  You help me by answering questions that will allow us to collaborate
in building a scheduling systems for your company. Your answers typically are short, just a
few sentences each.  If you don't have information to answer my questions, you provide a
plausible answer nonetheless."
```

Example Surrogate Run



Outline

1) **Scope**
2) **Teaming example – schedulingTBD**
3) **Design for trustworthy teaming**
4) **Summary and future plans**

Summary Design Recommendations

- Captures expert knowledge in the interview plan
- Use agents prevasively.
  - The new way to factor a problem
  - RAG for agent-based conversation planner?
- Use a well-conceived DSL
- V&V by Chain-of-thought analysis to intermediate models
  - Background surrogate conversation for
- Factor conversation into areas / personae.
  - For example, process, resources, data

What comes next

- **Complete exploratory software for human use**
  - Wayne State testing with students
  - Demonstrate effectiveness of intermediate analytical work products

- **ISO/IEC JTC1 / SC42 (AI) WG 4 (Use cases)**
  - Contributing to New Work Item 42109 "Human/Machine Teaming“
  - Meeting at NIST with WG4 convener and editors (Japan delegation)

- **A pilot for adaptive integration of digital twins**