

Above there should be the title slide, but sometimes it does not appear.

It reads “Design Choices for Teaming on Complex and Long-running Endeavors”
Peter Denno 2025-09-17 & 18

(For SERC “AI4SE & SE4AI” 2025 Workshop)

- Research in
 - Manufacturing systems integration
 - Human/AI teaming in manufacturing.
- Informing standards and guidelines.
- Targeting support for small and medium-size manufacturers in the use of the complex tools of advanced manufacturing.

Why is NIST at this workshop?

We do research in advanced manufacturing.

The research I am presenting today is about how humans and AI could team to solve manufacturing problems, such as production scheduling and process planning. Specifically in this presentation we'll discuss an interviewing techniques that NIST developed to facilitate teaming in long-running and complex endeavors.

There is an unprecedented opportunity now to drive down the cost of integrating manufacturing equipment, a very large expense for US industry. A somewhat similar idea (self-integrating systems) has had currency in our division for many years, however, until recently it couldn't be easily implemented.

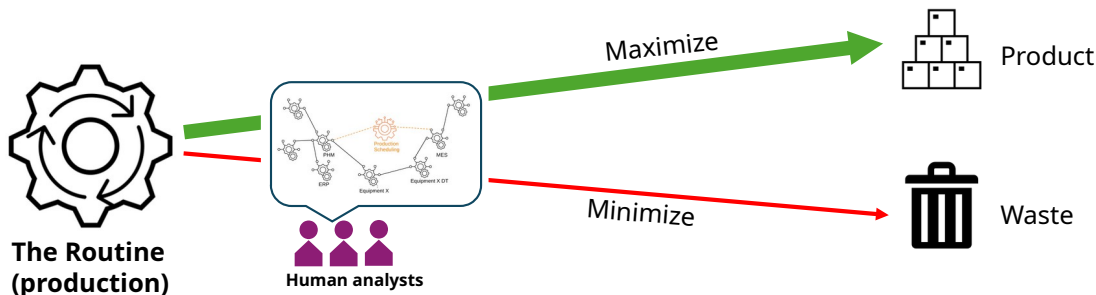
We are active in ISO/IEC JTC 1 SC42. We have developed exportatory software to inform our standards work. We are investigating the possibility of its use with small and medium-sized manufacturers through MEP.

We think the work has relevance beyond manufacturing.

- Intro: human/AI teaming to support use of complex tools
- AI-led interviews to understand system environment and goals
- Summary

Establish the context and some basic terminology

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



4

- What do I mean by “long-running and complex endeavors?”
- Endeavor = to work with a set purpose, for example in this slide, production of product and reduction of waste is the endeavor.
- Long-running – 1 week to 10 years
- Complex – coupling of effects, non-linearity, emergent behaviors, path dependence, uncertainty propagation, no decomposition to sub-problems.
- Here the endeavor is the **design and use of a tool**. Imagine our early ancestors creating a tool by tying a rock to a stick. They use it to mash grain. They’d only bother to make the tool because they intend to use it over, time again (long-running). The quality of the tool and the skill with which it is used determines how much goes into product vs waste.

(long-running, but not complex).

-
- The diagram illustrates the process of optimizing a production routine. It starts with a gear icon labeled "The Routine (production)". An arrow points from this gear to a box labeled "Rules defining the routine (automation)". Above this box, three human figures are labeled "Human analysts", with an arrow pointing down to the box. A callout bubble above the analysts shows a network diagram with nodes labeled "Equipment A", "Equipment B", "Equipment C", and "Equipment D", connected by lines, with a central orange gear icon labeled "Production Management System". From the "Rules defining the routine (automation)" box, two arrows emerge: a green arrow labeled "Maximize" pointing to a stack of four cubes labeled "Product", and a red arrow labeled "Minimize" pointing to a trash can icon labeled "Waste".

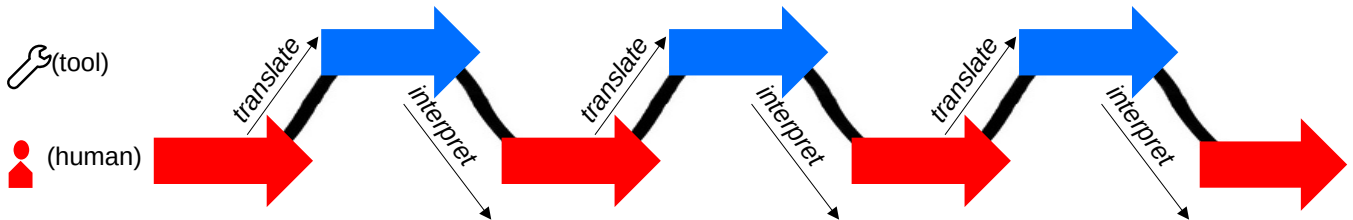
Many billions of dollars is spent on this each year.


Human/AI Teaming

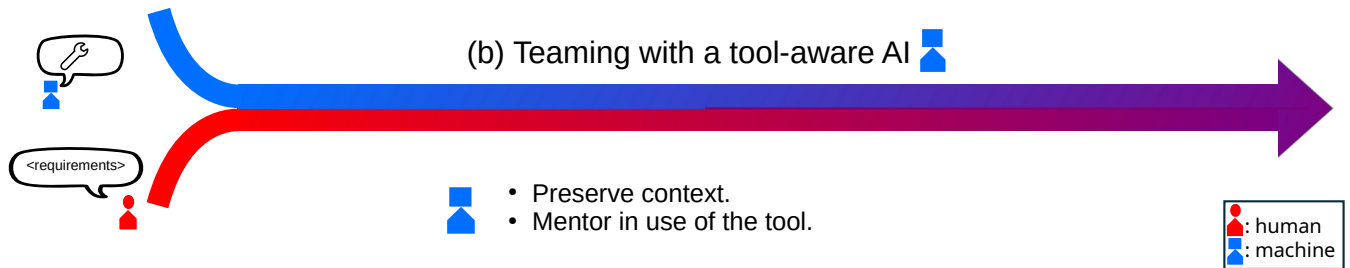
(as a better way to use complex tools)

NIST

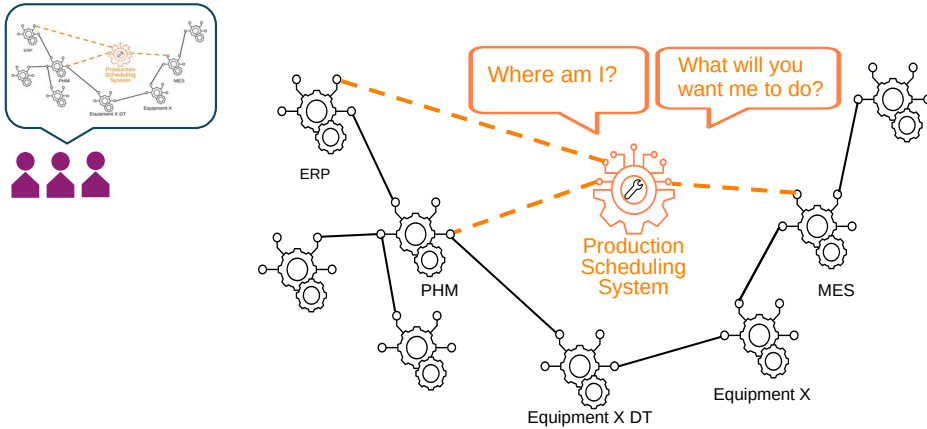
(a) Typical work with complex tools 



(b) Teaming with a tool-aware AI 



- Imagine having a person on your team who understands some esoteric but powerful tool useful to the team and, listening to your discussion, can map discussion into use of that tool.
- Bring the amplitude way down, and blend in – becoming purple!
- But this is all figurative. How can it work in reality?
- We first look at one place where it is really needed: manufacturing.



How do we create a new capability and integrate it into the system? The orange is an envisaged capability (artistically depicted) suggesting that the production scheduling system would embed a tool and talk to the ERP, PHM and MES systems. We aim for the orange tool to be a reusable entity that you can add to your production system. In simplistic terms, it integrates into your production environment by answering the questions “Where am I?” and “What will you want me to do?” (in run time). Answering those questions adequately should exercise much of systems engineering best practice.

The remainder of the presentation is about practices we’ve developed to create and integrate a capability. We used production scheduling, but similar should be possible to create many such embedded systems.

Terminology



tool – a computational capability with associated domain-specific language (DSL)

Examples:

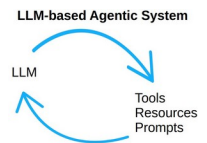
Problem types	Capability	DSL
scheduling, resource allocation, supply chain optimization, portfolio selection	constraint programming, mathematical optimization, combinatorial search	MiniZinc
dynamic system behavior, physical system simulation, control system design	multi-domain physical modeling, differential-algebraic equation solving, continuous/hybrid simulation	Modelica
structural consistency, design validation, configuration verification	relational logic modeling, constraint satisfaction, counterexample finding	Alloy



tool steward – an agentic system that wraps a tool to enable its integration into target systems through requirements discovery and configuration.

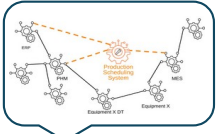




situated tool steward – a tool steward that, through human/AI teaming with system participants, has (1) expressed system-specific requirements using the tool's DSL, and (2) created an API translating between the system's operational needs and the tool's capabilities.



Note: [In case you forgot to say this earlier.] Neither the tool steward, nor the situated tool steward have any role in the run-time operation. Tool stewards help with systems engineering and systems integration in a system enhancement project, then get out of the way until the next project comes along. Motivating this design choice are security concerns about agentic systems in industrial operations.

Initiating SE with a Tool Steward (assumptions)



- Your goal: add a new capability to your system (e.g. **production scheduling**).
- There is a suitable tool steward for that capability. 
- You help the tool steward answer the two questions.
 - This launches human/steward teaming on a systems engineering project.
 - Steward mentors in use of the tool's DSL and related body of knowledge (BoK).
 - Steward keeps a record of everything you do together.
 - Result: a situated tool steward and new system capability ready to use. 
 - You can return to the conversation to refine the system at any time.

DSL = domain-specific language, a declarative formal language (visual or lexical) that allows one to express key concepts in a domain of study.
Some DSL-based tools support inference, providing ability to simulate behavior and support decisions.

Recall from a few slides back collection of people who were considering adding more sophisticated scheduling to their production operations (maybe they were working FIFO before)

And assuming that there is a tool steward for production scheduling, you would embark together with the agentic system on a systems engineering project to develop and integrate the new capability.

Remember: neither tool steward (situated or not) has any role in the run-time system.

But the situated tool steward is always there when you decide an opportunity exists to **further improve** outcomes. For example, you start getting ASN for raw materials.

schedulingTBD Sur Craft Beer

Surrogate Expert

1. Milling Grains (500 gallons)
2. Mashing (500 gallons)
3. Lautering (500 gallons)
4. Boiling (500 gallons)
5. Fermentation (500 gallons)
6. Conditioning (500 gallons)
7. Filtering (500 gallons)
8. Carbonation (500 gallons)
9. Bottling/Canning (500 gallons, typically yielding around 4000 to 4500 standard-sized bottles/cans)
10. Packaging (4000 to 4500 units)

Interviewer, Mon Nov 18 2024

Earlier, you listed the process steps typically used in making product (e.g. 1. Milling Grains, 2. Mashing, etc.) and for each you specified typical durations. Now we'd like you to tell us what raw materials and intermediate product go into those process steps. For example, if you were making sandwich cookies, you might simultaneously make the dough for the wafers and make the filling. You might then place the dough in molds and bake to produce wafers. With that done, you would add some filling to one wafer and place another wafer on top. Were the cookie bakers to create a list we seek from you, using their process step list (which had 5 steps) it would look like this:

1. Make Cookie Dough (flour, water, eggs, sugar, chocolate chips)
2. Make Filling (sugar, water, vanilla flavoring)
3. Bake Wafers (use dough from Make Cookie Dough)
4. Assemble Cookies (use wafers from Bake Wafers, use filling from Make Filling)
5. Package (use cookies from Assemble Cookies)

Surrogate Expert

1. Milling Grains (barley, wheat, malt)
2. Mashing (milled grains from Milling Grains, water)
3. Lautering (mash from Mashing)
4. Boiling (wort from Lautering, hops, any additional flavorings or adjuncts as needed)
5. Fermentation (boiled wort from Boiling, yeast)
6. Conditioning (beer from Fermentation)
7. Filtering (conditioned beer from Conditioning)
8. Carbonation (filtered beer from Filtering, CO2 if force carbonated)
9. Bottling/Canning (carbonated beer from Carbonation, bottles/cans, caps)
10. Packaging (bottled/canned beer from Bottling/Canning, packaging materials like boxes or cases)

```

int : nProducts = 1;
let of int: Product = 1..nProducts;
enum Task = {millingGrains, mashing, lautering, boiling, fermentation, conditioning, filtering, carbonation, bottlingCanning, packaging};
% Durations are in units of hours and may be rounded.
% If any of the durations are negative values, we had problems understanding; you should fix these by hand.
array (Product, Task) of float: taskDuration = [[1.000, 1.500, 1.000, 1.000, 252.000, 60.000, 2.000, 24.000, 3.000, 3.000]];

float : minTime = 0;
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 variable.
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 + d2 <= s1;

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

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

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

% Every other task can start after the previous task ends:
constraint forall (p in Product) (taskEnds[p, millingGrains] == taskStarts[p, mashing]);
constraint forall (p in Product) (taskEnds[p, mashing] == taskStarts[p, lautering]);
constraint forall (p in Product) (taskEnds[p, lautering] == taskStarts[p, boiling]);
constraint forall (p in Product) (taskEnds[p, boiling] == taskStarts[p, fermentation]);
constraint forall (p in Product) (taskEnds[p, fermentation] == taskStarts[p, conditioning]);
constraint forall (p in Product) (taskEnds[p, conditioning] == taskStarts[p, filtering]);
constraint forall (p in Product) (taskEnds[p, filtering] == taskStarts[p, carbonation]);
constraint forall (p in Product) (taskEnds[p, carbonation] == taskStarts[p, bottlingCanning]);
constraint forall (p in Product) (taskEnds[p, bottlingCanning] == taskStarts[p, packaging]);

% A task ends taskDuration time units after it starts.

```

Type message here...

This is a screen shot of our exploratory software for production scheduling geared for use by small and medium-sized manufacturers.

sTBD is a tool steward. When you start talking to it about scheduling <something>, it becomes a situated tool steward situated in <something>.

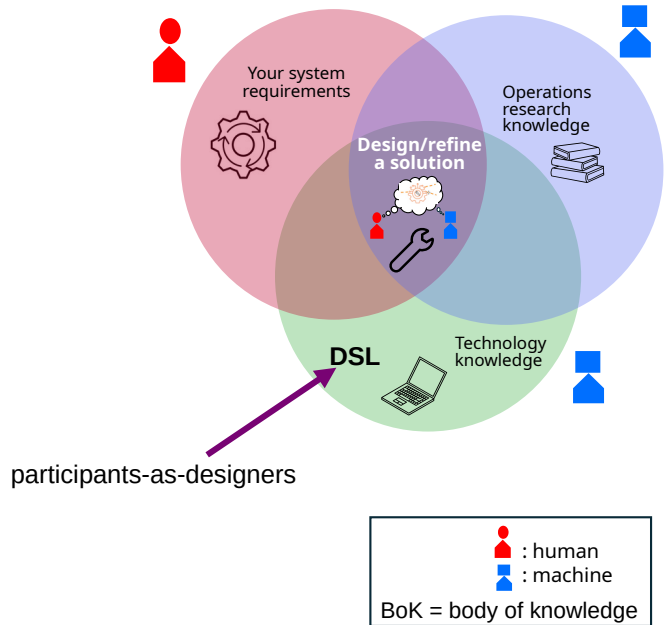
[Describe parts. Magnify DSL for segue to next slide.]

[BTW, We will change this slide from being about craft beer to something else. Probably re-man alternators. I just can't do it on this laptop with the connection I have.]



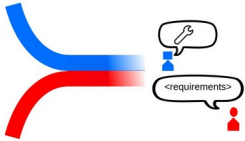
Requirements

- Participants **know**
 - the existing production system and processes,
 - what they want to achieve with a new project.
- Participants **need not know**
 - how the wrapped tool works
 - the BoK underlying the tool (e.g. combinatorial optimization)
 - the BoK underlying the disciplines (e.g. scheduling, operations research)
 - software development & programming
- Participants **shall leave the teaming experience knowing**
 - how the system uses their data in the tool's primary DSL,
 - (possibly) knowing a little bit more about the underlying BoKs.



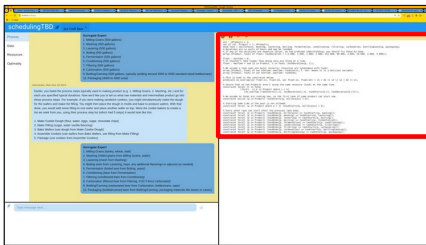
[This is a bit repetitive of the assumptions slide.
Maybe that's a good thing??]

Emphasize use of DSL. We are gambling a bit on DSLs being an important part of HMT in industrial settings.
[The design and integration capabilities of tool stewards is going to be rather straightforward in comparison, I think.]



Examples

- *Modelica*
- *MiniZinc*
- *Alloy...*



- Domain-specific Languages (DSLs)
 - Key point of interaction!
 - To translate requirements to a solution
 - To avoid a black box
 - To verify captured understanding
 - To share with others
 - To record development history
- Qualities of good DSLs for teaming
 - Declarative
 - Either inclusive of all needs or compositional
 - Large community of practice

12

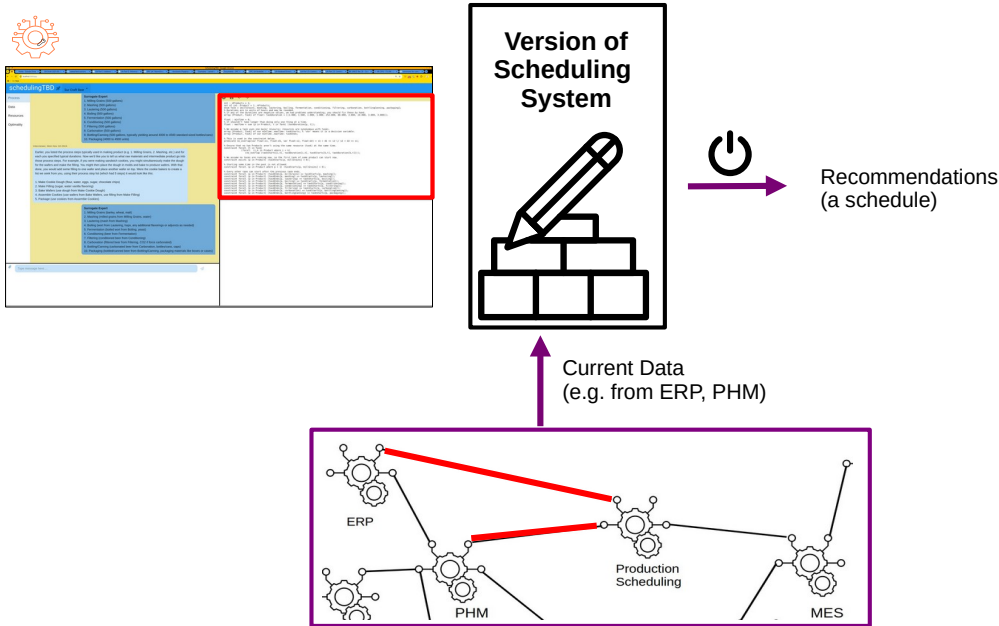
We are interested in tools with DSLs because we believe them to be useful to teaming and mentoring in the use of sophisticated tools.

[This is the final slide of the “concepts and terminology” part.]

- Intro: human/AI teaming to support use of complex tools
- AI-led interviews to understand system environment and goals
- Summary

We discuss AI-led interviewing because it is the key capability of the functions of the tool steward. By answering the questions “Where am I?” and “What will you want me to do?” you help integrate the tool capability into your operations. You situate a tool steward.

How does it work? (when you run the scheduling system)



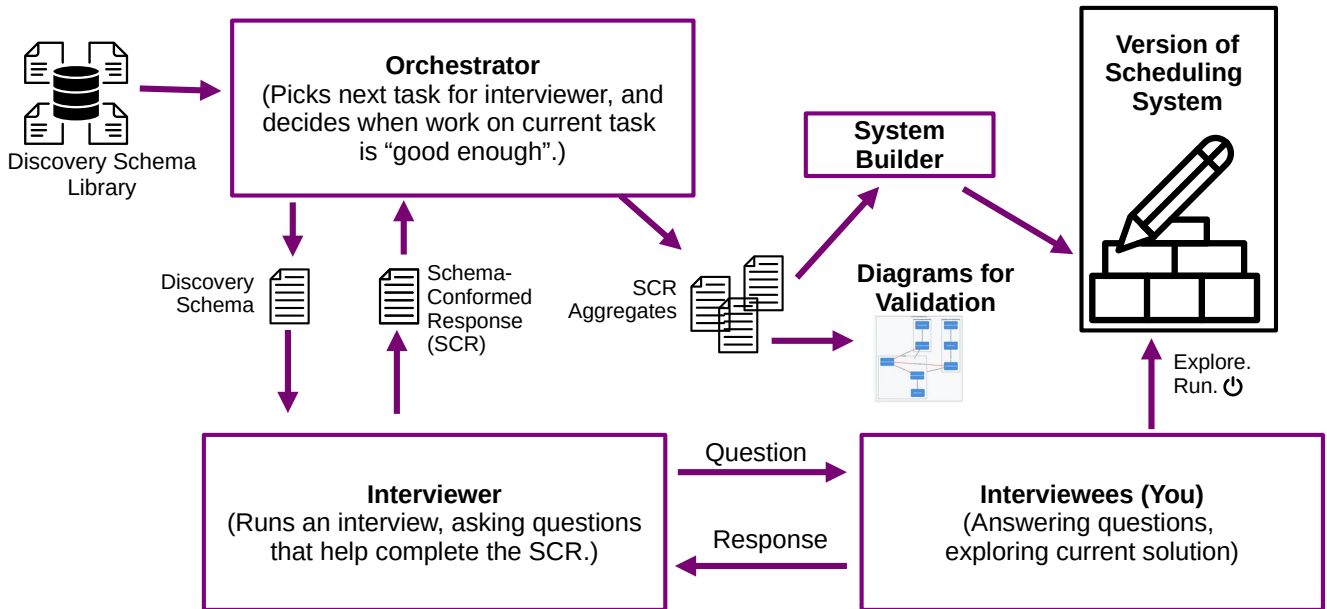
This is the run-time execution of the system. Notice that the TS is not involved, but the designed solution, as expressed in the language of the DSL is, as are the interfaces to the ERP and PHM systems (for data).

Security-wise (thinking industrial system safety) TS being disengaged in run-time is prudent.

I provide this slide just to remind you of the run-time system, which also appears in the next slide...

How does it work? (as you design the scheduling system)

cf. partially-observable Markov decision process



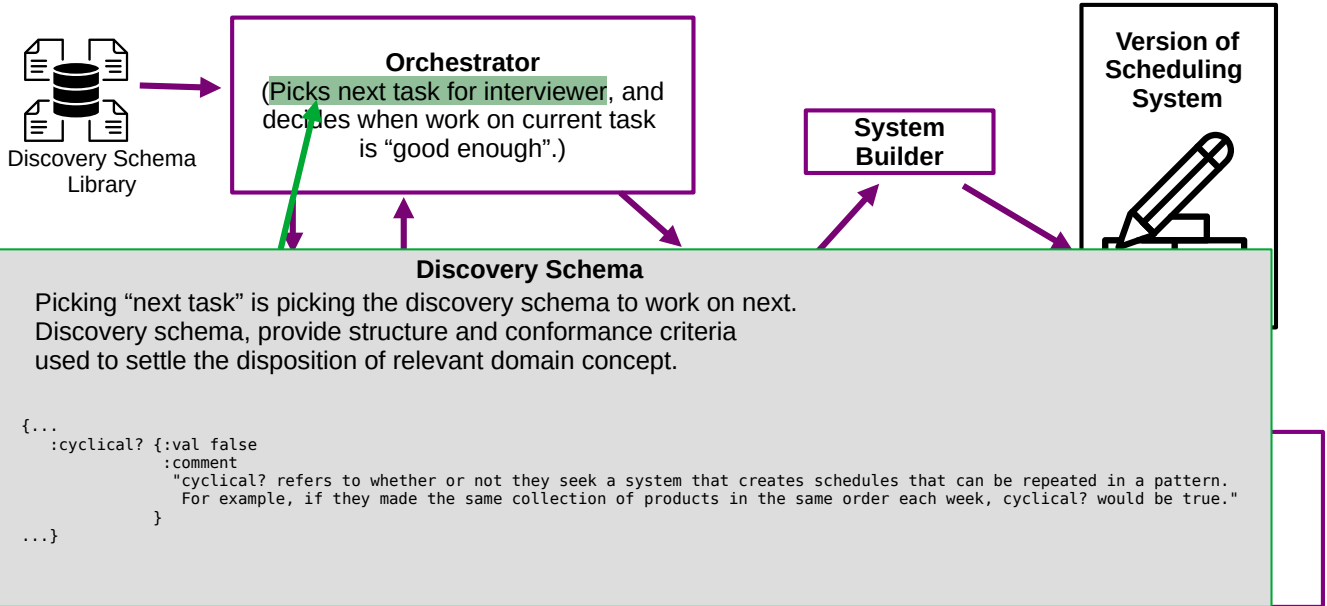
You see here the run-time system (upper RHS)

Walk backwards through interviewee/interviewer...

How does it work? (as you design the scheduling system)

cf. partially-observable Markov decision process

NIST



16

The orchestrator delegates to interviewers by means of discovery schema, which are schema (things providing well-formedness constraints). Here we see a fragment of one that is about determining whether the interviewees scheduling challenges could be answered by cyclic scheduling. The formal definition of cyclical scheduling (which the interviewer has access to) could be a bit **off-putting**.

The interviewer's job is try to get an answer without being off-putting.

How does it work? (as you design the scheduling system)

cf. partially-observable Markov decision process

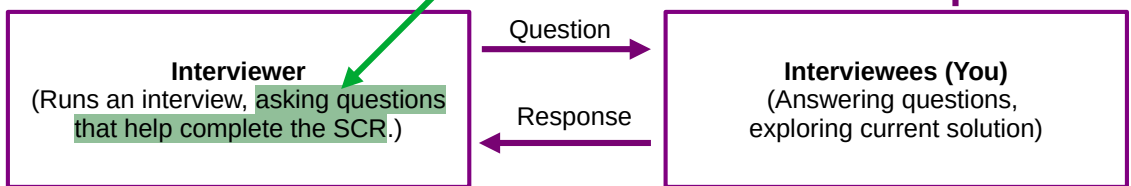
Discovery Schema to Schema Conforming Response (SCR)

```
{...
  :cyclical? {:val false
             :comment
             "cyclical? refers to whether or not they seek a system that creates schedules that can be repeated in a pattern.
             For example, if they made the same collection of products in the same order each week, cyclical? would be true."
             }
...}
```

Interviewer asks question and formulates a *schema conforming response* (SCR)

```
{... :cyclical? false...}      -- or --
```

```
{... :cyclical? {:val false
                :comment "An automotive job shop is unlikely to be able to use cyclical scheduling."}
...}
```



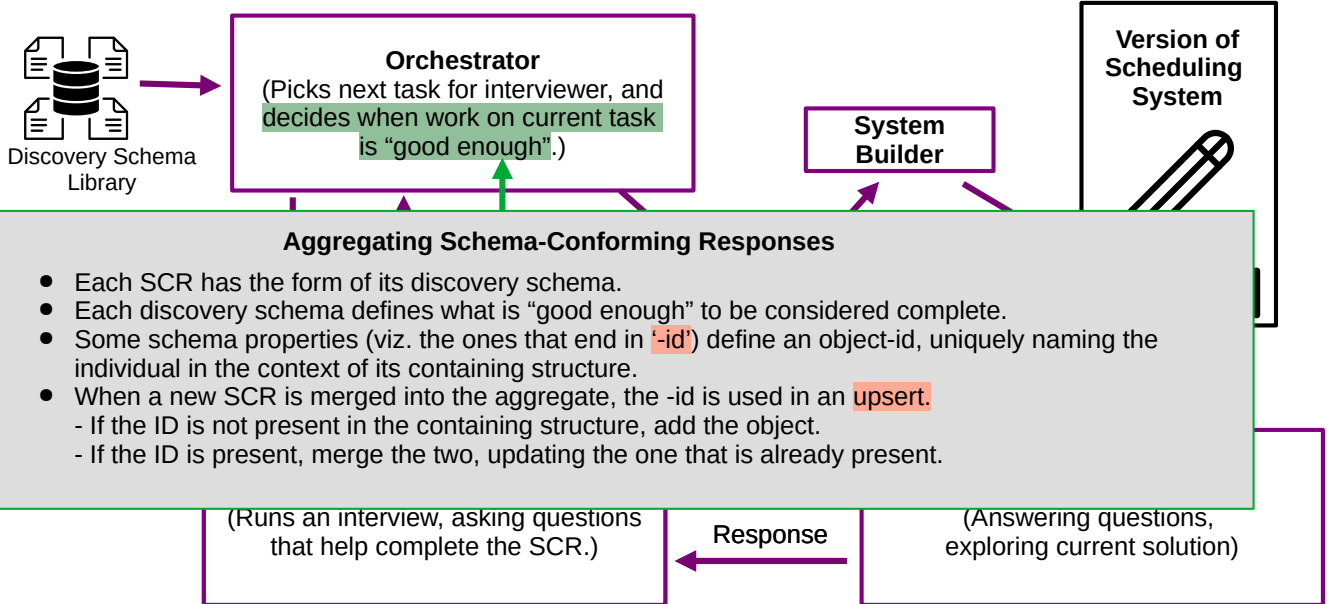
The interviewer has a question asking budget. In some cases, the interviewer might guess from context the value of a SCR facet. Example: Since you are an automotive job shop, you don't do the same thing every week (scheduling period).

Maybe the **non-off-putting** question here would be: "Do you make the same products in the same quantities each scheduling period?" The answer recorded in the SCR is just Cyclical? = true/false, however.

How does it work? (as you design the scheduling system)

cf. partially-observable Markov decision process

NIST



18

Every discovery schema has its own methods for combining SCRs. Likewise determining when complete. In some, the interviewer can decide when it is done, in others the orchestrator sends it a new discovery schema, indicating time to move on to a new area of discussion.

cf. partially-observable Markov decision process

(Picks next task for interviewer, and decides when work on current task is “good enough”).

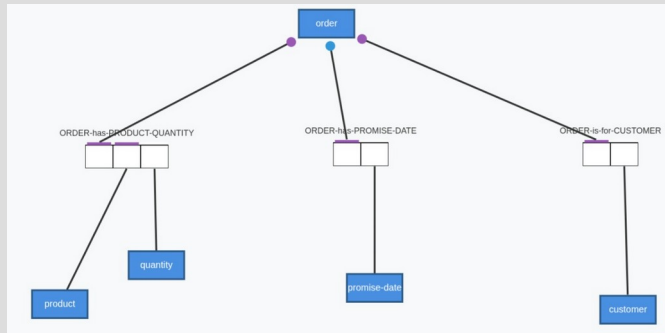
System Builder

Version of Scheduling System

The diagram illustrates the structure of the model, organized into three main sections: **period-wise fitting**, **period-wise production**, and **profit-wise production**.

- period-wise fitting** (top section) contains:
 - period-wise production** (left box):
 - oil annual stock
 - oil reserves to date
 - oil stock to date
 - profit-wise production** (right box):
 - oil reserves to date
 - oil stock to date
 - oil stock to date
- stock-wise production** (bottom section) contains:
 - oil stock to date
 - oil stock to date
 - oil stock to date

Arrows indicate the flow of information between these components, showing how data from the top level feeds into the bottom level.



19

[NB:The tool steward will need to be able to explain the intent of some diagrams (for example ORM such as shown here).]

BTW, we created these two diagram capabilities (ORM and FFBD) in just 2 days using a library we weren't familiar with using an AI programming assistant. [No time in body of talk to describe the relationship between this and working with a tool steward, but maybe in the Q&A.]

cf. partially-observable Markov decision process

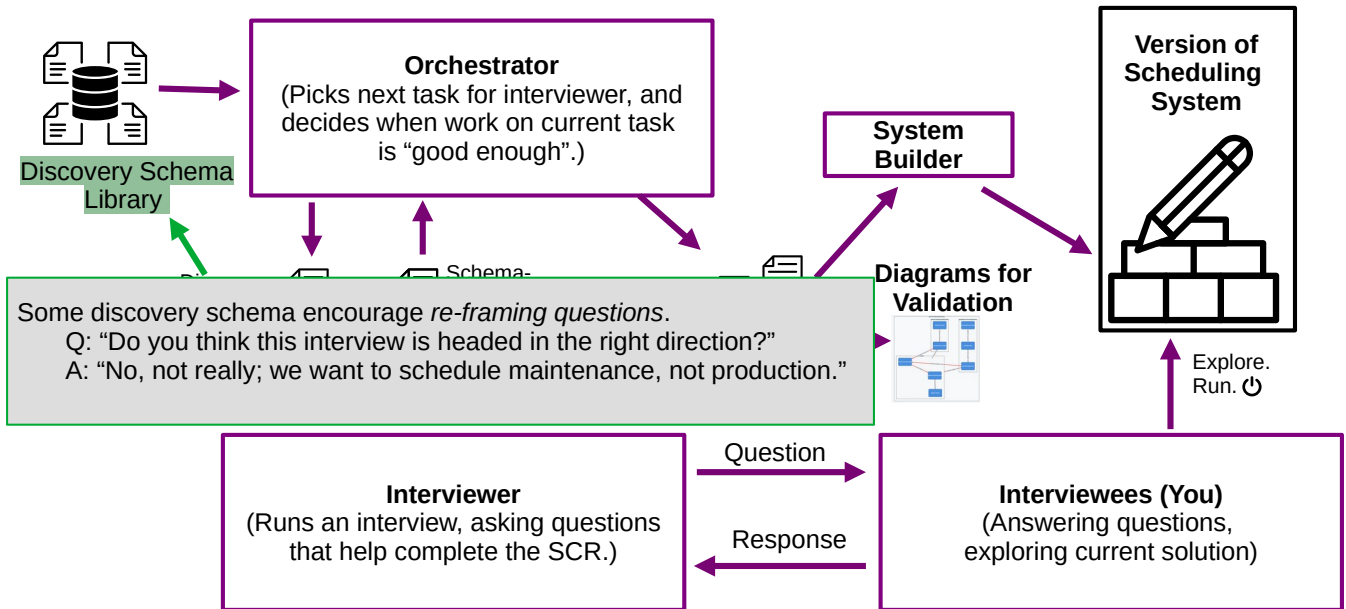
20

- Warm up questions.
- Reframing questions.

How does it work? (as you design the scheduling system)

cf. partially-observable Markov decision process

NIST



21

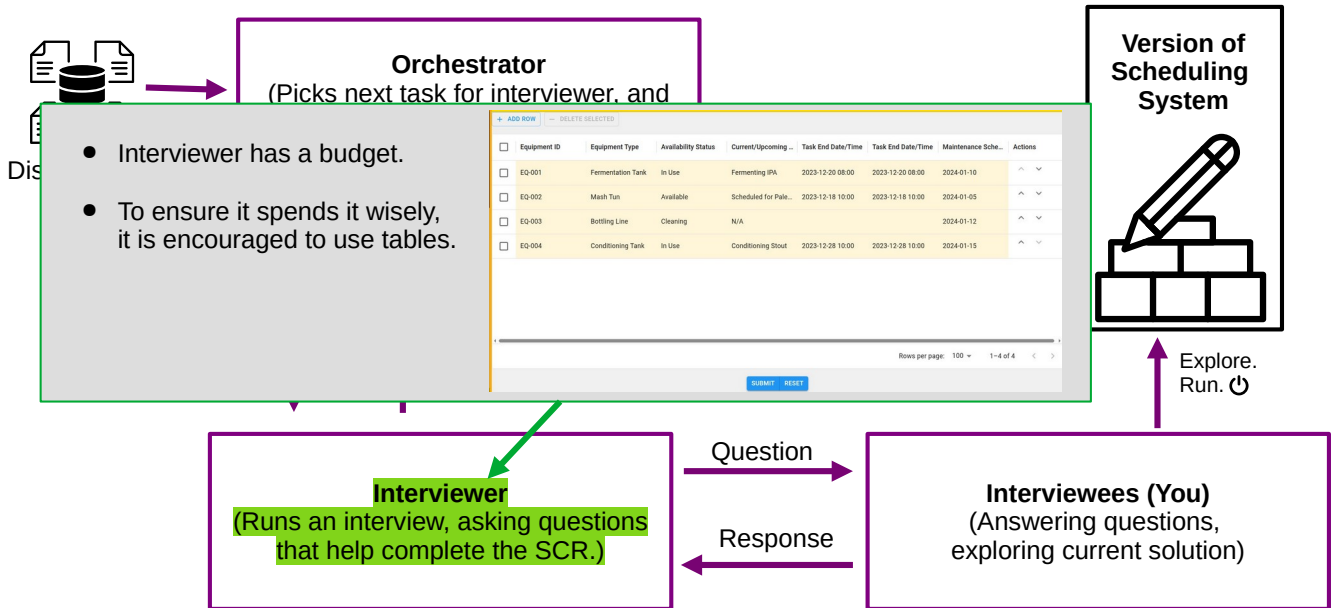
BTW, in much of our testing to date, the interviewees are AI agents charged with being domain experts.

I think it was with "cookie making" that the agent first told us they were interested in scheduling production but were interested in scheduling equipment maintenance. That's fine, most human consultants would also want to know how they make money (see work in management science on dynamic capability) before knowing the pain points. But that answer should affect the choice of downstream discovery schema.

How does it work? (as you design the scheduling system)

cf. partially-observable Markov decision process

NIST



22

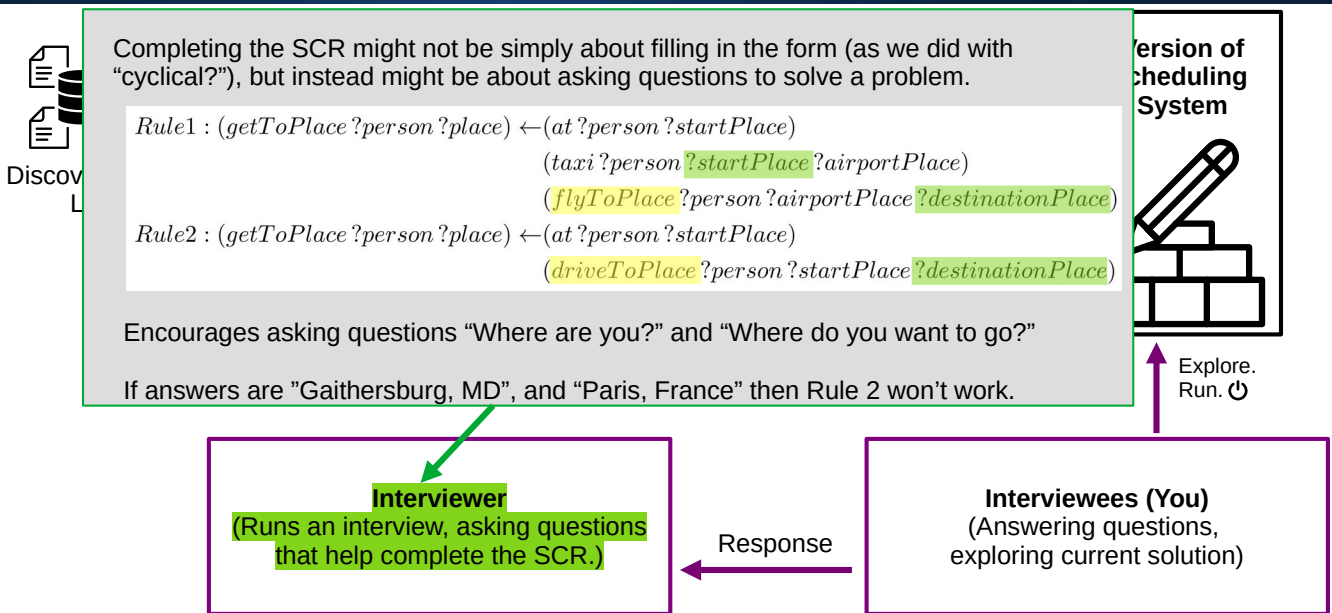
The use of a budget and encouraging use of tables has proven its worth in driving monotony out of interviewing.

We provided hints in the DS where we thought table might be a good idea.

How does it work? (as you design the scheduling system)

cf. partially-observable Markov decision process

NIST



23

In theory, we believe that DS need not be simple forms, they could involve solving problems (“puzzles”?). We are experimenting with this in manufacturing process planning. Give the agent instructions for a simple plan like planning transportation for a trip, and encourage it to resolve the ground facts on the RHS of rules by asking the interviewers the key questions.

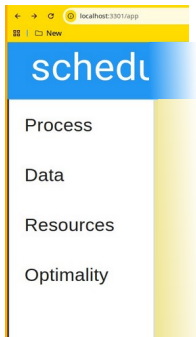
If the DS is a template that describes a simple use (like the Gaithersburg-to-Paris example, but also includes process steps for more complex planning problems (we are using manufacturing of athletic shoes) the interviewer will be seeking to discover specific characteristics (ground facts) of the shoe the interviewees are trying to plan. Some rules just aren’t part of making that shoe, others are.

- Problem: mentoring and interviewing should present a consistent understanding of the domains involved.
 - Interview orchestrator should lead participants in a fruitful path through it.

- Solutions

- Ontology knowledge graph of viewpoint.
- Partitions conversation by topic.

1) FLOW-SHOP-SCHEDULING-PROBLEM: the problem of scheduling jobs where all jobs execute the same task
2) TIMETABLING-PROBLEM: the problem of assigning collections of resources to event time slots
3) PROJECT-SCHEDULING-PROBLEM: the problem of defining start and finish dates to all activities
4) JOB-SHOP-SCHEDULING-PROBLEM: the problem of scheduling jobs where the order in which the jobs are processed is fixed
5) SINGLE-MACHINE-SCHEDULING-PROBLEM: the problem of choosing the sequence by which each of several jobs is processed



There are some aspects we won't have time to discuss. One is partitioning the interview by universe of discourse (bigger units than DS).

Another is ontology / Knowledge graph – we believe this will be important to keep the interviewer's perspective in line with our viewpoint on the BoK. We have DOLCE / RDF ontologies of modeling and scheduling BoKs but have not yet integrated them.

- Intro: human/AI teaming to support use of complex tools
- AI-led interviews to understand system environment and goals
- Summary

- “Wrap” complex tools in an agentic system (tool steward).
 - an idea that has had currency in our division at NIST for many years; just not possible previously.
- Discussed teaming in systems engineering and integration as an instance of more general teaming
 - Perform agent-led interviews to discover the operational environment and system goals.
 - Build-in verification through integrated testing and analysis of steward-created diagrams
 - Will mentoring with the DSL leaves participants feeling empowered to enhance the system when the opportunity arises?

- Technical Report on DSLs in Teaming with ISO/IEC JTC1 SC42
- Exploration of areas other than production scheduling
- Engage with small and medium-sized manufacturers

This the 3rd year I've presented at this workshop series.

Two years ago we knew LLMs were going to be key to interview process but not much more. Since then we've seen very significant improve in the capabilities of LLMs and the infrastructure (such as MCP) BTW, our work has been agentic for two years but we are just getting started with an MCP implementation, which we think is an easy fit.

We are busy with SC42. We'll have a paper on interviewing in JIDPS soon. We are going to be focusing on best practice in using DSLs, and more applications than just scheduling. Listening to SMM for suggestions.

- **BACKUP SLIDES**

Why not just use an AI programming assistant?

- Tool steward embodies and employs best practices in the BoKs and tool use.
- Integrate tools for V&V and testing.
- Capture and curate the entire design decision process in a project database.
- Mentor participants with DSL to avoid a black box solution.
- Tool steward is a product.
 - Collective action on a common pool resource

Agent-wrapped tool is a product. The product can be (the AI assistant is used in a service).