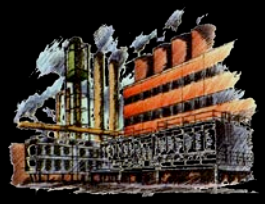
Roman Barták, MFF UK Praha (bartak@ktiml.mff.cuni.cz)

KNOWLEDGE ENGINEERING FOR PLANNING AND SCHEDULING: FROM PROBLEMS TO SOLVERS AND BACK

Two worlds

- planning vs. scheduling
  - planning is about finding activities to achieve given goal
  - scheduling is about allocating known activities to limited resources and time
- generic (AI) vs. specific (OR) approaches
  - flexible techniques but bad worst-case runtime (due to search)
  - guaranteed runtime and schedule quality, but inflexible techniques
- theory vs. practice

What you ~~can~~ will hear in factory



- "We are different…"
  - means, what you know is useless here
- "Outsiders cannot understand it, it takes a lot of time…"
  - means, you have to listen to us or to spend part of your life here
- "Methods that suite others cannot implemented here…"
  - means, your experience and knowledge are impressive, but you have to start from scratch

Theory vs. practice

- Academy
  - the researcher's world consists of resources and their usage
    - "how can I use the resources to get max X and min Y…"
    - "how can I get, using objective metrics, a plan that for the long term, will improve the plant efficiency…"
- Factory planners
  - the planner's world consists of products and their flow
    - "how can I produce this product now, and this one and that one…"
    - "how can I satisfy Mr. X from sales and Mr. Y from the plant and the customer at the same time, without getting into new troubles…"

Our goal

- Bring optimization technology to regular users that are not experts in optimization
- How?
  - familiar user interface
  - advanced and flexible optimization techniques
  - hidden from the user (no complex set-up)
  - full customization of produced results

Our approach

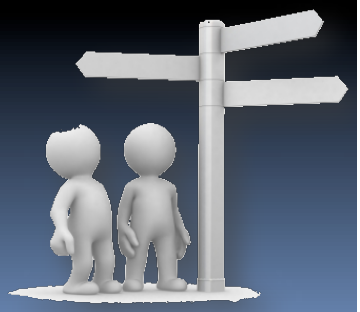
- Be close to the customer
  - use notions that factory planners are familiar with
- Translate the problem to solving formalism
  - use flexible modelling and solving approach
- Solve the problem to acceptable quality
  - combine heuristics and inference

CP

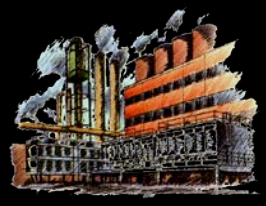
- Allow customers to modify the solution
  - support interactive changes of solutions

Talk outline

- Problem description
  - production scheduling
- Problem formalisation (modelling)
  - workflows, temporal networks with alternatives
- Model verification
  - complexity and algorithm
- Problem solving
  - constraint-based scheduling
- Solution visualization
  - interactive Gantt Viewer
- System demo
  - FlowOpt project

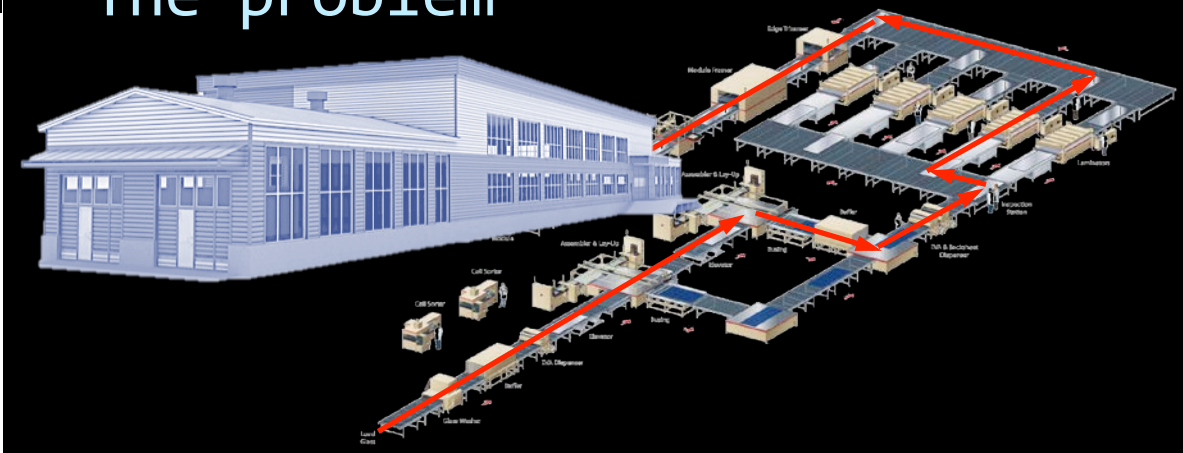


Motivation



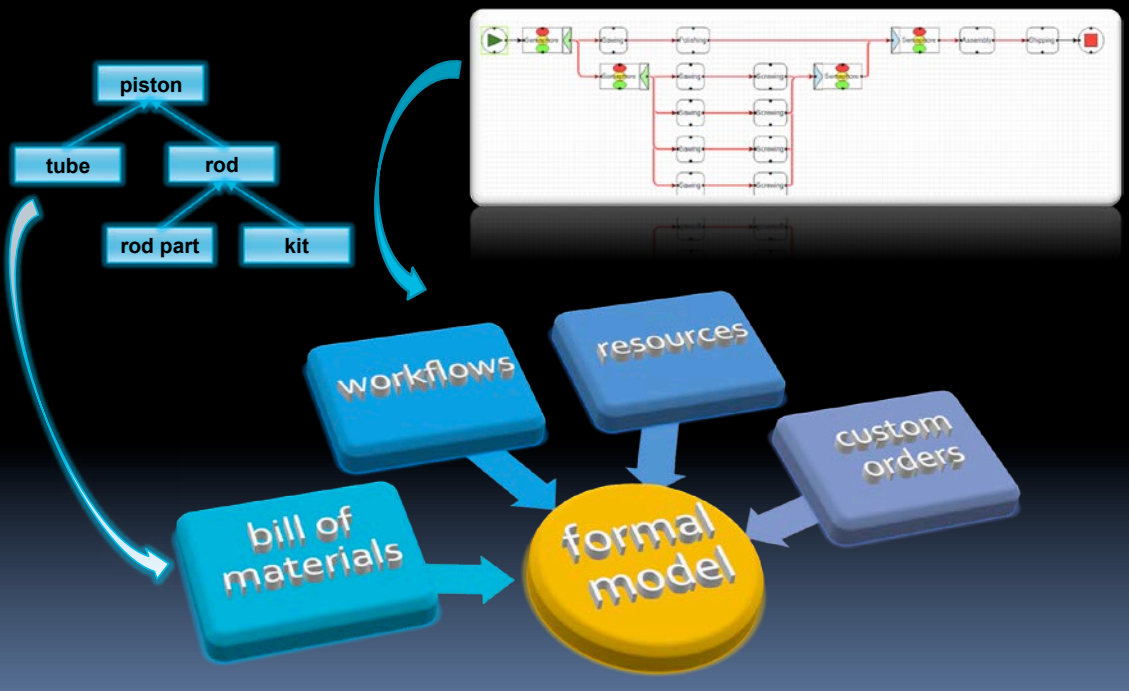
- help small and medium enterprises (SMEs) to optimize their production
- Why?
  - increasing competition from low wage economies
  - existing optimization practice no more feasible
  - existing tools too expensive and to rigid
- SMEs specifics
  - connected to area of origin (no move elsewhere)
  - high variety of products (no mass production)

The problem

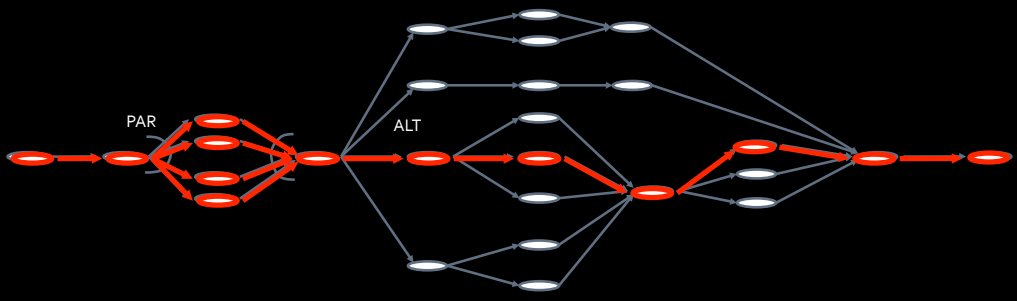


- How to formally describe (manufacturing) processes?
- How to find a production plan/schedule?
- How to present the result to the user?

Problem formulation



Formal model

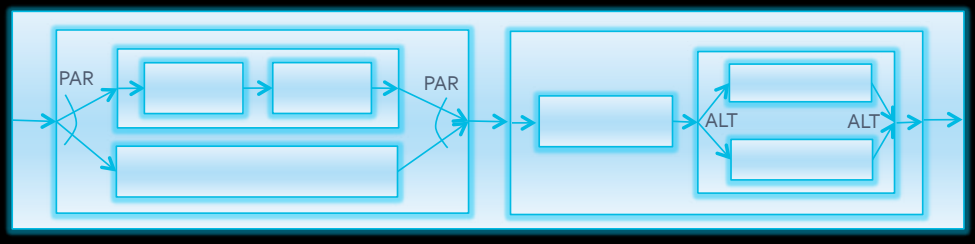


- description of workflow as a **Temporal Network with Altenatives**
- the first task is to select a valid process

The problem of selecting a problem containing given operations is **NP-complete**!

- typical workflows have a **specific structure**

Nested wofkflows



- workflow is obtained by task decomposition

The problem of selecting a valid process containing given operations is tractable.

However, if we add temporal constraints then the problem becomes NP-complete again.

Workflow Verification

- When users „draw" workflows, they can introduce flaws.
  - flaw – some activity cannot be part of any valid process selected from the workflow
  - nested structure is flaw-free (it forces users to use „good" design principles)
  - however extra constraints may introduce flaws (for example a cycle of activities)
- The goal of workflow verification is to detect the flaws.
- We formalize the **verification problem** as the problem whether there exists at least one valid process for each task in the nested workflow with extra constraints.

Causal Constraints

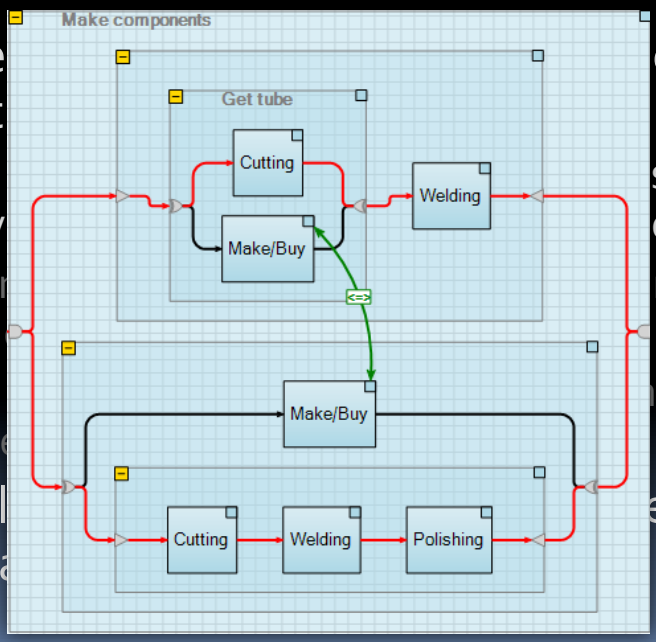
definitions

- define relations restricting appearance of activities in the process
  - A ⇒ B: if activity A appears in the process then activity B must appear in the same process as well
  - A xor B: activities A and B cannot appear in a single process together
  - A ⇔ B: both activities A and B appear in the same process
- Useful for expressing relations between alternatives from different nests.

Causal Constraints

definitions

- define ce of activit
  - A ⇒ ss then activ ess as well
  - A xor r in a singl
  - A ⇔ the same proc
- Useful een alterna

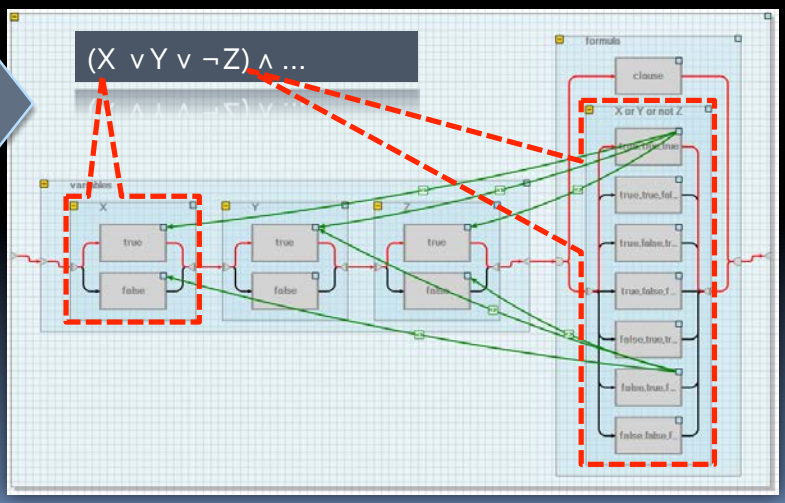


Implication Constraints

verification

- The problem whether there exists or not a valid process in a nested workflow with extra implication constraints is **NP-complete**.

3-SAT



Precedence Constraints

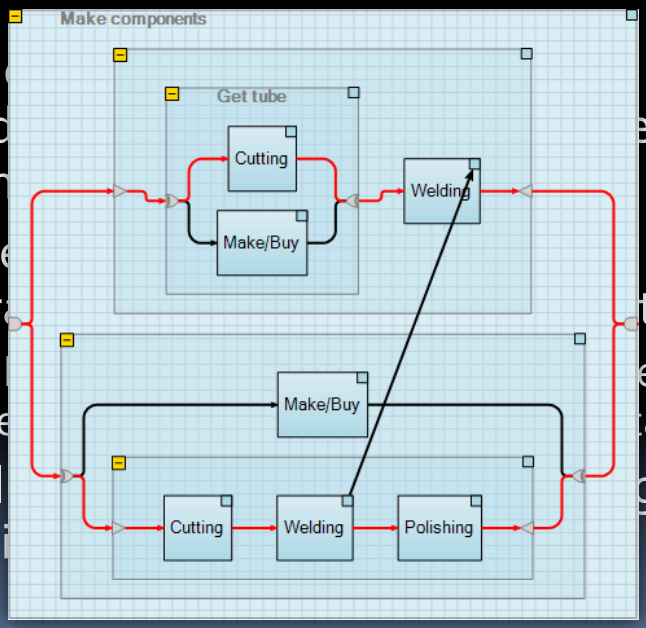
definitions

- workflow structure includes implicit precedence constraints (for example via serial decomposition)
- but users can add extra precedence constraints beyond the nested structure
  - A → B: if activities A and B appear in the same process then activity A ends before B starts
- Useful to express additional ordering of activities.

Precedence Constraints

definitions

- workfl
  - preced
  - decom
- but use
  - constr
  - A →
  - proce
- Useful
  - activiti



e via serial

ture

e same

arts

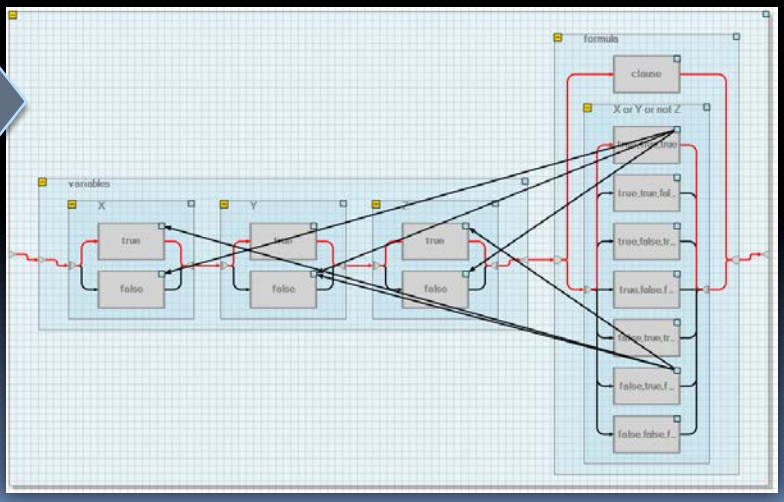
g of

Precedence Constraints

verification

- The problem whether there exists or not a valid process in a nested workflow with extra precedence constraints is **NP-complete**.

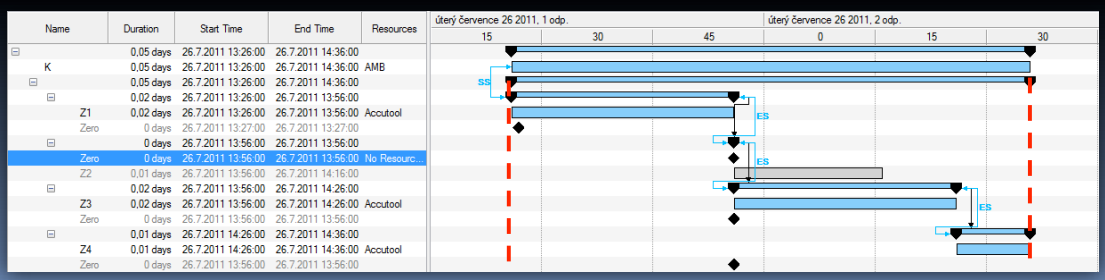
3-SAT



Synchronisation Constraints

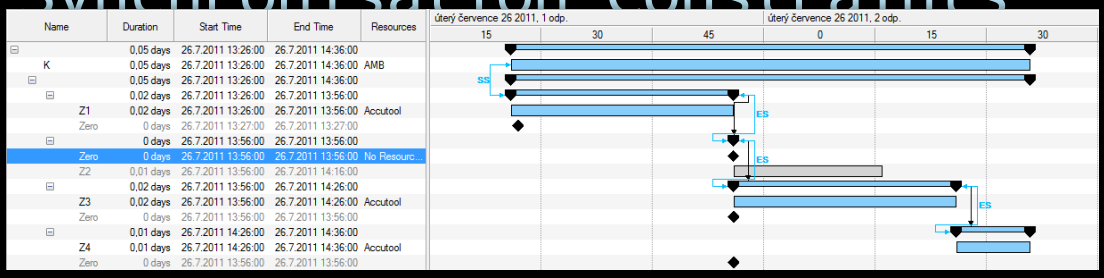
definitions

- Sometimes , users need to express stronger **temporal relations** between the activities:
  - A se B – activity A starts exactly when activity B ends
  - A ss B – activities A and B starts at the same time
  - A ee B – activities A and B end at the same time
- There are already some implicit temporal synchronisations, for example, the parent task starts when the first child task starts.

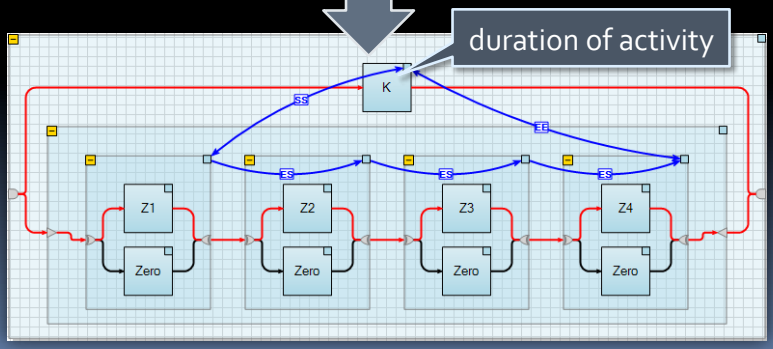


Name	Duration	Start Time	End Time	Resources
	0,05 days	26.7.2011 13:26:00	26.7.2011 14:36:00	
K	0,05 days	26.7.2011 13:26:00	26.7.2011 14:36:00	AMB
	0,05 days	26.7.2011 13:26:00	26.7.2011 14:36:00	
	0,02 days	26.7.2011 13:26:00	26.7.2011 13:56:00	
Z1	0,02 days	26.7.2011 13:26:00	26.7.2011 13:56:00	Accutool
Zero	0 days	26.7.2011 13:27:00	26.7.2011 13:27:00	
	0 days	26.7.2011 13:56:00	26.7.2011 13:56:00	
Zero	0 days	26.7.2011 13:56:00	26.7.2011 13:56:00	No Resourc...
Z2	0,01 days	26.7.2011 13:56:00	26.7.2011 14:16:00	
	0,02 days	26.7.2011 13:56:00	26.7.2011 14:26:00	
Z3	0,02 days	26.7.2011 13:56:00	26.7.2011 14:26:00	Accutool
Zero	0 days	26.7.2011 13:56:00	26.7.2011 13:56:00	
	0,01 days	26.7.2011 14:26:00	26.7.2011 14:36:00	
Z4	0,01 days	26.7.2011 14:26:00	26.7.2011 14:36:00	Accutool
Zero	0 days	26.7.2011 13:56:00	26.7.2011 13:56:00	

Synchronisation Constraints



Subset-sum problem
Given a set of positive integers Z_i and integer K, does the sum of some subset of $\{Z_i \mid i = 1,\ldots,n\}$ equal to K?



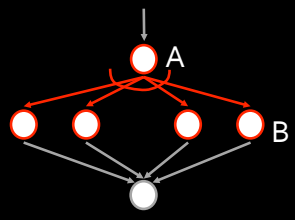
Solving technology

- constraint satisfaction techniques
  - inference + search
- Variables
  - Boolean variables V_X to describe selection of operation
  - temporal variables $\mathbf{T_X}$ to describe time allocation
- Constraints
  - $V_X = \Sigma_{i=1,\ldots,k} V_{Yi}$ (process splitting)
  - $V_X * V_Y * (T_X + a) \leq T_Y \wedge V_X * V_Y * (T_Y - b) \leq T_X$. (temporal distance)
  - $V_X * V_Y * (T_X + Dur_X) \leq T_Y \vee V_X * V_Y * (T_Y + Dur_Y) \leq T_X$ (unary resource)

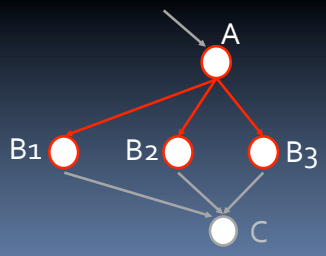
[FLAIRS 2007]

Logical constraints

- The path selection problem can be modeled as a **constraint satisfaction problem**.



- each **node** A is annotated by {0,1} variable V_A
- each arc (A,B) from a **parallel branching** defines the constraint $V_A = V_B$



- let arc (A,B1),…, (A,Bk) be all arcs from some **alternative branching**, then $V_A = \Sigma_{i=1,\ldots,k} V_{Bi}$

[RAC 2008]

Temporal constraints

- So far we assumed that an arc in the graph describes a **precedence**.
- We can annotate each arc (X,Y) by a **simple temporal constraint** [a,b] with the meaning $\mathbf{a \le Y\text{-}X \le b}$.
  - **(Nested) Temporal Network with Alternatives**
- Base constraint model:
  - each **node** A is annotated by a **temporal variable** T_A with a domain $\langle 0, \text{MaxTime} \rangle$, where MaxTime is a constant given by the user.
  - Temporal relation [a,b] between nodes X and Y must hold if both nodes are valid!

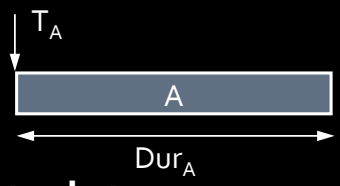
$$\mathbf{V_X * V_Y * (T_X + a) \le T_Y \ \wedge V_X * V_Y * (T_Y - b) \le T_X.}$$

Notes:

- $V_X = 0 \vee V_Y = 0 \rightarrow 0 \le T_Y \ \wedge 0 \le T_X$
- $V_X = V_Y = 1 \rightarrow (T_X + a) \le T_Y \ \wedge (T_Y - b) \le T_X$.
- The above temporal constraint does not assume the type of branching!

Resource constraints

- **standard scheduling model**
  - start time variable: $\mathbf{T_A}$
  - duration variable: $\mathbf{Dur_A}$

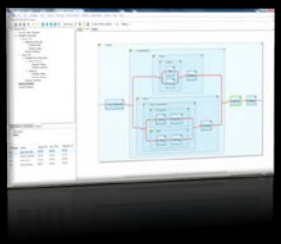


- unary (disjunctive) **resource constraints**
  - two operations allocated to the same resource do not overlap in time

$$V_X * V_Y * (T_X + Dur_X) \leq T_Y \ \vee \ V_X * V_Y * (T_Y + Dur_Y) \leq T_X$$

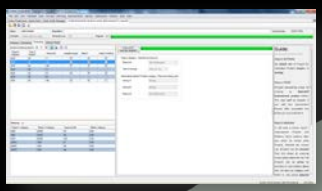
  - or, we can use **existing global constraints** modeling unary resource (edge-finding, not-first/not-last, etc. inference techniques) extended to optional operations
    - (in)valid operations: $Val_A = 1 \Leftrightarrow Dur_A > 0$

FlowOpt system



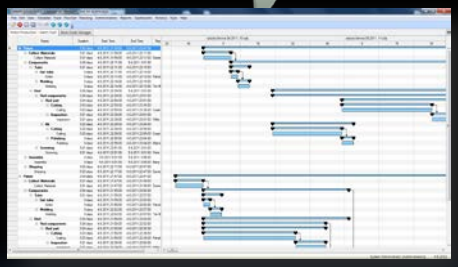
- **FlowOpt** tools build on top of enterprise optimisation system MAK€ for SMEs
  - build-to-order (engineer-to-order) production
  - on-time-in-full objective (earliness/tardiness)
- The system supports the following tasks:
  - interactive graphical design of workflows
  - creating and scheduling custom orders
  - visualisation and modification of schedules
  - schedule analysis

FlowOpt design process

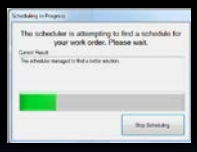


Gantt Viewer

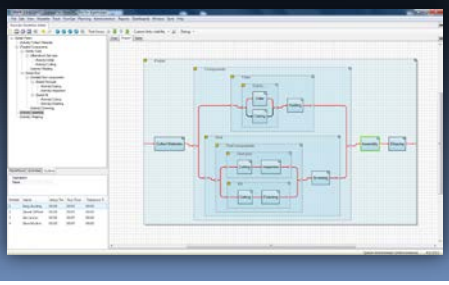
Analyzer



Optimizer

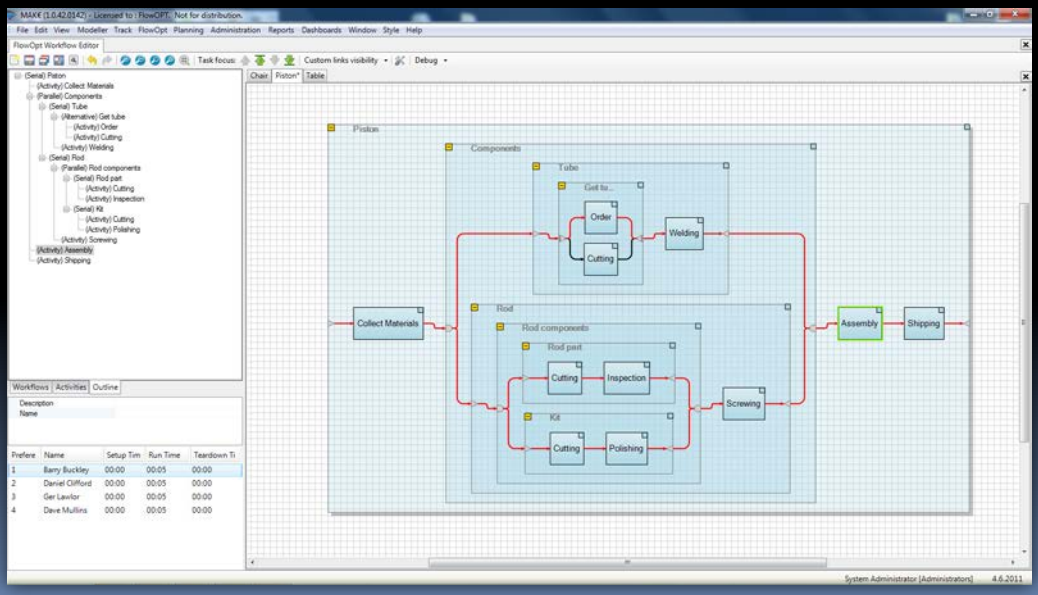


Workflow editor



Workflow editor

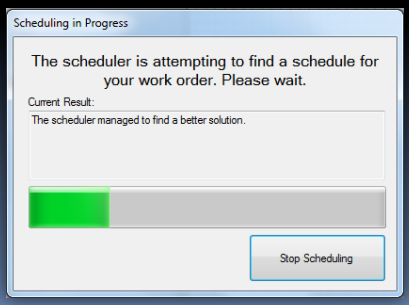
- top-down and bottom up approach to design **nested workflows**
- supports **extra logical** (mutual exclusion,…) and **temporal** (synchronization,…) **constraints**



Optimiser

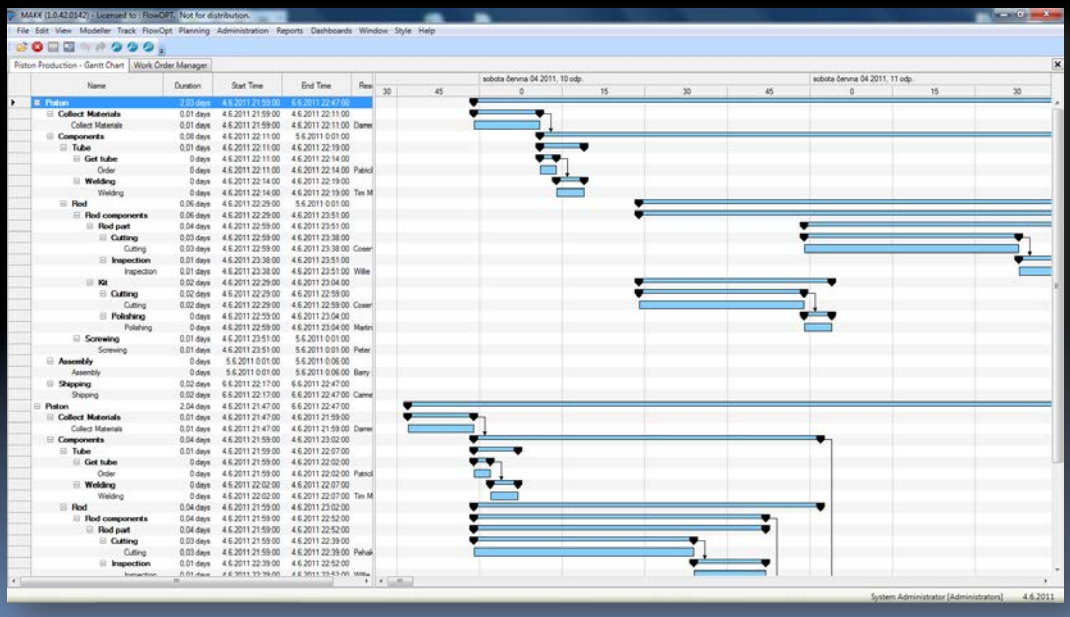
a fully **automated scheduler** that takes description of workflows for ordered products and generates a schedule

- implemented in ILOG CP Optimiser (OPL Studio)
- branch-and-bound optimisation (earliness and lateness costs and cost for alternatives are assumed)



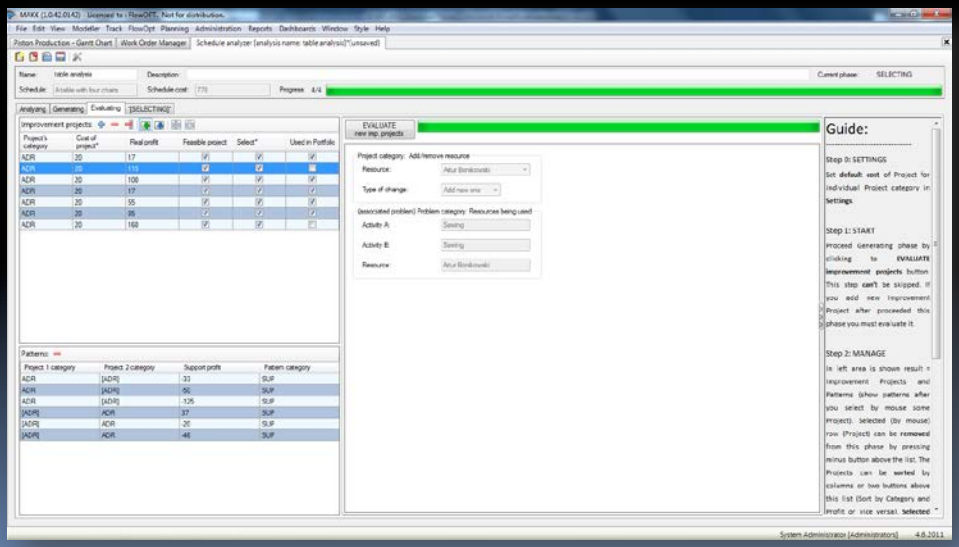
Gantt Viever

- visualization and modification of schedules



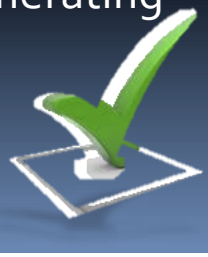
Analyser

- analysis of problems in schedules (late deliveries) and suggestions for enterprise improvements (buying a new resource)



Summary

- a novel formal model of workflows with a nested structure and alternatives
- automated verification of workflows
- theoretical complexity of the problems
- automated generation of the scheduling model
- generic optimization techniques for generating production plans/schedules
- vizualization and schedule repair



Future?

- automated design of workflows
- extraction of scheduling rules
- dynamic re-scheduling
- improved analysis of schedules
- stronger integration of analysis and optimization
- ....

KNOWLEDGE ENGINEERING FOR PLANNING AND SCHEDULING: FROM PROBLEMS TO SOLVERS AND BACK

Charles University in Prague, Czech Republic

Roman Barták (bartak@ktiml.mff.cuni.cz)
Ondřej Čepek
Pavel Surynek
Martin Hejna
Milan Jaška
Ladislav Novák
Vladimír Rovenský,
Tomáš Skalický

Entellexi, Ireland

Martin Cully
Con Sheahan
Dang Thanh-Tung

supported by the Czech Science Foundation (KnowSched, P202/10/1188) and by EU FP7-SME-2007-1 (ValuePOLE, contract 222218).