CroFlow
Enabling Highly Automated **Cr**oss-**O**rganisational Work**Flow** Planning

Violet Ka I Pun

I-MDE-A workshop

16 May 2023

Western Norway University of Applied Sciences

With funding from The Research Council of Norway

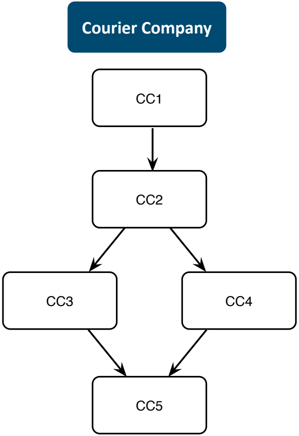
About myself

- Associate professor at Western Norway University of Applied Sciences
- Background:
  - Program languages
  - Concurrency
  - Formal modelling & Semantics
  - Types
  - Static program analysis
  - Cost analysis
  - Multicore architecture
  - . . .

Our society is gradually moving towards a digitalised future

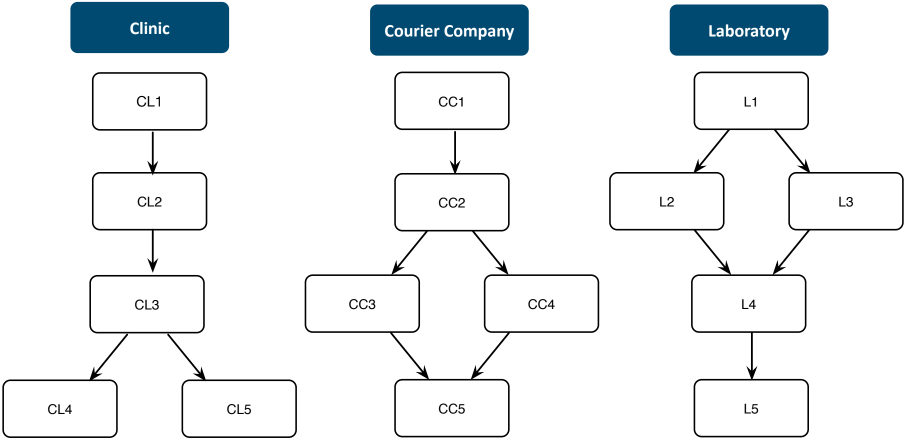
What are workflows?

Single Workflow



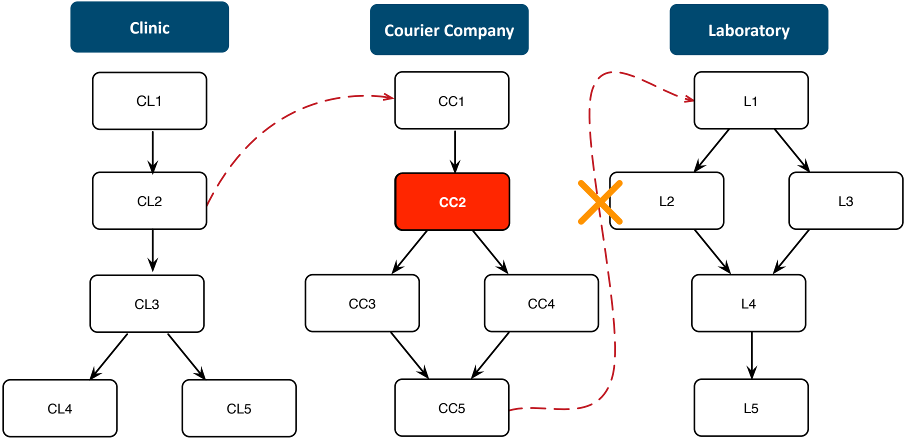
What are cross-organisational workflows

Multiple **connected** workflows



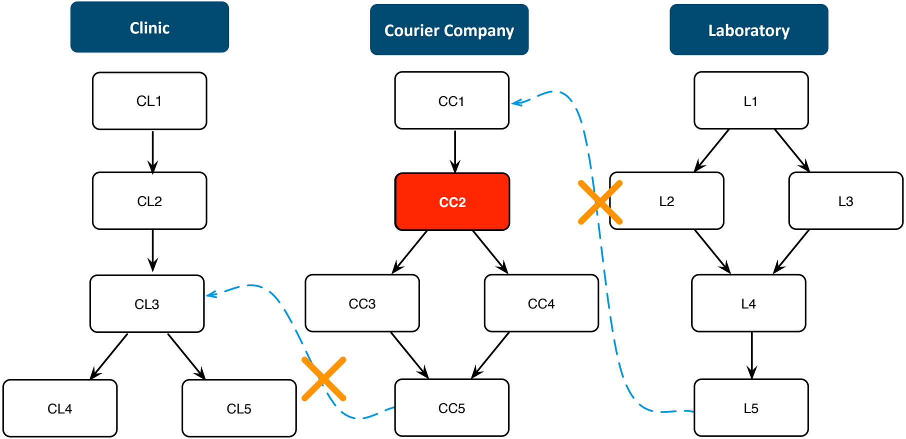
What are cross-organisational workflows

Multiple **connected** workflows



What are cross-organisational workflows

Multiple **connected** workflows



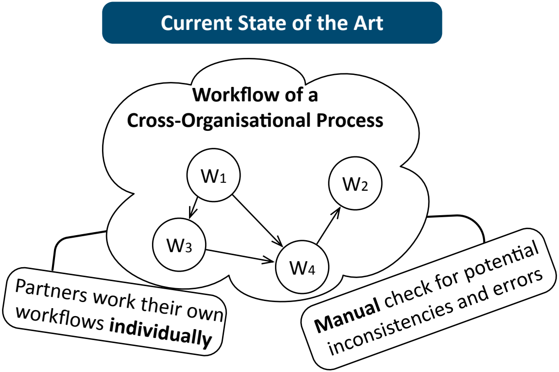
Not so good experience, perhaps because of not so ideal workflows

Fikk sjokk på Gardermoen

Camilla Bernal og sønnen Axel fikk sjokk da de måtte vente i åtte timer på Gardermoen, uten tilgang til mat og drikke.

https://www.dagbladet.no/nyheter/fikk-sjokk-pa-gardermoen/74126855

How workflows are usually planned?



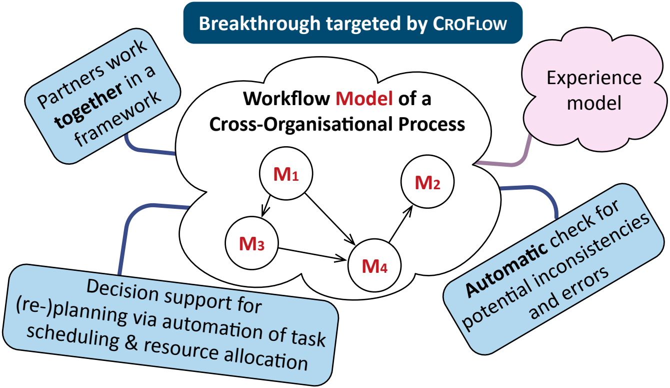
Changes can be **inconsistent**:

- Partners make changes to their own workflow (Wi).
- Changes are immediately reflected in the whole workflow, and effects of the changes are unclear.
- Manual check & changes are required for inconsistencies & errors

The goal

A new methodology to **automate** cross-organisational workflow planning and revisioning, which will eventually contribute to a **successful digitalisation**.

What we want to achieve?



Changes are **consistent**:

- Partners makes changes to their own workflow model (Mi) in the framework
- Changes are then automatically propagated to the whole workflow
- The revised model is simulated & verified according to task dependencies
- Changes are implemented after the model is modified

How do we achieve that?

Three phases

Phase 1: formal workflow modelling

Phase 2: automated task and resource scheduling, and bottleneck discovery

Phase 3: automated plan revisioning

based on **formal abstract executable models**, and integrates **concurrency theory** and **programming language theory**

Some preliminary results

- Core modelling language with the notion of
  - Resources
  - Task dependency

- Cost analysis, specifically WCET, for workflow models

The language

$$
\begin{aligned}
P &::= R\ \overline{CD}\ \{\overline{T\ x\ ;}\ s\} \\
R &::= [r \mapsto (b, \mathrm{Q})] \\
CD &::= \textbf{class}\ C\ \{\overline{T\ x\ ;}\ \overline{M}\} \\
M &::= Sg\ \{\overline{T\ x\ ;}\ s\} \\
Sg &::= T\ m(\overline{T\ x})\ DP \\
DP &::= \overline{CM} \mid DP \vee DP \\
CM &::= C.m \mid CM \wedge CM \\
T &::= B \mid \textbf{Fut}\langle B \rangle \\
B &::= C \mid \text{Bool} \mid \text{Int} \mid \cdots
\end{aligned}
$$

$$
\begin{aligned}
e &::= x \mid b \mid fs \mid \textbf{this} \\
fs &::= f? \mid fs \wedge fs \\
rs &::= \emptyset \mid \{\mathrm{Q}\} \cup rs \\
rid &::= \emptyset \mid \{r\} \cup rid \\
s &::= x = rhs \mid \textbf{skip} \mid \textbf{if}\ e\ \textbf{then}\ s\ \textbf{else}\ s \\
&\quad \mid \textbf{await}\ f? \mid \textbf{add}(rs) \mid \textbf{hold}(rs) \\
&\quad \mid \textbf{release}(rid) \mid \textbf{return}\ e \mid s\ ;\ s \\
rhs &::= e \mid \textbf{new}\ C \mid f.\textbf{get} \\
&\quad \mid e.m(\overline{e})\ \textbf{after}\ \overline{fs} \mid e!m(\overline{e})\ \textbf{after}\ \overline{fs}
\end{aligned}
$$

Our models

```
[r1 ↦(true,{Driver,TruckDriver,5}),r2 ↦(true,{Driver,VanDriver,5}),r3↦(true,{Van,Delivery,1500}),. . .]
```

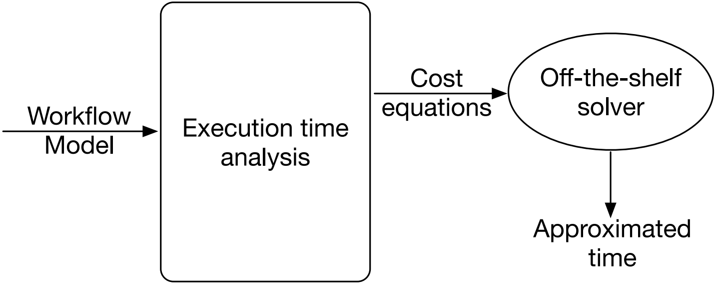
```
class Retailer{
  Unit sale(Retailer rt, Supplier sp, Courier cr){
    Fut<Unit> f1; Fut<Unit> f2; Fut<Unit> f3; Warehouse wh;
    wh = new Warehouse;
    if(In—Warehouse){
      f1 = !pack(wh) after dl 5;
      wait(f1);} // wait for packing
    else{
      f2 = !supply(sp,cr) after dl 15;
      wait(f2);} // wait for supply
    f3 = !deliver(cr) after dl 10;
    wait(f3);} // wait for delivery confirmation
}
```

```
class Warehouse{
  Unit pack(Warehouse wh){
    cost(k1);} // Packing time
}
```

```
class Supplier{
  Unit supply(Supplier sp, Courier cr){
    Fut<Unit> f4;
    cost(k2); // Packing time
    f4 = !deliver(cr) after dl 10;
    wait(f4);}
}
```

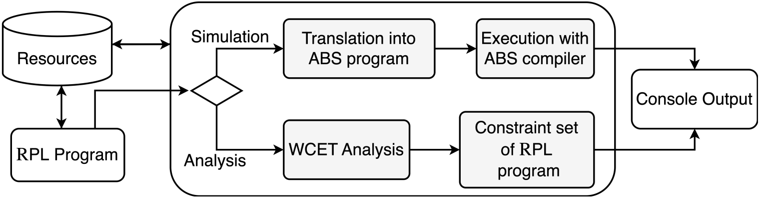
```
class Courier{
  Unit deliver(Courier cr){
    Rid r;
    r = hold({Driver,VanDriver,5},{Van,Delivery,1500});
    cost(k3); // Delivery Time
    release(r);}
}
```

```
{
  Retailer rt; Supplier sp; Courier cr;
  rt = new Retailer; sp = new Supplier; cr = new Courier;
  sale(rt,sp,cr) after dl null;
}
```



$$
\mathcal{T}_{S_m}(I, \Psi, o, t_a, t, s) =
\begin{cases}
1. \quad \mathcal{T}_{S_m}(I', \Psi', o, t'_a, t', s'') & \text{if } s \text{ is } s'; s'', \text{ and } \\
 & \quad (I', \Psi', t'_a, t') = \mathcal{T}_{S_m}(I, \Psi, o, t_a, t, s') \\
2. \quad (I, \Psi + e, t_a, t + e) & \text{if } s \text{ is } \textbf{cost}(e) \\
3. \quad (I, \Psi + c_{m'}, mt_a, mt + c_{m'}) & \text{if } s \text{ is } x = m'(o', \overline{e}) \textbf{ after } \overline{fs} \textbf{ dl } e', \text{ and} \\
 & \mathcal{D} = \{(I', \Psi', t'_a, t') = \textbf{trans}_{S_m}(I, \Psi, o, t_a, t, \mathtt{getF}(fs)) \mid fs \in \overline{fs}\}, \\
 & \text{and,} \\
 & mt_a = max(t_a(\mathcal{D})), \text{ and} \\
 & mt = max(t(\mathcal{D})) \\
4. \quad (I[f \mapsto S_m(o)], \Psi, mt_a + c_{m'}, mt) & \\
 & \text{if } s \text{ is } f = !m'(o', \overline{e}) \textbf{ after } \overline{fs} \textbf{ dl } e', \text{ and } o' \in S_m(o), \text{ and} \\
 & \mathcal{D} = \{(I', \Psi', t'_a, t') = \textbf{trans}_{S_m}(I, \Psi, o, t_a, t, \mathtt{getF}(fs)) \mid fs \in \overline{fs}\}, \\
 & \text{and,} \\
 & mt_a = max(t_a(\mathcal{D})), \text{ and} \\
 & mt = max(t(\mathcal{D})) \\
5. \quad (I[f \mapsto S_m(o')], \Psi[S_m(o') \mapsto \mathcal{E} \cdot \langle c_{m'}, 0 \rangle], mt_a, mt) &
\end{cases}
$$

Prototype of a tool



Type system

- Type system ensuring task dependency, and no deadlock in the workflow models.

$$\frac{\begin{array}{c}\textsc{(T-Sync-Call)}\\ \mathit{conform}(\overline{fs}, DP)\\ \forall f \in \overline{fs}.\ \Gamma \vdash f : \mathbf{Fut}\langle B\rangle\\ \Gamma \vdash e : C \quad \Gamma \vdash \overline{e} : \overline{T} \quad \Gamma(C)(m) = \overline{T} \rightarrow T :: DP\end{array}}{\Gamma \vdash e.m(\overline{e})\ \mathbf{after}\ \overline{fs} : T}$$

Proposition - Deadlock freedom

A cycle in the transitive closure of the dependency graph of a well-typed program implies that no method call to any member of a cycle (SCC) can be made in this program (i.e. especially not from the main entry point).

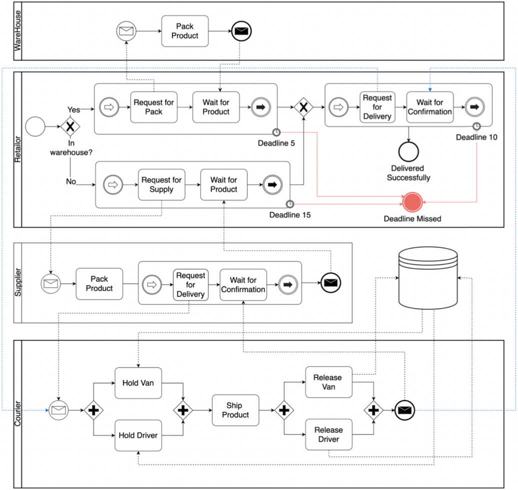
Next steps

Phase 1: formal workflow modelling

Phase 2: automated task and resource scheduling, and bottleneck discovery

Phase 3: automated plan revisioning

First phase


WareHouse
Pack Product
Retailor
Yes
In warehouse?
No
Request for Pack
Wait for Product
Deadline 5
Request for Delivery
Wait for Confirmation
Deadline 10
Delivered Successfully
Request for Supply
Wait for Product
Deadline 15
Deadline Missed
Supplier
Pack Product
Request for Delivery
Wait for Confirmation
Courier
Hold Van
Hold Driver
Ship Product
Release Van
Release Driver

Other phases

Phase 1: formal workflow modelling

Phase 2: automated task and resource scheduling, and bottleneck discovery

Phase 3: automated plan revisioning

Questions?

https://croflow.github.io/