Las Vegas, NV
June 30 - July 3, 2014

Extending eFFBD formalism to task model

Daniel Prun
Ecole National de l'Aviation Civile
Interactive Computing Lab

HSI

- Human System Integration (HSI)
  - "*a set of processes for integrating human related considerations within and across all system elements*" (DOD – Defense Acquisition Handbook 2010)
  - **Objective**: to bring human centered disciplines and concerns into SE processes to achieve an optimal system performance.
- Human Factor (HF) engineering
  - Is one principal consideration of HSI
  - Consists to understand human capabilities and to take them into account during the entire system life-cycle.
  - A recommended method: task analysis

SE and HF

- System Engineering and Human Factors remain 2 separated disciplines
  - Historic: They do not have a common shared origin
  - Focus:
    - SE is centered on functionalities and physical components
    - HF is centered on usability and humans
  - Organization: They are performed by different teams
  - Techniques: They use different formalisms (task model, eFFBD, SYSML …)

- But they share some similar objectives:
  - Identify / analyze / classify tasks & functions
  - Decide best allocation on humans and automated components
  - At the higher level, build an optimized system

Consequences

- Short term:
  - Risk of inconsistencies
  - Extra cost due to work duplication, redundancies
  - Extra cost due to inter team coordination, formalism transformation…

- Long term:
  - User needs not fully understood and elicited
  - Global system not optimized between human and automated parts
  - System engineering considered as an interdisciplinary approach is hard to be put in practice

=> Failure of HSI ?

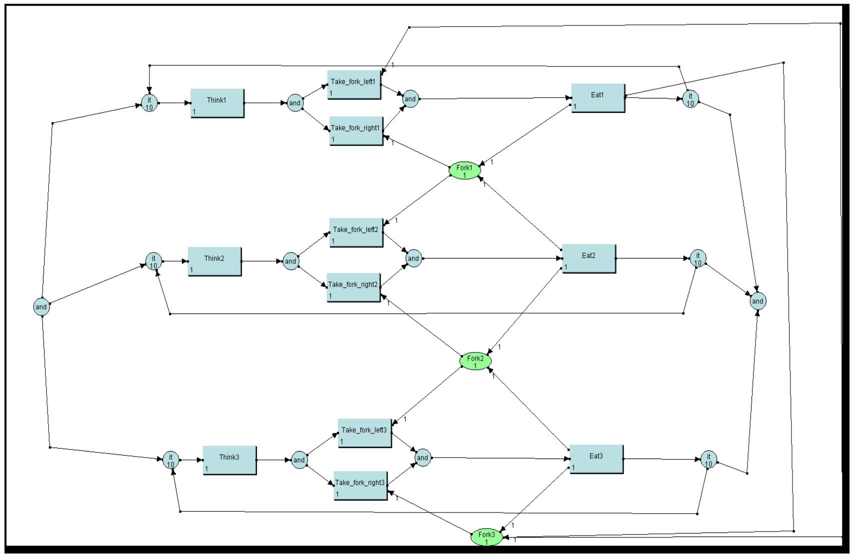
Our objective (general)

- Objective?
  - to reduce the gap between SE and HF

- How?
  - We think that one of the main restriction lies in the different formalisms used
  - So we tried to mix some best used formalisms
  - By extending eFFBD to task models

Do you know eFFBD?

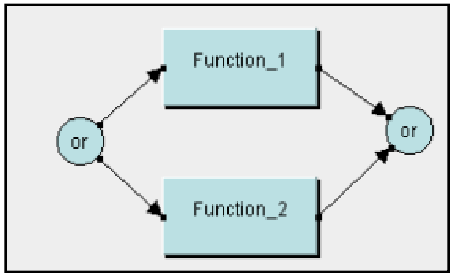
- eFFBD:
  - extended Functional Flow Block Diagram
- History:
  - 50’s: FFBD introduced by TRW
  - 60’s: widely used by NASA
  - 90’s: eFFBD (Long 95)
  - 08: formal semantic (Seidner 08)
- Still widely used for:
  - Supporting functional analysis
  - Sharing information
  - Teaching



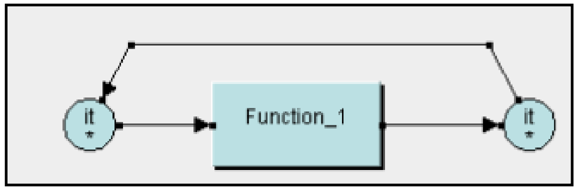
eFFBD

- Basic element: function
  - "*task, action, or activity performed to achieve a desired outcome*" (EIA 1999)

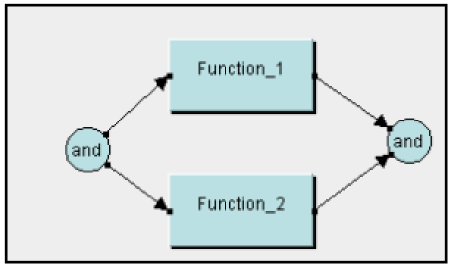
- Basic operators:



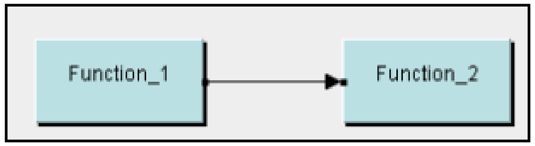
Alternative



Loop



Parallelism



Sequence

Do you know task model?

Las Vegas, NV
June 30 - July 3, 2014

- Basic element: task
  - "*activities required to achieve a goal*" (ISO 9141 Ergonomics of human-system interaction, Part 210)
  - "*what a user is required to do, in term of actions and/or cognitive processes to achieve an objective*"

- Basic operators:

T_1 [] T_2
Choice

T_1 |=| T_2
Order Independency

T_1 ||| T_2
Concurrency

T_1 [> T_2
Interruption

T_1 |> T_2
Suspend/resume

$T1^*$
Iteration

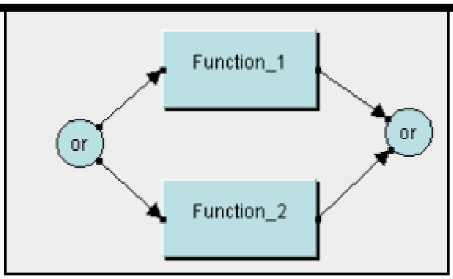
T_1 >> T_2
Enabling

[T_1]
Optional

Comparison

$T_1 \, [] \, T_2$

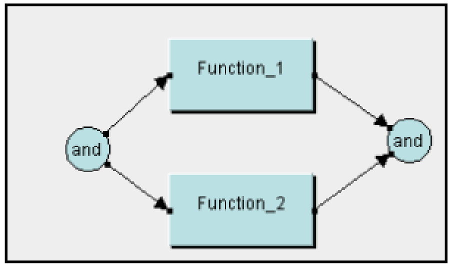
Choice



Alternative

$T_1 \, ||| \, T_2$

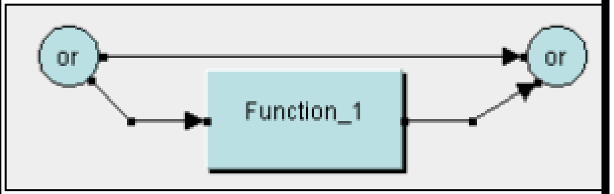
Concurrency



Parallelism

$[T_1]$

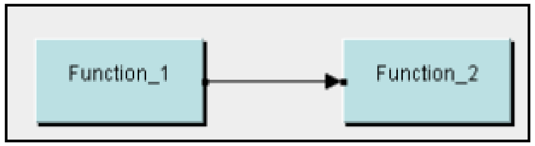
Optional



Alternative

$T_1 >> T_2$

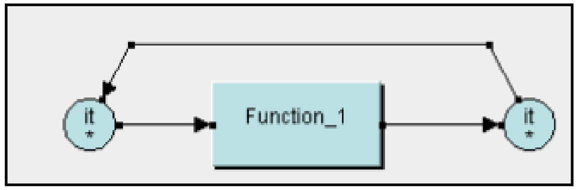
Enabling



Sequence

$T1^*$

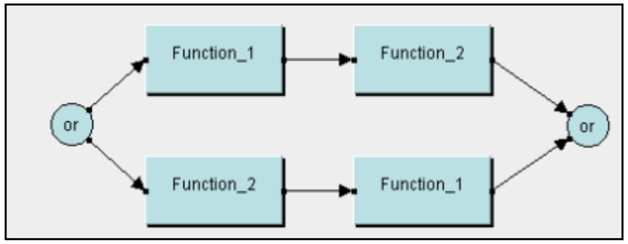
Iteration



Loop

$T_1 \, |=| \, T_2$

Order Independency



Alternative+sequence

Comparison

INCOSE International Symposium
Las Vegas, NV
June 30 - July 3, 2014

- Result :
  - 3 control operators are missing in eFFBD:

$\boxed{T_1 \,[\!> T_2}$

Interruption

$\boxed{T_1 \,|\!> T_2}$

Suspend/resume

- Remark: “kill” ≈ “Interruption”
  - eFFBD has a “kill” operator: “*when the execution of the <u>last function</u> of a kill branch finishes then all other functions performed in parallel are stopped*.”
  - Semantic of task model “Interruption” operator is broader than “kill” operator: parallel functions can be stopped <u>at any time</u> (not necessarily when the last function finishes).

Our objective (refined)

- Introduction of 3 new operators to eFFBD:
  - Disable (for interruption)
  - Suspend
  - Resume

- eFFBD+Disable+Suspend+Resume= xFFBD

- Our definition must be done according to 3 axis:
  - **Graphic**: define what these operators look like
  - **Syntax**: define how to use them in an eFFBD
  - **Semantic**: define what do they mean

Starting point

- eFFBD formal semantic defined in Seidner 08

- An eFFBD is defined by a tupple :

$$\varepsilon = (N, I, A, count, n_0, I_0, \alpha, \beta, K)$$

Nodes:
- Function
- And
- Or
- It
- …

Items

Arcs

Max number of iteration

Initial node

Initial item amount

Functions duration

Kill arcs

Starting point

- eFFBD semantic:

$$\|\varepsilon\| = (S, s_0, N, \rightarrow)$$

State:
- Current activity for each node
- Number of current iterations
- Number of items
- Elapsed time for each function

Initial state

Transition rules from a state to another:

- Inactive
- Enabled
- Executing
- Executed

$$(A,C,I,v) \xrightarrow{n} (A',C',I',v') \Leftrightarrow \begin{cases} A(n) = enabled \wedge PreCondition \\ or\ \left(n \in FC \wedge A(n) = executing \wedge P'_F\right) \\ A'(n) = NextActivity \end{cases}$$

$$\begin{cases} n \notin IT_{out} \cup LP_{out} \end{cases}$$

Ne

$$\begin{cases} n \in AND_{in} \cup OR_{out} \cup LP_{in} \Rightarrow P_{gen} \\ n \in AND_{out} \Rightarrow P_{AND-OUT} \\ n \in OR_{in} \Rightarrow P_{OR-IN} \end{cases}$$

Pre

$$\begin{cases} \forall n' \in N^n, A'(n') = \begin{cases} enabled \text{ if } n' \in Post(n) \\ A(n') \text{ otherwise} \end{cases} \\ C' = C \\ I' = I + Prod(n) \\ \alpha(n) \leq v(n) \leq \beta(n) \end{cases}$$

$$P_F = \begin{cases} \forall n' \in N^n, A'(n') = A(n') \\ C' = C \\ I \geq Cons(n) \wedge I' = I - Cons(n) \\ \forall f \in FC \begin{cases} v'(f) = 0 \text{ if } f = n \\ v'(f) = v(f) \text{ otherwise} \end{cases} \end{cases}$$

Disable operator

- Syntax:
  - an arc can tagged as disable using the keyword “**dis**”.
- Construct rule:
  - a disable arc must belong to a AND structure
- Informal semantic:
  - when the target node of the “disable” arc starts its execution, then all functions belonging to the others branches of the AND structure are interrupted

Disable operator

- xFFBD:

$$\varepsilon' = (N, I, A, count, n_0, I_0, \alpha, \beta, K, Dis)$$

Disable arcs

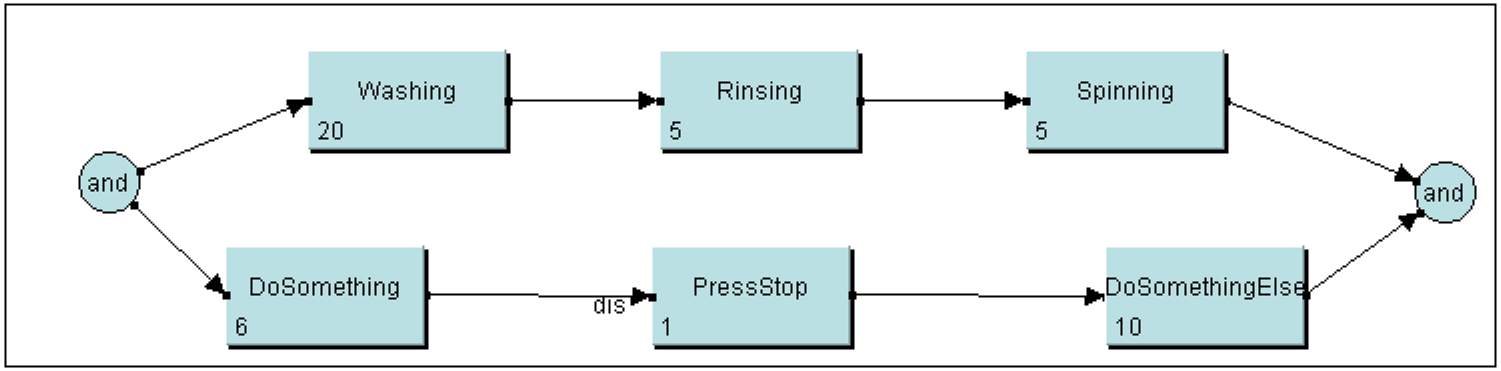
$$\|\varepsilon'\| = (S, s_0, N, \longrightarrow)$$

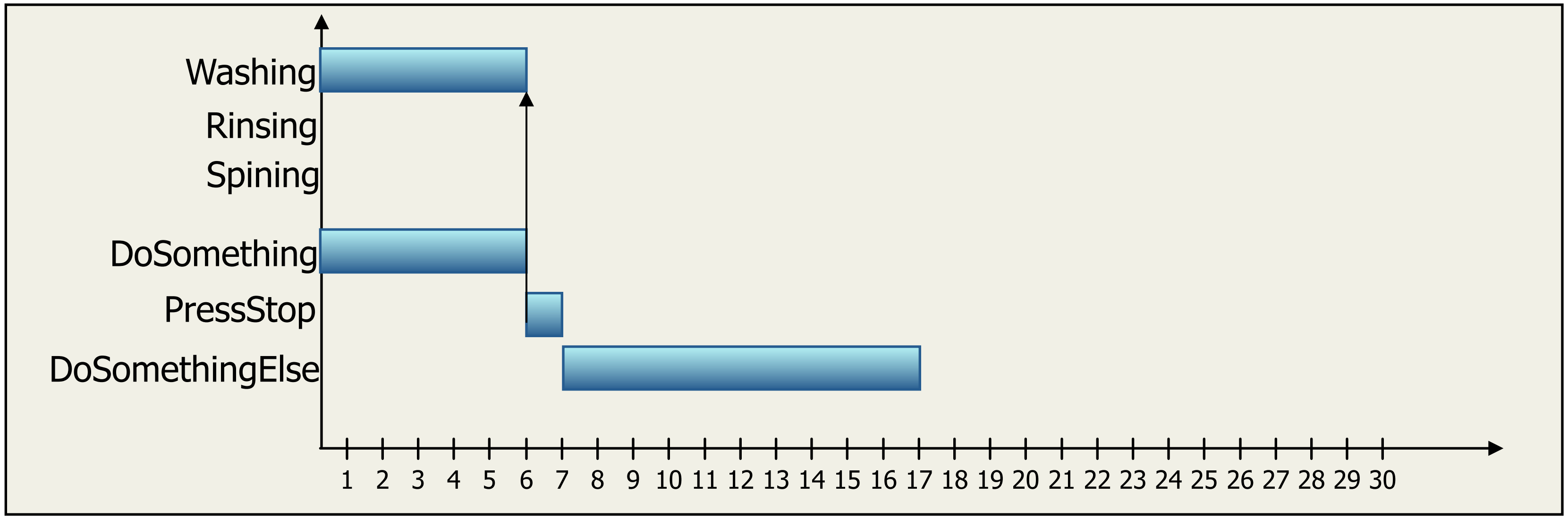
New transition from a state to another

$$P_{F\_Dis} = \begin{cases} \forall n' \in N^n, A'(n') = \begin{cases} inactive \text{ if } n' \in C(and_{in}, and_{out}) \setminus (C(and_{in}, n) \cup C(n, and_{out}) \setminus Pre(and_{out})) \\ executed \text{ if } n' \in Pre(and_{out}) \setminus (C(and_{in}, n) \cup C(n, and_{out})) \\ A(n') \text{ otherwise} \end{cases} \\ C' = C \\ I \geq Cons(n) \wedge I' = I - Cons(n) \\ \forall f \in FC \begin{cases} v'(f) = 0 \text{ if } f = n \\ v'(f) = v(f) \text{ otherwise} \end{cases} \end{cases}$$

The next formula describes the behavior of a node preceded by a disable arc: if n is a currently enabled function node, preceded by a disable arc, then inputs are consumed and elapsed time is set to 0 and nodes of other branches are set to inactive (or to executed is it is a predecessor of a AND_{out} node)

Disable operator





Suspend / Resume operators

- Syntax:
  - an arc can tagged as suspend (resp. resume) using the keywords “**sus**” (resp. “**res**”).

- Construct rules:
  - suspend and resume arcs must belong to a AND structure
  - for each suspend (resp. resume) arc, there is exactly one resume (resp. suspend) arc belonging to the same AND branch.
  - Suspend arc must precede its corresponding Resume arc.

Suspend / Resume operators

- Informal semantic:
  - when the target node of the “suspend” arc starts its execution, then all functions belonging to the others branches of the and structure are suspended.
  - a suspended node has neither time progression, nor resource consumption and production.
  - when the origin node of the “resume” arc finishes its execution, then all functions belonging to the others branches of the and structure are resumed to the state and time progression reach at suspension time.

Suspend / Resume operators

- xFFBD:

$$\varepsilon' = (N, I, A, count, n_0, I_0, \alpha, \beta, K, Res, Sus)$$

Resume arcs

Suspend arcs

$$\|\varepsilon'\| = (S, s_0, N, \rightarrow)$$

State:
- Current activity for each node
- **Memorized activity for each node**
- Number of current iterations
- Number of items
- Elapsed time for each function

- Inactive
- Enabled
- Executing
- Executed
- **Suspended**

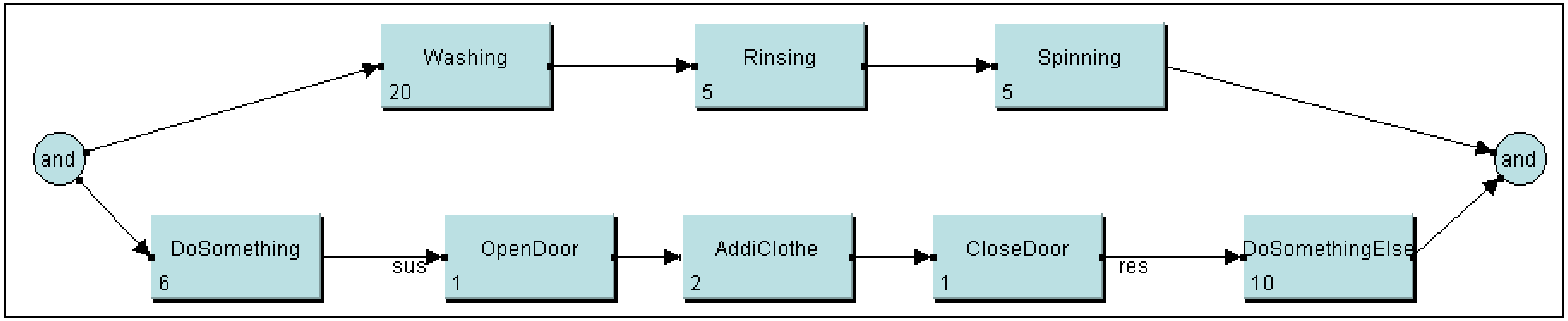
New transition from a state to another

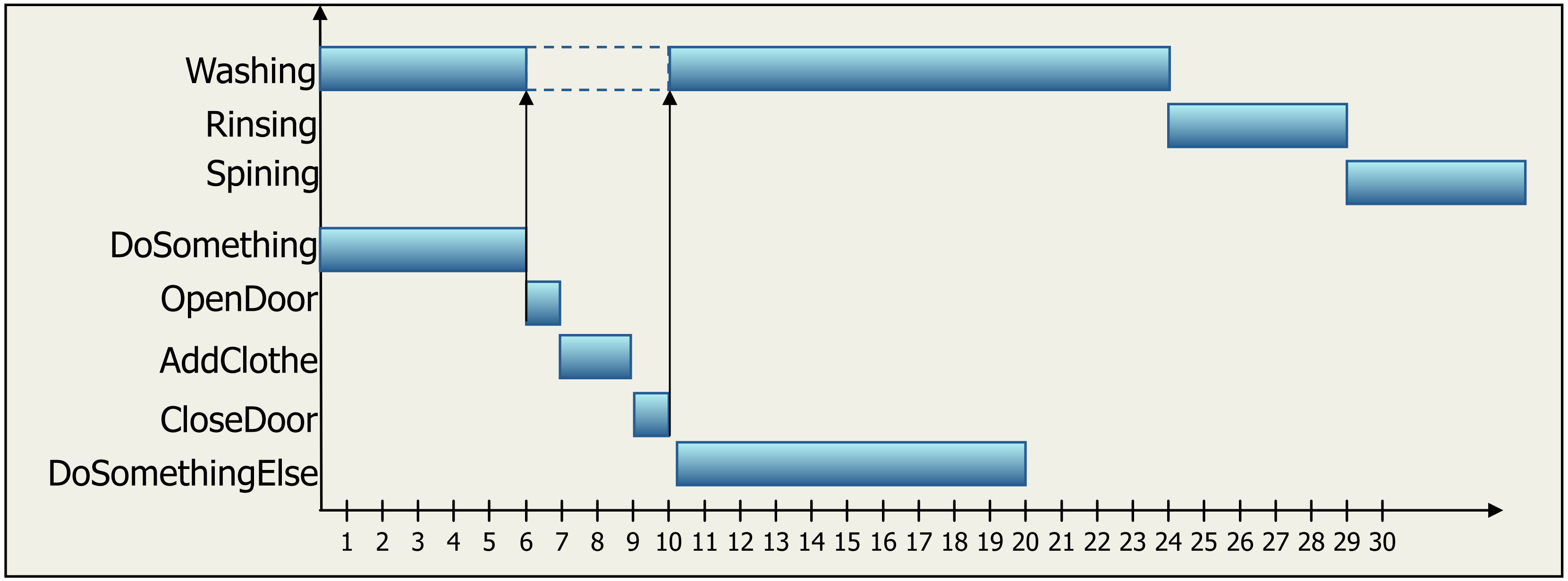
$$P'_{F\_\mathrm{Res}} = \begin{cases} \forall n' \in N^n, A'(n') = \begin{cases} M(n') \; if \; n' \in C(and_{in}, and_{out}) \setminus (C(and_{in}, n) \cup C(n, and_{out})) \\ enabled \text{ if } n' \in Post(n) \\ A(n') \text{ otherwise} \end{cases} \\ C' = C \\ I' = I + Prod(n) \\ \alpha(n) \le v(n) \le \beta(n) \end{cases}$$

$$P_F$$

$$\begin{cases} I \ge Cons(n) \wedge I' = I - Cons(n) \\ \forall f \in FC \begin{cases} v'(f) = 0 \text{ if } f = n \\ v'(f) = v(f) \text{ otherwise} \end{cases} \end{cases}$$

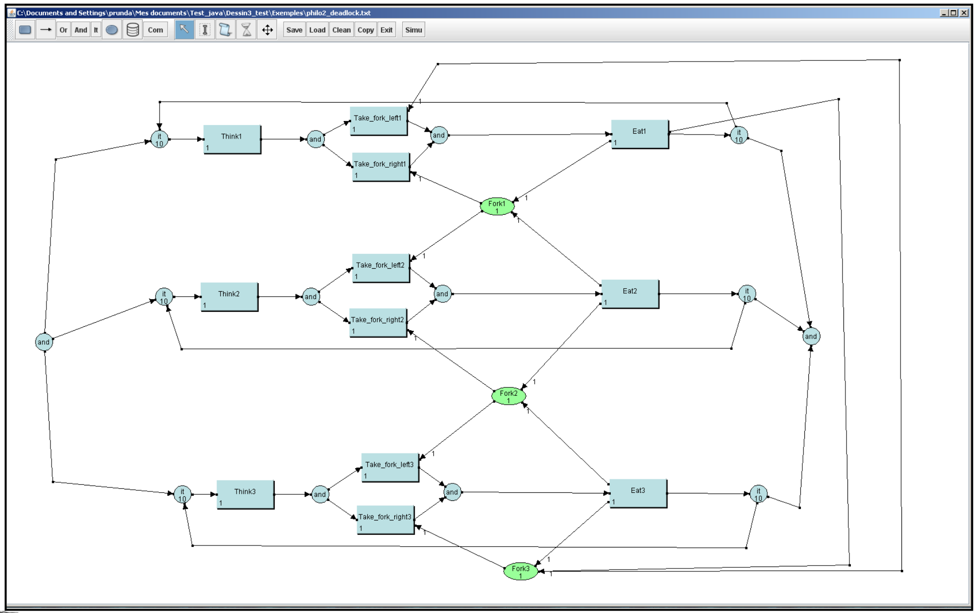
Suspend / Resume operators





Implementation

- LEXE (Light Experimental Xffbd Editor)
  - A tool to build, visualize and simulate xFFBD
  - A demonstrator: implement the syntax and semantic described above
  - An academic tool: open, developed in Java/Swing, can be easily modified.



Conclusion

- We have a good starting framework: graphic, syntax and semantic

- Target:
  - We have used it on small systems: improvement of interaction between SE and HF looks promising
  - To be used on big size systems

- A first step in a bigger project:
  - In INCOSE IS 2012, we defined xFFBD as an ambitious project aiming at developing eFFBD to provide to system designers an enriched functional language

    (*xFFBD: towards a formal yet functional modeling language for system designers*)
  - A lot of other improvements remain to be done…