The Petri_hyb knowledge base for Dynamic Reliability modeling

Marc BOUISSOU
EDF-R&D
Ecole Centrale Paris

26 November 2010

Outline

- An introduction to Dynamic Reliability, PDMP
- A simple Dynamic Reliability problem
- Building a model with the KB3 workbench tools
- Solving it with two Monte-Carlo methods
  - Time discretization
  - Space discretization
- Conclusions

What is « dynamic reliability » ?

- Models and calculation methods taking into account the bi-directional interaction between
  - discrete events causing sudden state changes

    and
  - continuous physical processes

State vector of the system = $(X, I)_t$, where :
X = vector of continuous variables
I = index of discrete state

Examples

- Actions of I on X
  - Pumps and valves act on pressure, levels, temperatures…
- Actions of X on I
  - Temperature, pressure act on failure rates
  - When continuous variables hit thresholds, they can trigger the startup/shutdown of components

The theoretical model in dynamic reliability: PDMP

- Standard model (with continuous trajectories for continuous variables)

$$\frac{dX}{dt} = g(X, I)$$

$$\Pr(I(t+\Delta t) = j / I(t) = i)) = a(i, j, X(t)) + o(\Delta t)$$

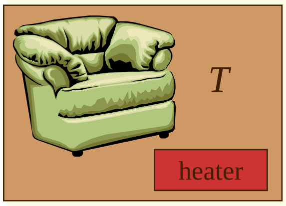
« Piecewise deterministic Markov process »
(Davis 1984)

The time *t* itself is often included in *X:*
Allows to model non exponential distributions

- Extended model (discontinuities are allowed for « continuous » variables when I changes)

Marc Bouissou (EDF-R&D / ECP)

A heating system



External temperature

T_E

Heater:
- on at Tmin, off at Tmax
- subject to failures and repairs

$$\frac{dT}{dt} = heater\_on(t).Power.K1 - (T(t) - T_E).K2$$

$$EX : \frac{dT}{dt} = heater\_on(t) \times 5 - (T(t) - 13) \times 0.1$$

(time in hours, temperatures in Celsius degrees)

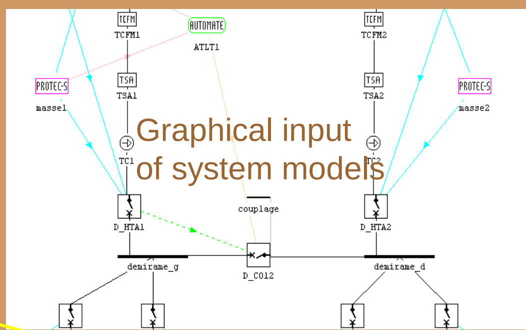
Marc Bouissou (EDF-R&D / ECP)

Solving this problem with the KB3 workbench

Principles of the KB3 workbench

Knowledge Base (in Figaro language)

Generic description of components



Figaro 0 (**textual** model)

Fault-tree

BDMP

Standard fault-tree processors:
Aralia, Risk-Spectrum
- ✓Minimal cutsets
- ✓Reliability, Availability

Sequences Generator: FIGSEQ
- ✓Most probable sequences
- ✓Reliability, MTTF
- ✓Asymptotic availability

Monte-Carlo simulator: YAMS
- ✓Most probable sequences
- ✓Reliability, availability
- ✓Mean values of numeric variables…

The FIGARO modeling language

- Developed in 1990
- Validated by hundreds of complex system studies
- Characteristics
  - Object-oriented (multiple inheritance, objects = instances of generic classes)
  - Dynamic behaviour described by rules -> easily understandable
  - Two levels: Order 1 and Order 0
- Numerous available knowledge bases
  - «Abstract kb» (Markov, Petri, Reliability block diagrams, BDMP)
  - Kb describing physical components
    - The most complex: Topase = 27,000 lines of Order 1 FIGARO

Marc Bouissou (EDF-R&D / ECP)

A simple KB for dynamic reliability: « hybrid » Petri nets

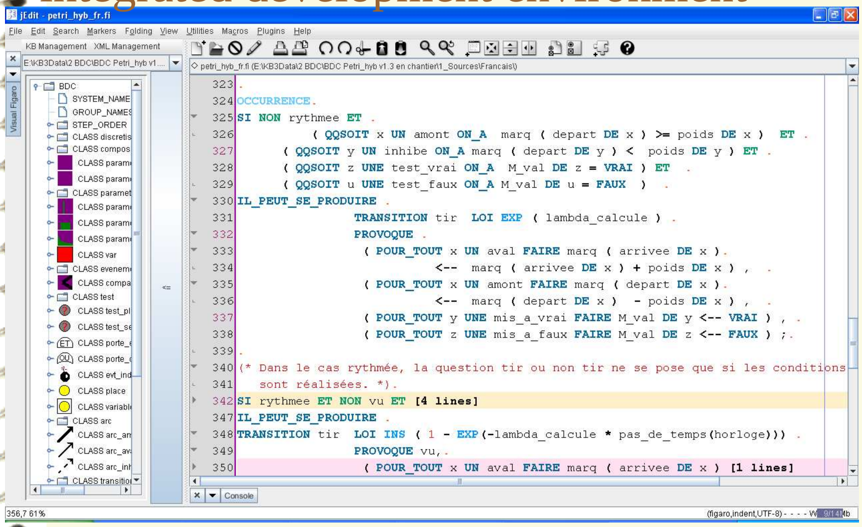
Includes:

- standard Petri nets
- Boolean messages
- Boolean functions on messages
- Randomly distributed parameters
- Continuous variables
- Special behavior of timed transitions

KB size (lines of FIGARO language):
- Petri nets: 215 lines
- Hybrid Petri nets: 405 lines

Marc Bouissou (EDF-R&D / ECP)

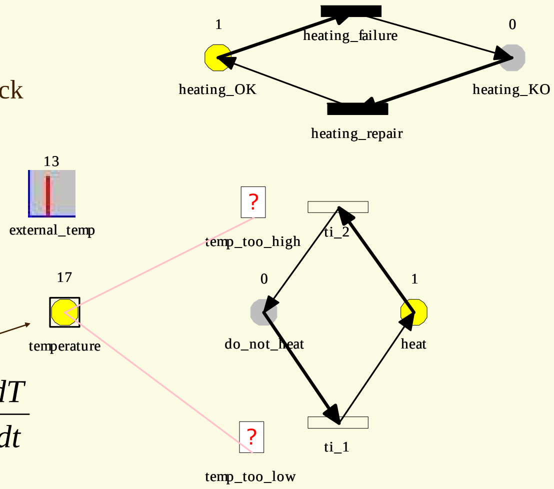
The Petri_hyb KB in the Visual Figaro integrated development environment



Marc Bouissou (EDF-R&D / ECP)

A heating system: model with the « Hybrid Petri net » KB

Any model built with this KB includes a clock if some transitions are "cadenced"



The expression of $\frac{dT}{dt}$ is input here

Marc Bouissou (EDF-R&D / ECP)

YAMS features (1/2)

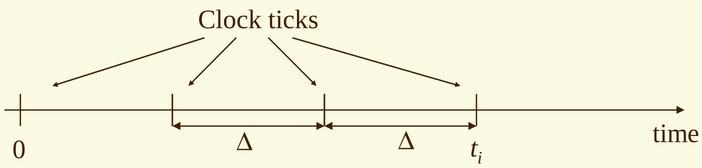
- Event driven Monte-Carlo simulation
- Distributions for times to events
  - EXP, T_C, UNI, TRIANG, ERL, WEIBULL, GUMBEL, FRECHET, PARETO, NORMAL, LGN, GAMMA, BETA
  - POINT
  - CYCLE
- Distributions « with memory »
- Conditional dist.: $F_{T0}(t) = \Pr(X \leq t \,/\, X > T0)$

YAMS features (2/2)

- Input language: Figaro 0 + extensions
- Extensions
  - Usual maths functions
  - RAND
  - CURRENT_DATE
  - HAS_BEEN_TRUE (bool_expression)
  - SOJOURN_TIME (bool_expression)
  - INTEGRAL (real_expression)

First solving method : time discretization

Principle of time discretization



At clock tick i, perform the following calculations :

$X(t_i) \leftarrow X(t_{i-1}) + \Delta.g(X(t_{i-1}), I(t_{i-1}))$ (Deterministic value)

$I(t_i) \sim I(\Delta, I(t_{i-1}), X(t_{i-1}))$ (Random value)

If one of the variables has hit a threshold,
Change (X, I) as needed

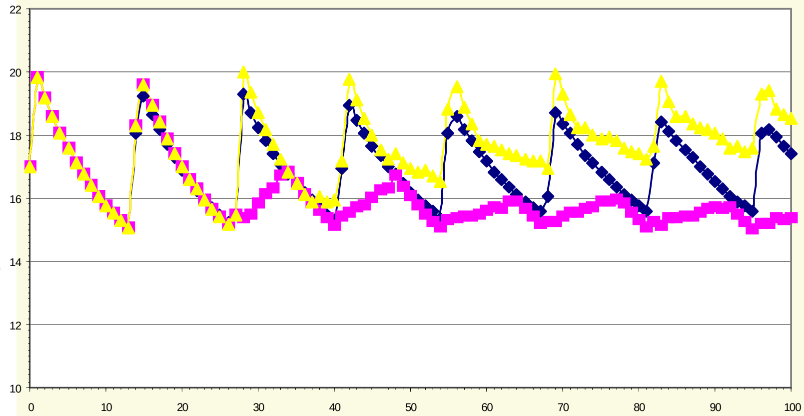
Advantages

- Easy to understand: the markovian dynamic reliability model is explicitly represented
- Relatively easy to implement

Problems…

- The probability that two random events happen in the same time interval is not zero
  - IS a problem if sequential behavior
- Cpu time: many calculations of random numbers instead of… one for an event which is not influenced by physical variables
- Non exponential distributions are hard to implement
  - Requires explicit function giving the hazard rate
  - Requires to add dimensions to X, corresponding to the starting date of random processes

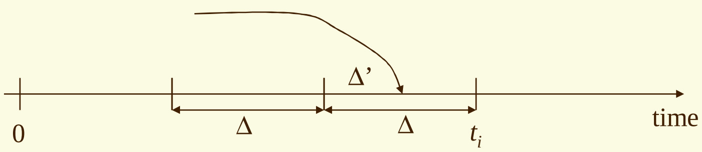
Results (1000 simulations)



cadenced timed transitions, time discretization (step : 1mn) – cpu 305s

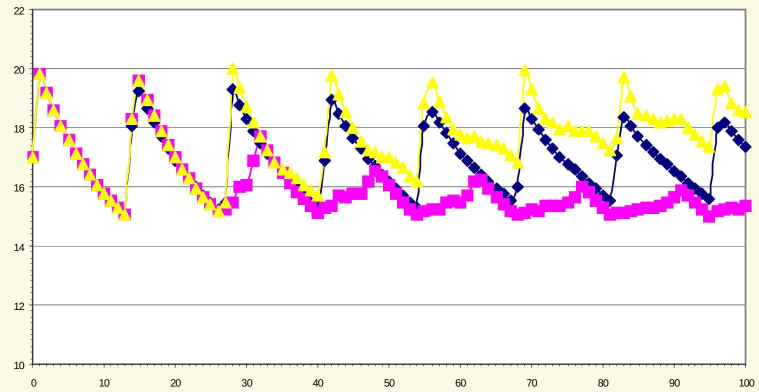
Improvement

Event with an occurrence rate independant from physical variables



- Saves many random numbers calculations
- Avoids (in most cases) the problem of random events falling in the same time interval
- But requires an intermediary calculation for the state of the whole system with time step Δ’

Results (1000 simulations)



timed transitions not cadenced, time discretization (step : 1mn) – cpu 206s

Second solving method : state space discretization

Principle

X_d = discretized version of X

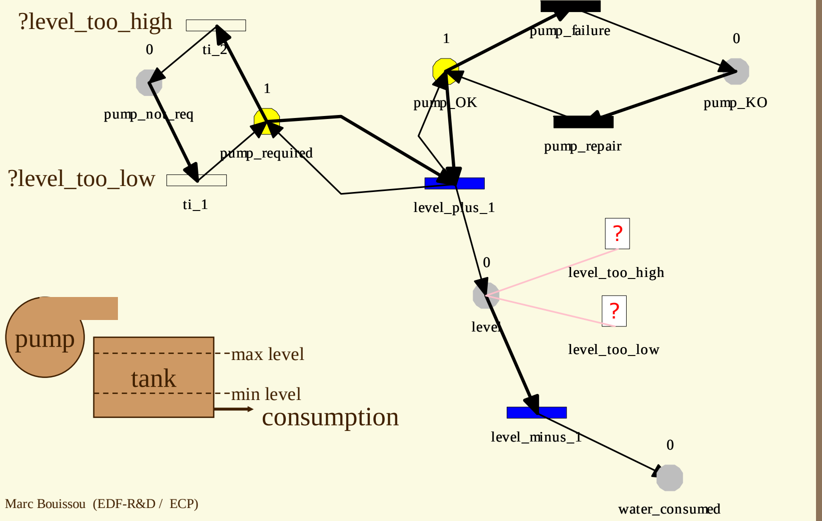
$$\frac{dX}{dt} = g(X, I)$$

$$\Delta t_i = \frac{\Delta x_d^i}{g_i(X, I)}$$ ⇒ Time before next change of X_d =min(Δt_i)

One can then perform a standard event driven simulation, each change of one of the continuous variables causing an « event » in the scheduler

If the model is a Petri net there must be two timed transitions for each variable (to increment/decrement it)

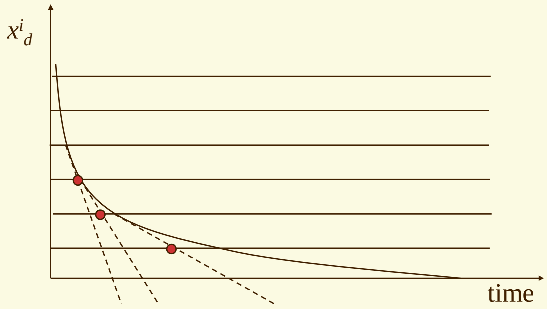
Example



What if the deterministic variables are not linear in time ?

$$\frac{dX}{dt} = g(X, I)$$

$$\Delta t_i = \frac{\Delta x_d^i}{g_i(X, I)}$$



At each change of X_d, Δt must be re-evaluated

Example: exponential evolution $\frac{dx}{dt} = kx \Rightarrow \Delta t \approx \frac{\Delta x_d}{kx_d}$

(exact solution: $\Delta t = \frac{1}{k} Ln\left(1 + \frac{\Delta x_d}{x_d}\right)$)

Marc Bouissou (EDF-R&D / ECP)

Space discretization

Unit = 1/10 °C

$$\frac{\Delta T}{\Delta t} = heater\_on(t) \times 50 - (T(t) - 130) \times 0.1$$

Incrementing transition:

$$\Delta T = +1 <=> \Delta t = \frac{1}{50}$$

$$or\ \Delta t = +\infty$$

(depending on the heater state)

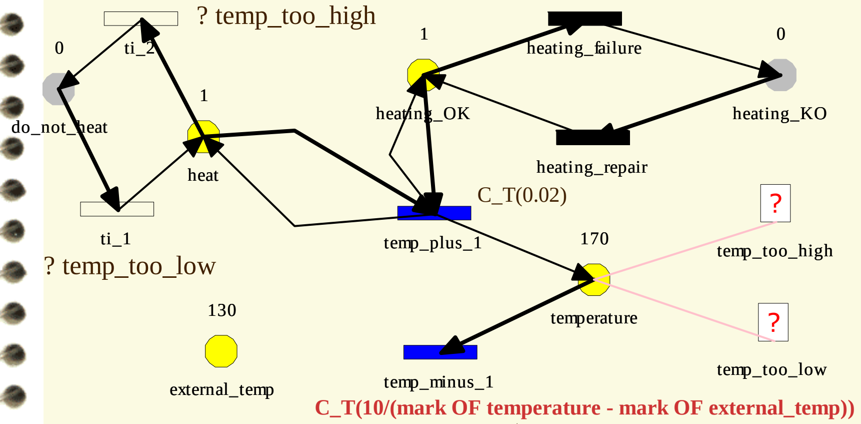
Decrementing transition:

$$\Delta T = -1 <=> \Delta t = \frac{10}{(T(t) - 130)}$$

To simplify, we will not consider initial values of $T<T_E$ => T will always be > T_E

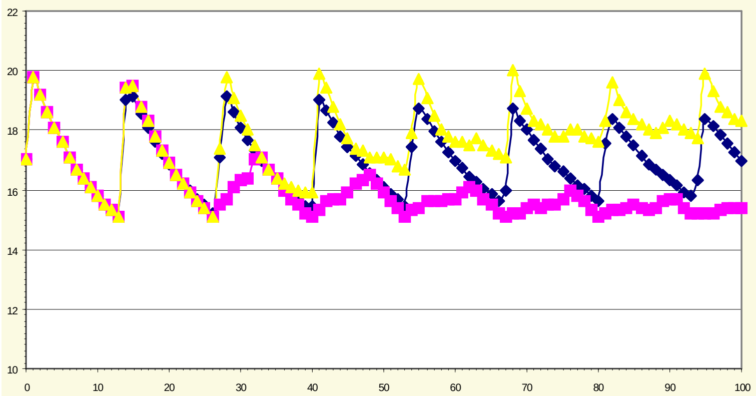
Marc Bouissou (EDF-R&D / ECP)

A heating system (method 2 – with a Petri net)



Only non-standard feature of this Petri Net

Results (1000 simulations)



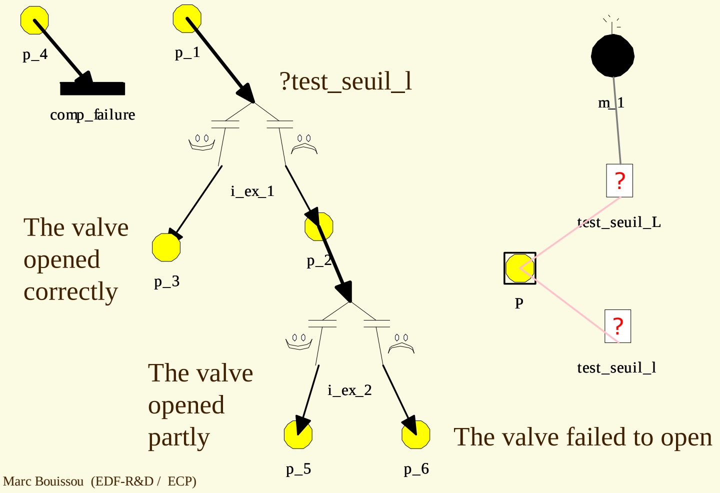
timed transitions not cadenced, value discretization (step : 0.1°C) – **cpu 8s**

Advantages/drawbacks

- Advantages
  - Can be implemented with (nearly) standard discrete system simulation tools
  - Non exponential distributions easy to implement
  - Precision can be improved if analytical solution of differential equations known
  - Discretization can be chosen in order to put thresholds exactly « on » discrete values
- Drawbacks
  - It is impossible (?) to model phenomena such as the increase of a failure rate with temperature

The Dynrel_1 test case

The model



Marc Bouissou (EDF-R&D / ECP)

Initial characteristics and derivative of P

Type	Object	Family	Characteristic	Pivot profile	Profil1
test_seuil	test_seuil_L	Constant	min	0	-1
test_seuil	test_seuil_l	Constant	min	0	-1
transition_exp	comp_failure	Attribute	lambda	0.001	0.04
transition_exclusive	i_ex_1	Attribute	gamma	0.01	0.06
variable_continue	P	Constant	pas_discretisation	1	0.1
transition_exclusive	i_ex_2	Attribute	gamma	0.01	0.33333
discretisation_temps	horloge	Constant	pas_de_temps	1	0.5
place	p_1	Attribute	marq	0	1
place	p_4	Attribute	marq	0	1
variable_continue	P	Attribute	V	0	1
test_seuil	test_seuil_l	Constant	max	0	2.99
test_seuil	test_seuil_L	Constant	max	0	3.99

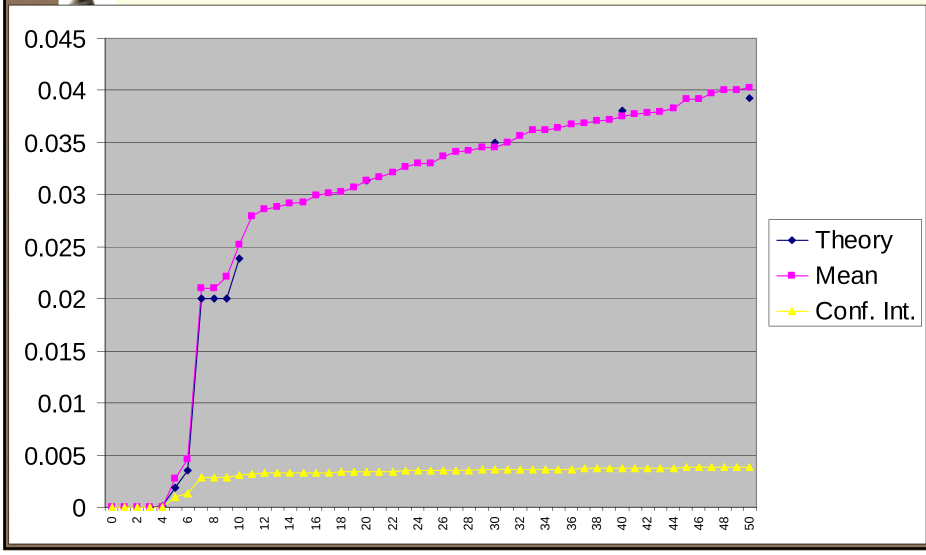
Derivative of P

V(P)*(0.2 * (marq(p_4)=1 ET (marq(p_1)=1 OU marq(p_6)=1)) +
-0.25* (marq(p_3)=1 ET marq(p_4)=1) +
-0.1 * (marq(p_5)=1 ET marq(p_4)=1) +
0.35* (marq(p_4)=0 ET (marq(p_1)=1 OU marq(p_6)=1)) +
-0.1 * (marq(p_3)=1 ET marq(p_4)=0) +
0.05* (marq(p_5)=1 ET marq(p_4)=0))

Marc Bouissou (EDF-R&D / ECP)

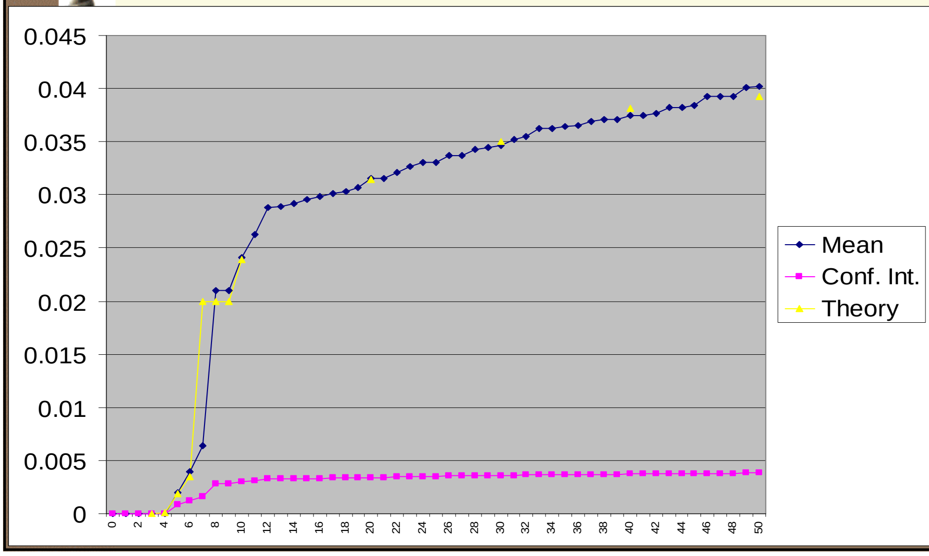
Pr(P>L)=f(t) (10000 sim., 11mn51s)

Discretization step: 6mn, no clock on exponential transitions



Pr(P>L)=f(t) (10000 sim., **24s**)

Discretization step: 0.1 (on P), no clock on exponential transitions



Conclusion

- Thanks to KB3, YAMS and the hybrid Petri Net KB, it is easy to solve various Dynamic Reliability problems
- A single model -> various simulation strategies
- Pure time discretization gives the poorest results in CPU time for comparable precision

Marc Bouissou (EDF-R&D / ECP)

Thank you for your attention !

To download the Visual Figaro, KB3 and YAMS tools:

look for: "Visual Figaro" in Google

Visual Figaro: open source project
KB3: free – unlimited in time - size limit = 80 objects
YAMS: free – no limitation

Marc Bouissou (EDF-R&D / ECP)