model futures
www.modelfutures.com

presents:

IntegratedEA

STRATEGY • OPERATIONS • TECHNOLOGY

www: http://www.integrated-ea.com
HashTag: #IEA12
Twitter: @IntegratedEA

VEGA
A Finmeccanica Company

NiTEWORKS

Abatis

TEAM ENSURE

de&s

Generic
YOUNG SPIRIT | SENIOR MIND

MooD
SMARTER DECISIONS

Reply
glue

mega

Kirsten Sinclair
SyntheSys Systems Engineers

Spicing-up IBM’s Enterprise Architecture tools with Petri Nets

“Too much spice?”

On Today's Menu

- Appetiser: **Background**
- Starter: **Use of Models**
- Main Dish: **Petri Nets**
- Breather: **Case Study**
- Dessert: **Benefits & Summary**
- Coffee/Mints: **Questions**

Background

About me:

- Worked in a partnership with Durham University & SyntheSys Systems Engineers
- Aim: help address incompatibility of military information exchanges
- Research interests include: getting the most out of modelling!

About SyntheSys

- Founded 1987
- Based in Whitby, North Yorkshire
- Systems engineering expertise: former military & engineering personnel

Background

It's all about complex systems!

- Internet
- Embedded systems (e.g. mobiles, household appliances)
- Military 'close air support' mission
- Baggage handling (remember T5?)
- Health-care life support
- Car breakdown recovery system:

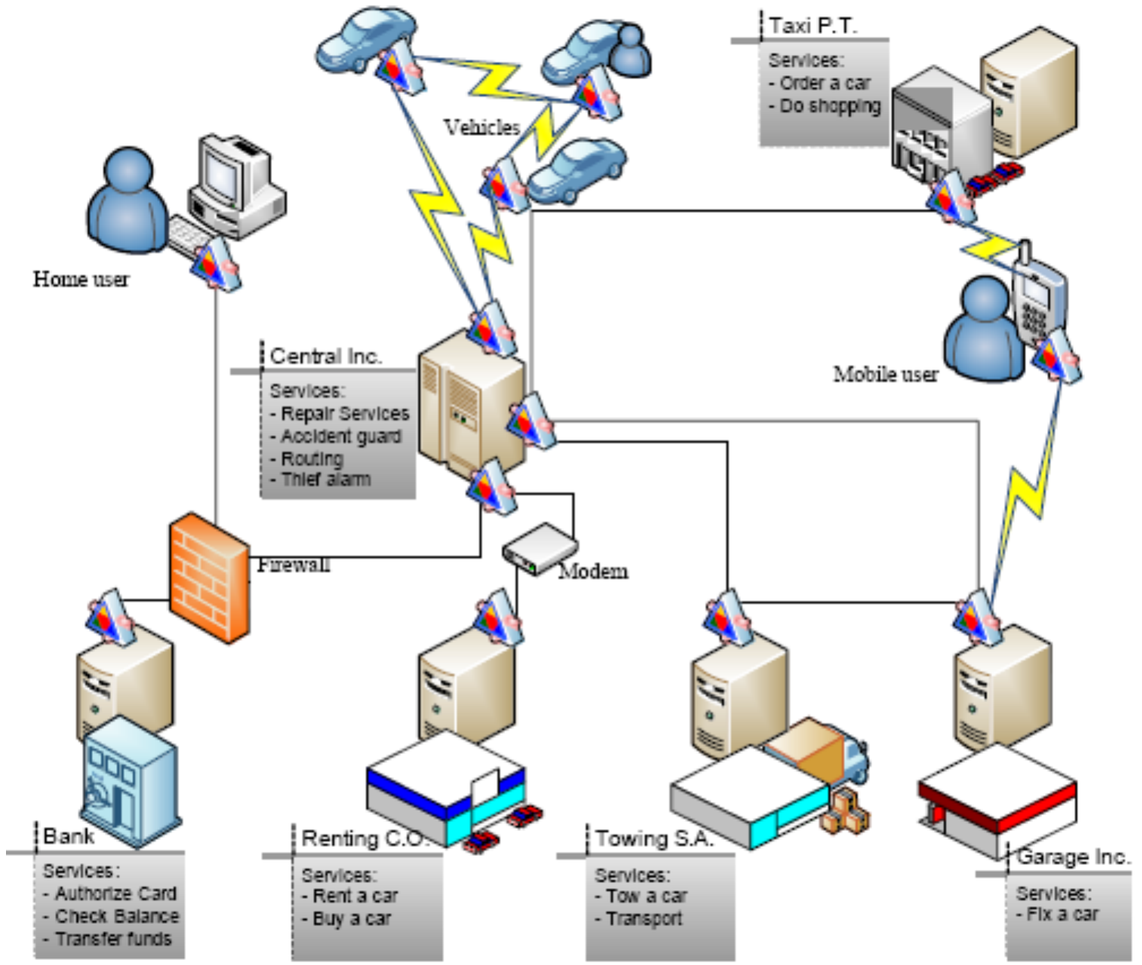
"It doesn't look good. Leave it with us overnight and hope it gets stolen."

Background

They’re difficult to develop:

- Components exist in parallel
- Unpredictable interaction (message loss, environmental event timing, activity sequencing)
- Huge number of possible execution paths
- Not all captured by designer

= Gaps in system specification



Background

We'd like them to be complete & correct:

- Particularly in critical systems (such as those in health, energy, military, and aircraft)

BUT when do we verify & validate them against requirements?

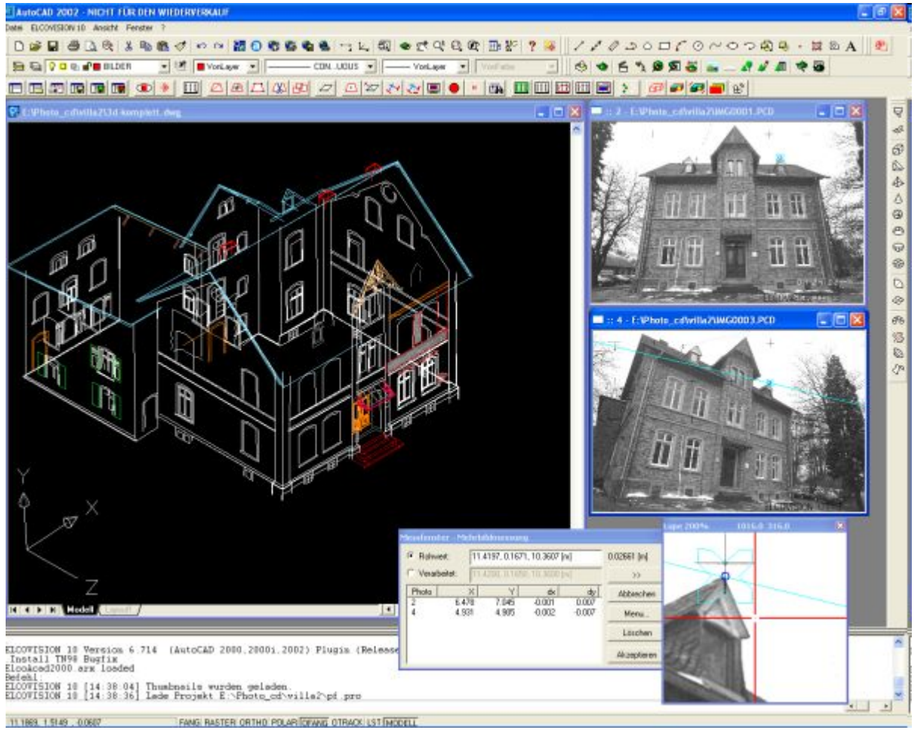
- Too late! This should be done early in the systems engineering lifecycle

Use of Models

Compare with building industry:

- Models used by architects at early stages of a building's development
- Can be paper (blueprints), 3-D (plywood), or computer-based (blueprints, 3-D)

Stakeholders can 'see' new building's function & 'look'



Use of Models

Models are used to:

- Check aesthetics, effects of turbulence & load, internal layout
- Abstract at different detail levels
- Allow stakeholders to detect & correct errors and omissions in building specification

When? Before costly construction phase!

Use of Models

Applied early in systems engineering lifecycle:

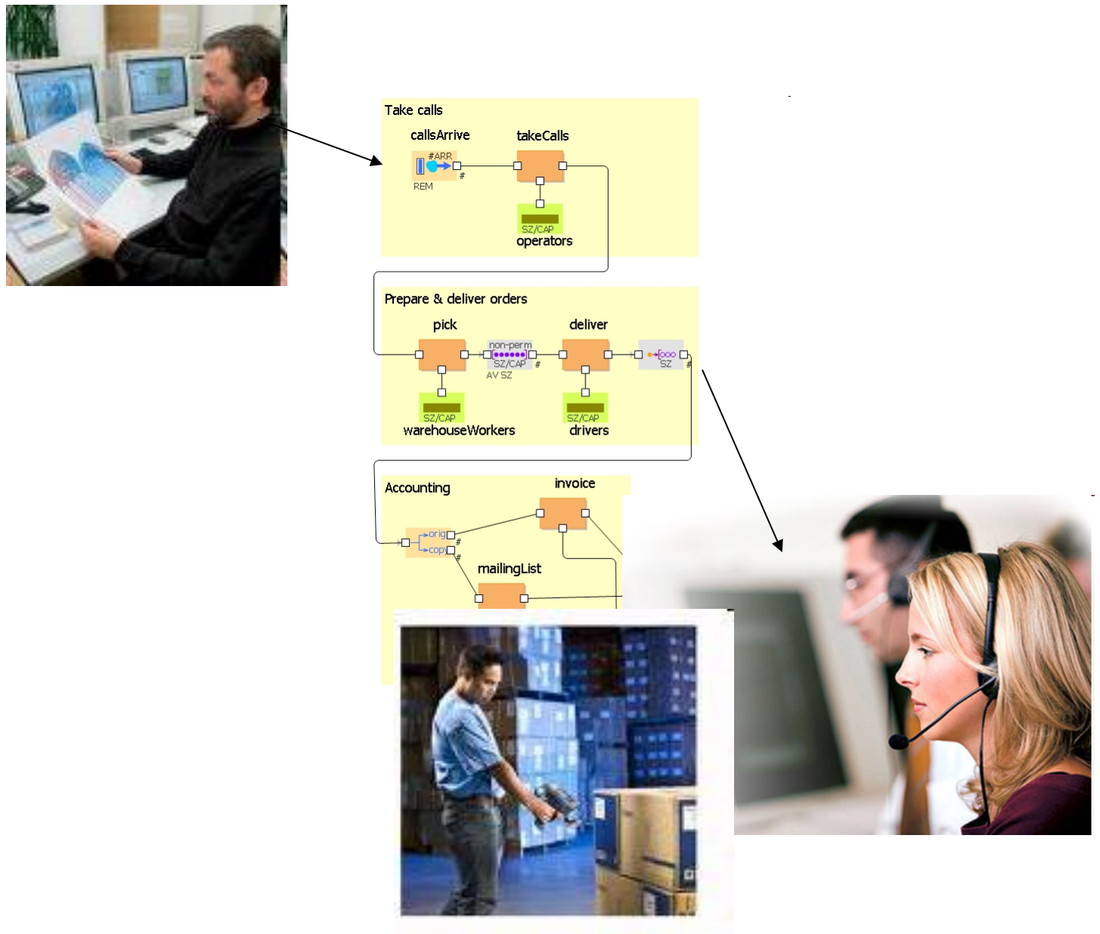
- Computer-based graphical models can help stakeholders understand the system being specified

How?

Bring models to life, i.e. make them interactive

Automatic exploration of a model can help produce:

- Complete specification
- Correct specification



Use of Models

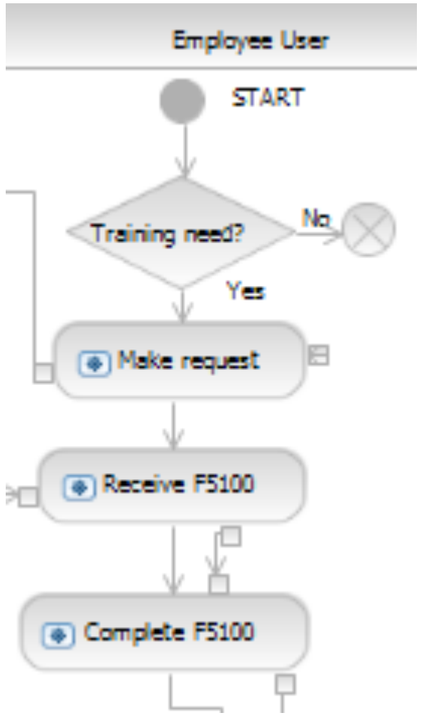
Q: What modelling languages (text & graphical) have you come across?

Use of Models

A: maybe… Unified Modelling Language (UML), SysML, Data Flow Diagrams, IDEF, sticks/circles & pencil, Foundational UML…

- UML: very popular graphical modelling language (especially software engineering)
- SysML becoming popular for systems engineering (extends UML)
- Supported by commercial & open source tools (e.g. IBM Rational, Sparx, Eclipse UML2 Tools, ArgoUML)

What's your experience of these? Useful? Mandated?



Use of Models

In developing a comprehensive complex systems engineering method for SyntheSys, used:

- Architecture frameworks (MoDAF, DoDAF, NAF)
- UML, SysML, BPMN
- Enterprise architecture tools from IBM Rational, Salamander, Sparx

Found:

- Lots of STATIC!
- UML state-machine & sequence diagrams model single execution thread
  - concurrency & abstraction not usually supported (scalability issues too)

IBM Rational Tool/ Verification & Validation	System Architect	Software Architect/ Rhapsody/Tau
Simulation (untimed)	BPMN	State-machine; Activity
Simulation (timed)	BPMN	State-machine
Exhaustive path-checking		State-machine (limited)

Use of Models

Main features of UML as a modelling language

UML Behavioural Diagram/ Specification Feature	State Machine	Sequence	Activity
Timing	Yes (via profile)	Yes (via profile)	Yes (via profile)
Scalability	Low	Low	Improved over state-machine & sequence
Process-based	No	No	Yes
Formal dynamic semantics	No	No	No
Verification & validation	Static	Static	Static

Lacks clarity for execution or exhaustive checking!

Cue: Petri Nets - how many of you have come across these?

Petri Nets: Introduction

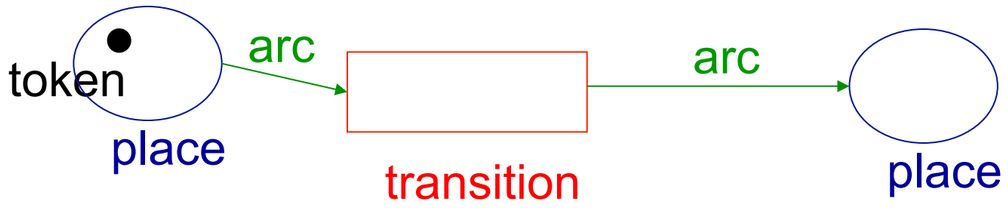
- Introduced by Carl Adam Petri
- Combine maths with graphical features
- Maths basis allows precise analysis of modelled behaviour
- Graphical features allow you to visualise behaviour

Been used to model:

- Biochemical processes
- Communication protocols for digital telephone networks
- Lab, air traffic & manufacturing control systems

Petri Nets: Introduction

- Graphical modelling language (‘sticks, circles & squares’)
- 4 main features: places, tokens, arcs & transitions
  - places are holders of something (‘tokens’)
  - transitions are activities
  - arcs direct connections between these
- Tokens are dynamic feature – define behaviour of model
- Spread of tokens = state of model
- State-/transition-/data-oriented



Petri Nets: How do they work?

Transitions have:

- at least 1 input place (data IN)
- at least 1 output place (data OUT)

Places hold:

- zero or number of tokens

Arcs can also define rules for token circulation

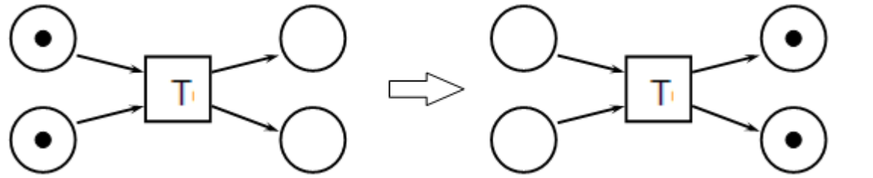
Petri Nets: How do they work?

- Tokens, places & arcs represent pre- & post-conditions of behaviour
- Transitions ‘switch in state’ (fire) to post-condition (if pre-condition is met)

Example: the ‘Integrated EA conference registration’ system

Transition ‘grant access’ needs pre-condition:
1 ‘delegate details’ & 1 ‘payment’!

If it gets these, it allows switch in state to:
Post-condition: 1 ‘badge’ & 1 ‘goodie pack’



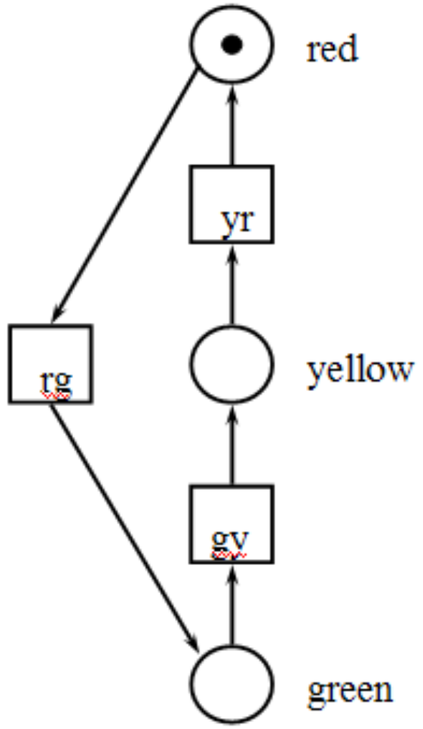
T good to switch SWITCH post-condition

Petri Nets: How do they work?

- States of a process are modelled by tokens on places
- Activities of a process are modelled by transitions
- Tokens can represent things (people, machines), states, or information
- Places can represent locations, buffers, conditions, or states

Petri Nets: Example

- Traffic lights example (with default pre-condition):



Petri Nets: Example

- Chemical reaction with non-default pre-condition:

BEFORE

AFTER

Enabled transition

Fired transition

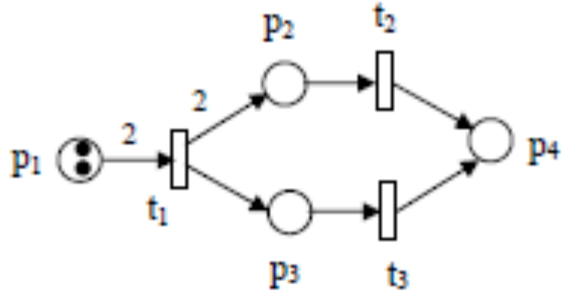
Petri Nets: Some maths

Formal, unambiguous syntax and semantics, e.g.

Net = (P, T, I, O, M_0), where

1. P = {p1, p2, …, p_m} is a finite set of places;
2. T = {t1, t2, …, t_n} is a finite set of transitions;
3. I: P x T -> N is an input function that defines directed arcs from places to transitions, where N is a set of non-negative integers;
4. O: T x P -> N is an output function that defines arcs from transitions to places;
5. M_0: P -> N is the initial marking

In contrast, UML & BPMN use semi-formal syntax & semantics



P = {p1, p2, p3, p4}
T = {t1, t2, t3}
I(t1, p1) = 2, I(t1, pi) = 0 for i = 2, 3, 4
I(t2, p2) = 1, I(t2, pi) = 0 for I = 1, 3, 4
I(t3, p3) = 1, I(t3, pi) = 0 for I = 1, 2, 4
O(t1, p2) = 2, O(t1, p3) = 1, O(t1, pi) = 0 for I = 1, 4
O(t2, p4) = 1, O(t2, pi) = 0 for I = 1, 2, 3
O(t3, p4) = 1, O(t3, pi) = 0 for I = 1, 2, 3
M_0 = (2 0 0 0)

Petri Nets: Why use them?

Dynamic event-driven complex systems have:

- Control sequence
- Concurrency
- Conflict
- Mutual exclusion
- Decision-making
- Synchronisation
- Priorities

Petri nets can capture these!

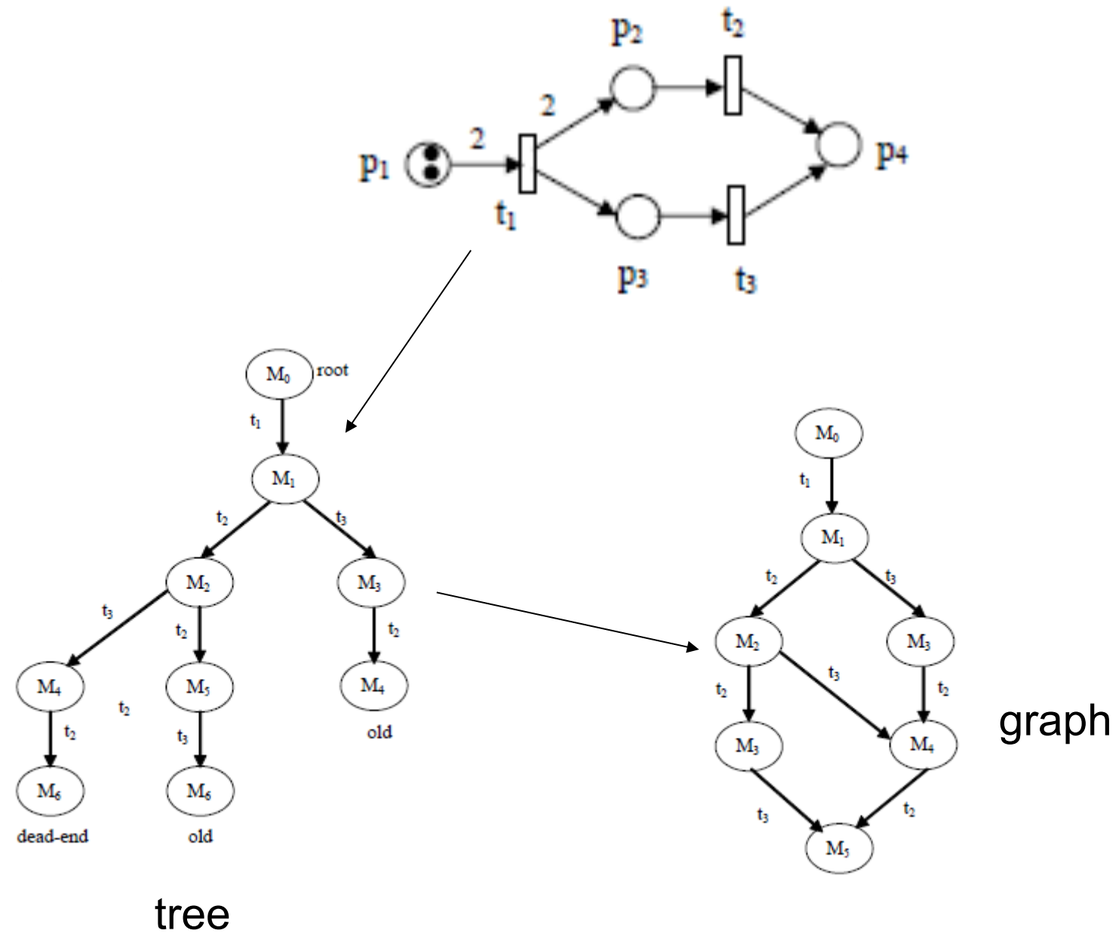
Petri Nets: Why use them?

Not only about ‘seeing’ a system visually (& statically):

- Need to analyse model for further check of modelled behaviour & logic

2 main analysis approaches:

- Simulation
- Reachability tree

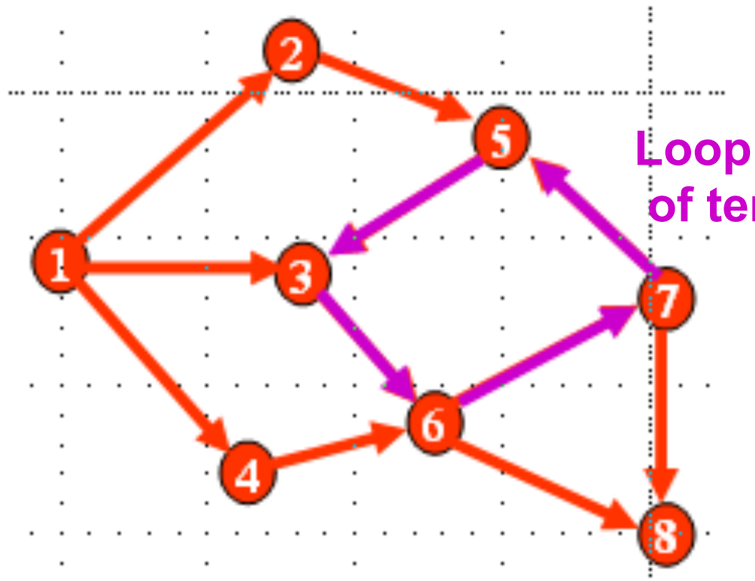


Petri Nets: Why use them?

Are there any deadlocks?

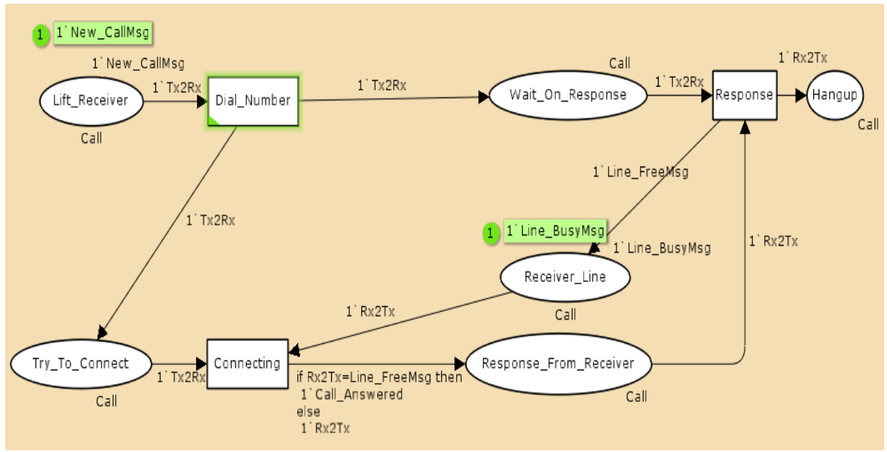
Is it possible to reach a particular state?

Does the system behave as expected?

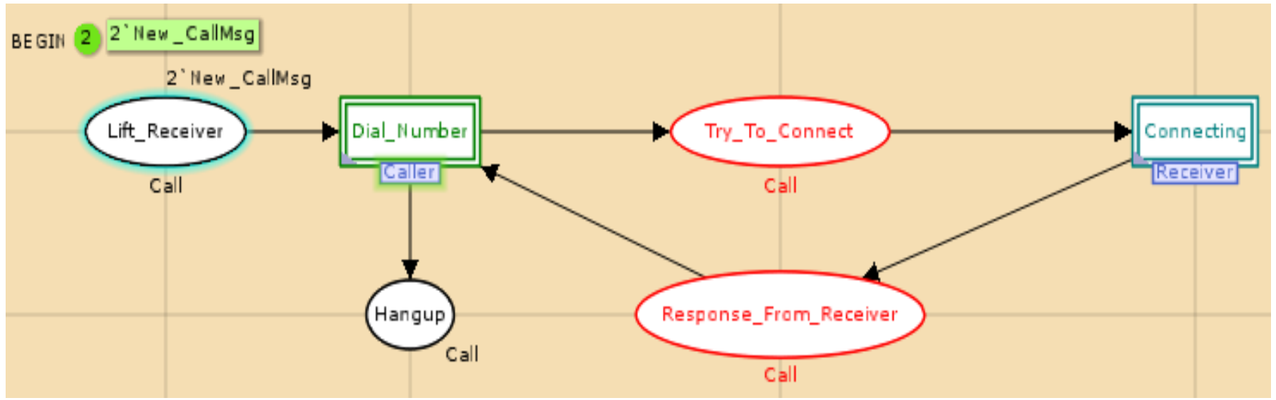


Petri Nets: Coloured (high-level) variant

- Combination of low-level Petri Nets & programming language
- Allow creation/manipulation of data types (colours)
- Add timing
- Scale better
- Model abstraction via hierarchy
- Can still simulate & do reachability tree/graph
  - Performance analysis too

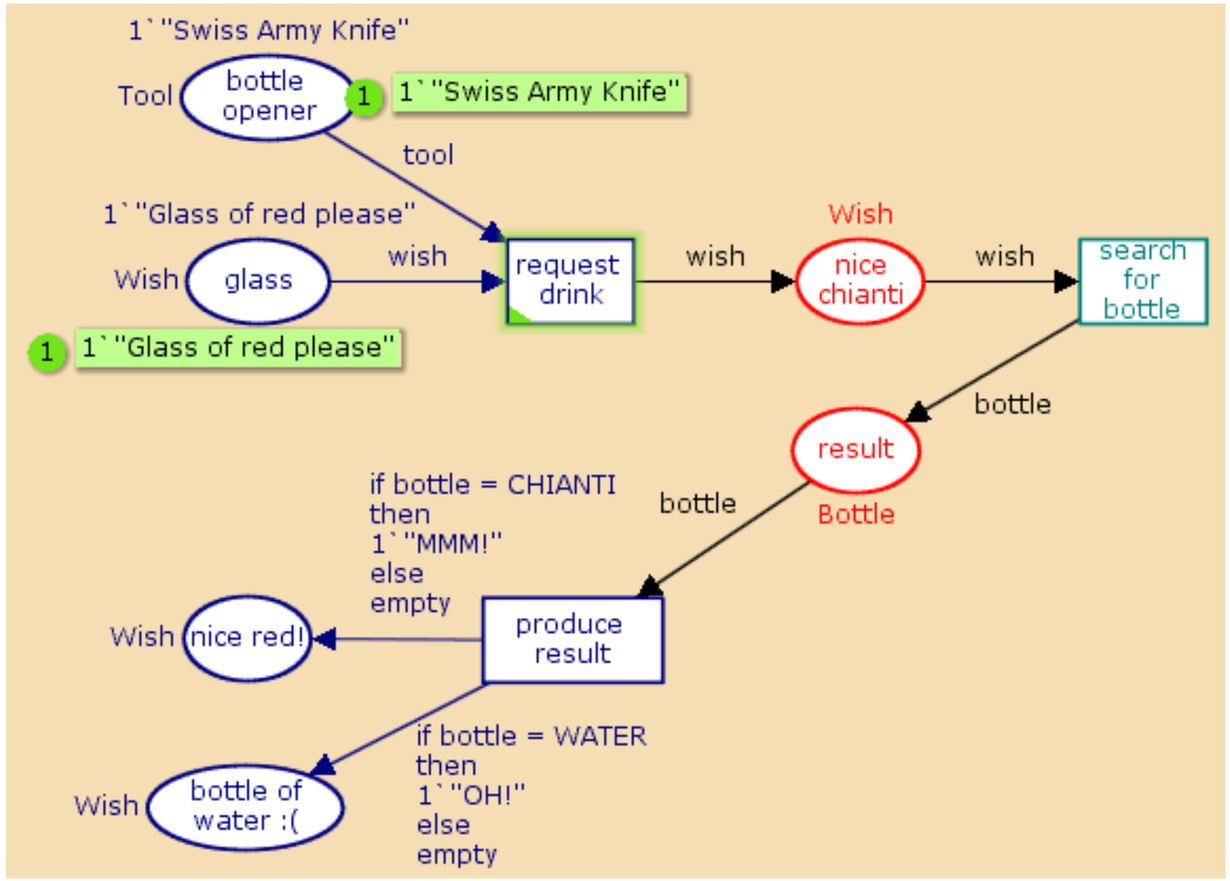


'flat' model, i.e. no abstraction



model with hierarchy

Petri Nets: Coloured (High-level) variant



Applications of Coloured Petri nets:

- Communications protocols; supply chain; penetration testing; critical process modelling…

Example Coloured Petri net

- Models process of getting a drink
- Ideal outcome is a glass of red!
- Can simulate to check outcomes & markings

Petri Nets: Coloured (High-level) variant

Interactive & Automatic Modes of Simulation

Helps to highlight:

- Prioritising token addition & removal
- Logic problems (looping, missing specification)
- Inappropriately coloured variables

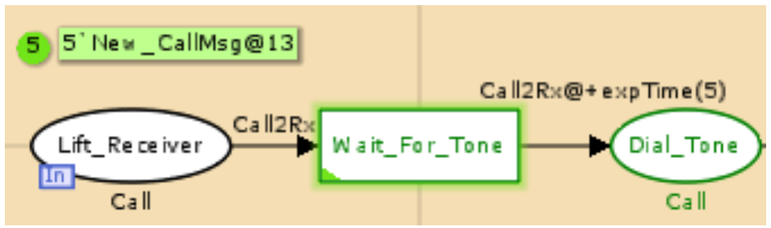
Can also look at performance of the system

Petri Nets: Coloured (High-level) variant

Performance analysis of model (add timing)

- Un-timed models insufficient to capture overall behaviour
- Allocate time of entry, processing & transfer delays to tokens
- Specify ordering of tokens over & above activity sequence
- Check model reaches desirable behaviour states (& recovery from undesirable) within realistic time & resource estimates
- Improve process efficiency

```
Phone Process
  colset Line = with Eng;
  colset Call = with New_CallMsg | Line_BusyMsg | Line_FreeMsg | Not_AnsweredMsg | Call_AnsweredMsg
                  | Engaged_Msg | Ringing_Msg timed;
  colset CallRec = record callType:Call*conn_start:INT*conn_end:INT timed;
```



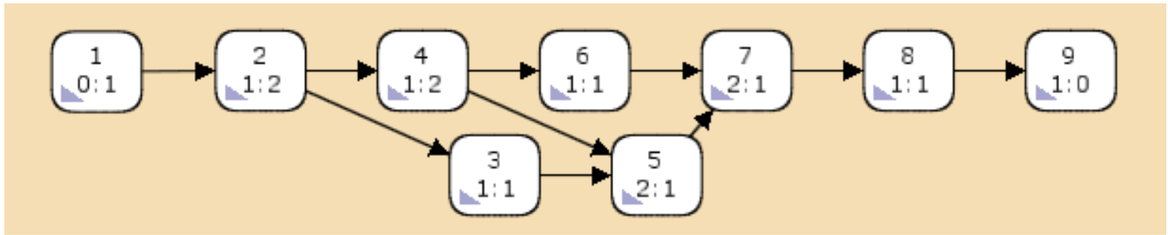
Petri Nets: Coloured (High-level) variant

Simulation helps to give:

- shared understanding via scenarios & initial verification

BUT it is NOT exhaustive

CUE: Reachability Analysis of Coloured Petri Nets!



- Construct reachability tree/graph for model
  - all possible switch sequences & model states
- Calculated automatically
- Answer verification & validation questions on logic & behaviour of model
- Does model have a desired state?

```
9:
Call_Business_Process'Lift_Receiver 1: empty
Call_Business_Process'Wait_On_Response 1: empty
Call_Business_Process'Try_To_Connect 1: empty
Call_Business_Process'Receiver_Line 1: 1`Line_FreeMsg
Call_Business_Process'Response_From_Receiver 1: empty
Call_Business_Process'Hangup 1: 1`Line_BusyMsg++
1`Call_Answered
```

Petri Nets: Coloured (High-level) variant

Reachability Analysis Report

- Relates to standard Petri net properties
- Errors in model highlighted here
- Shows:
  - number of markings, switches & time to calculate tree/graph
  - max/minimum number of tokens on places (loops?)
  - markings that can always be reached
  - dead markings (desired?)
  - dead transitions (redundancy or error?)

Can configure ad-hoc queries allowing more detailed checking of the model too

Reachability/State Space

```
Nodes:   9
Arcs:    10
Secs:    0
Status: Full

Home Markings                  [9]

Dead Markings                  [9]
Dead Transition Instances      None
Live Transition Instances      None

No infinite occurrence sequences.
```

Best Integer Bounds

Place	Upper	Lower
Hangup	2	0
Lift_Receiver	2	0
Receiver_Line	1	0
Response_From_Receiver	1	0
Try_To_Connect	2	0
Wait_On_Response	2	0

Best Upper Multi-set Bounds

Hangup	1`Line_BusyMsg++1`Call_Answered
Lift_Receiver	2`New_CallMsg
Receiver_Line	1`Line_BusyMsg++1`Line_FreeMsg
Response_From_Receiver	1`Line_BusyMsg++1`Call_Answered
Try_To_Connect	2`New_CallMsg
Wait_On_Response	2`New_CallMsg

Case Study

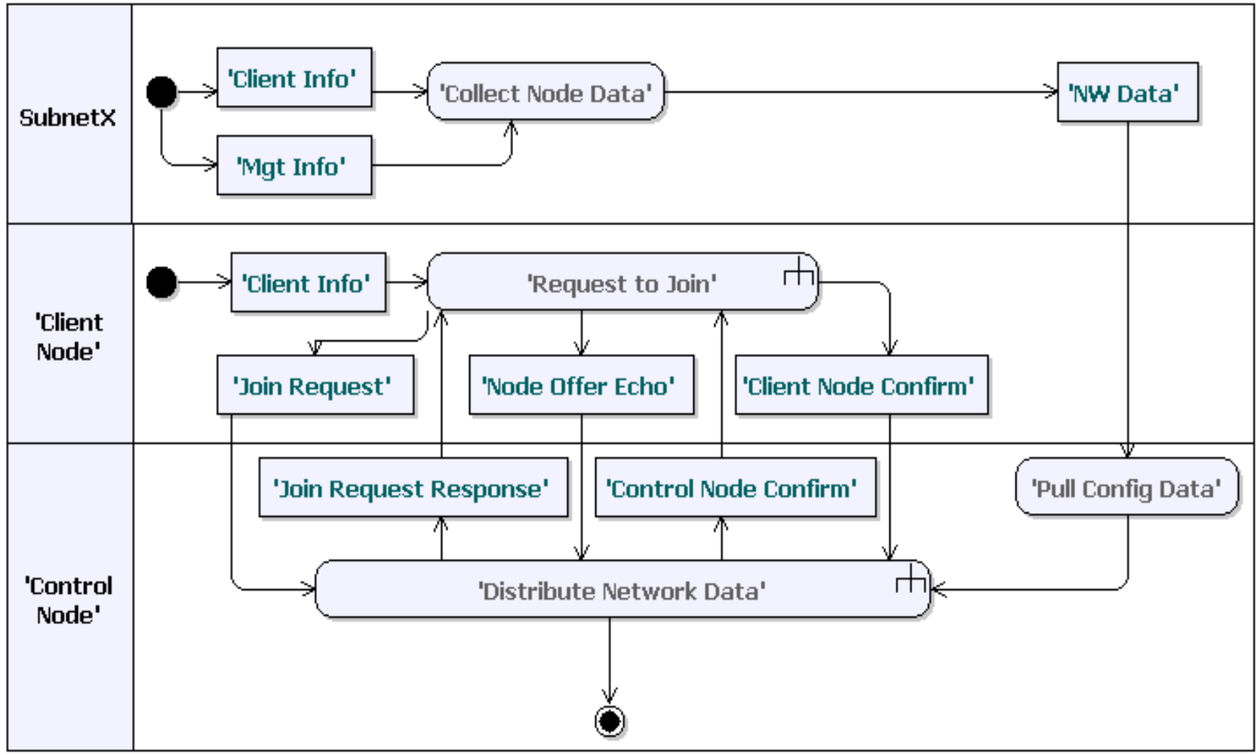
Military domain: 'Exchange Network Parameters':

- Dynamic networking parameters configuration
- MIL-STD-188-220D definition:

'If a station moves into or through different subnetworks, the Data Link addresses and operating parameters may change in each subnetwork, and exchange network parameters will automate the change of parameters and address from one subnetwork to another'

- Standard modelled initially using UML

Further verification & validation: timed coloured Petri nets



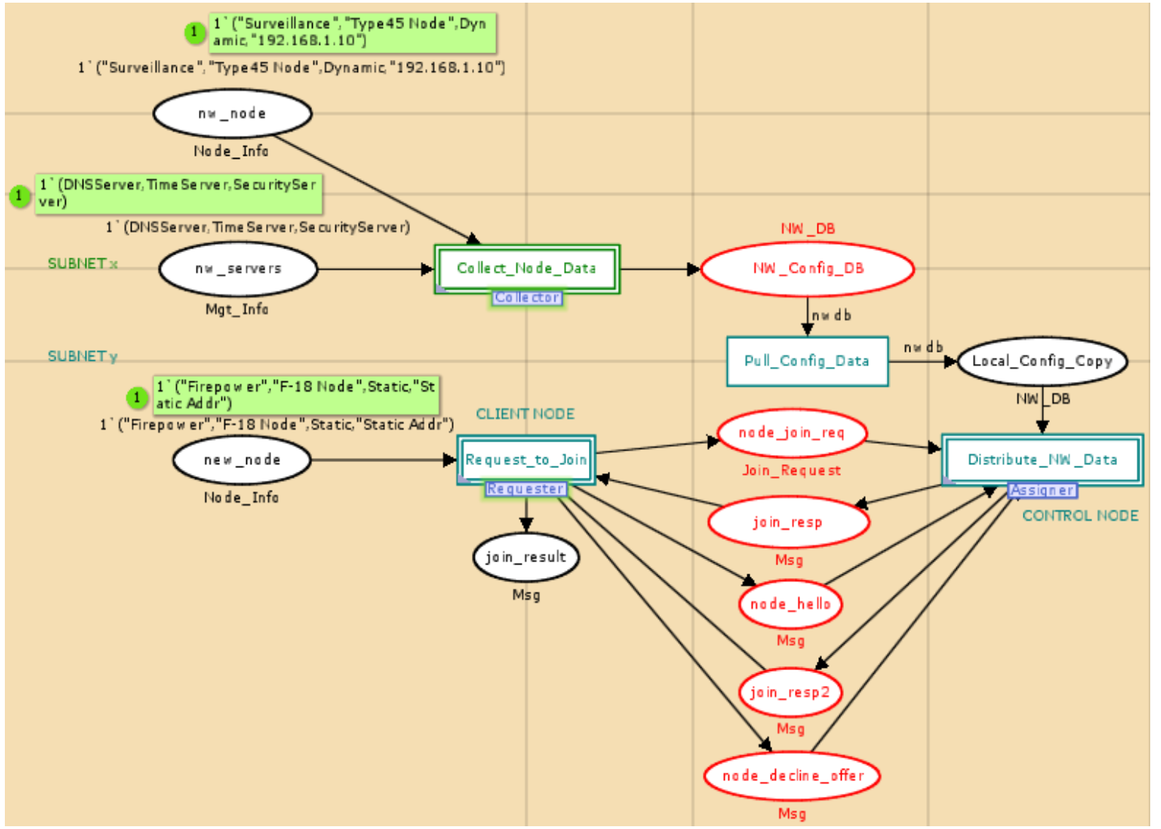
Case Study

Errors in military standard included:

- Reject response (issued to the client node)
- Error recovery (timeouts and retry attempts)
- Rejection of unauthorised control node/client node messages

Detected using:

- Simulation, performance analysis, checking model properties



Case Study

Performance Analysis of model

- Calculated average successful join request duration based on:
  - message transmission time of communications media
  - duration of activities
  - probability of transmission failure in join request process
  - probability of join request being accepted
- Linked this to cost of providing communications media (analysis-of-alternatives)
- Automatic simulation replications used to calculate this for multiple client requests

Discrete Distribution (Data Collection Monitor: Req_Success_Monitor)

REP 1 (30 time units)

1	229	229.000000	229	229
2	461	230.500000	206	255
1	219	219.000000	219	219

6 possible successes, 3 simulation replications:
4 successes, avg. 227.25 time units.

REP 2 (15 time units)

0	0	0.000000	0	0
2	259	129.500000	129	130
1	102	102.000000	102	102

6 possible successes, 3 simulation replications:
3 successes, avg. 120.33 time units.

Benefits of Petri Nets

Specification Feature	Timed Coloured Petri Net	EA Tool UML implementation
Concurrency	Yes	No
Performance analysis	Yes	Some (via profile, usually limited state-machine simulation-only)
Formal dynamic semantics	Yes	No
Verification & validation • Simulation • Model-checking	 Yes (timed & untimed) Yes	 Some (vendor-specific & usually untimed) Low (limited state-machine)

Design verification & validation? Think Petri Nets!

Summary

We've looked at:

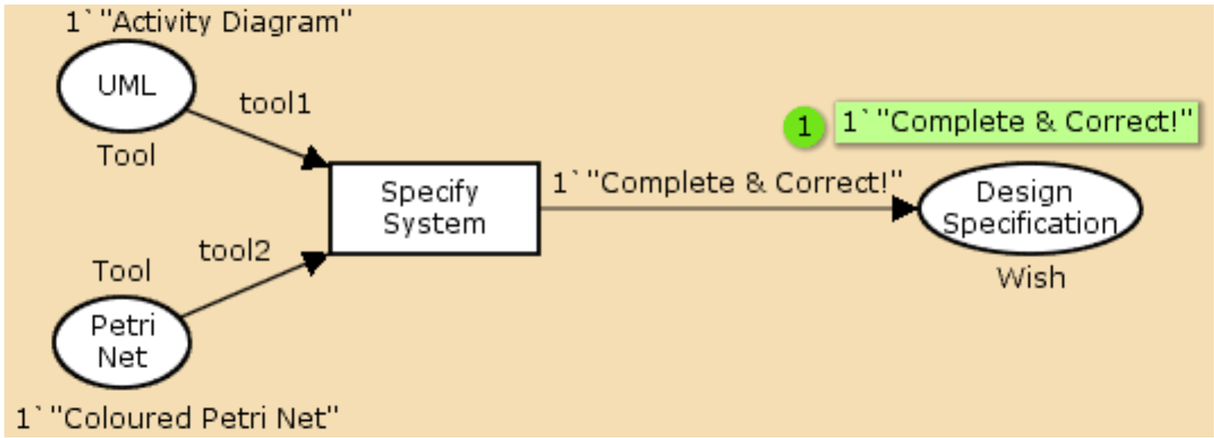
- Background to modelling
- Overview of Petri Nets (in particular timed coloured)
- Benefits of Petri Nets

Used in conjunction with EA tools like IBM's:

- Petri Nets enhance BPMN & UML

Increase confidence that:

- Shared understanding of system behaviour gained
- System design specification correct
- System design specification complete



Questions

Happy to take coffee/mints orders!

SyntheSys
MILITARY SYSTEMS