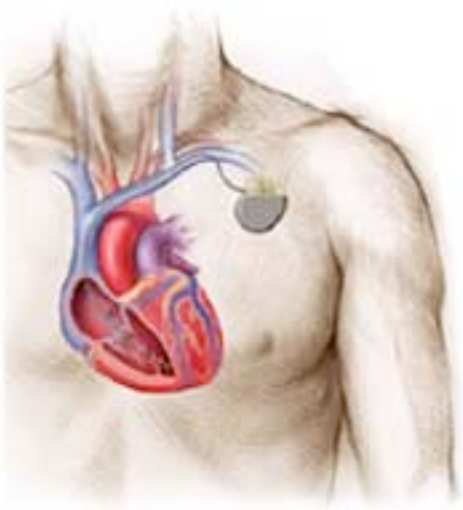
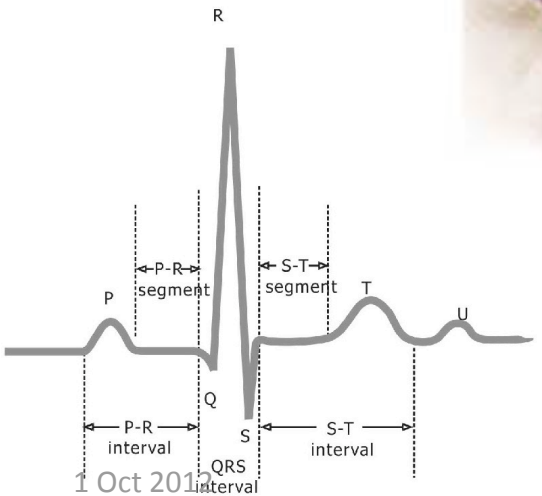
Modelling Cardiac Pacemaker Software: Towards Methodology for Complex Critical Systems

Mike Poppleton
Abdolbaghi Rezazadeh
Colin Snook





Roadmap

- Background & motivation
- Requirements ?
- Electrocardiology basics
- Requirements domain modelling – state diagrams
- Solution domain modelling – different state diagrams
  - Event-B *brief* recap
  - Modelling & refinement … towards method
- Further work

Cardiac pacemaker: a definition

- The device monitors and regulates a patient's heart rate.
- The device detects and provides therapy for bradycardia conditions.
- The device provides programmable, single- and dual-chamber, rate-adaptive pacing, both permanent and temporary.
- In adaptive rate modes, an accelerometer is used to measure physical activity, resulting in a sensor indicated rate for pacing the heart.

Boston Scientific. Pacemaker system specification. Technical report, 2007. http://sqrl.mcmaster.ca/pacemaker_spec.htm.

Background & motivation

One of the Grand Challenge case studies

Computer

First Steps in the Verified Software Grand Challenge

October 2006 (vol. 39 no. 10)

pp. 57-64

Jim Woodcock, University of York

DOI Bookmark: http://doi.ieeecomputersociety.org/10.1109/MC.2006.340

ABSTRACT

The computer science research community is collaborating to develop verification technology that will demonstrably enhance the productivity and reliability with which software is designed, developed, integrated, and maintained.

Motivation

"In 1996 (in the USA), 10% of all medical device recalls were caused by software-related issues. In June of 2006, software errors in medical devices made up 21% of recalls."

Jiang et al

In one study, 40% of cardiac complications in pacemaker-fitted patients were caused by the device

The specification (Boston Scientific): lots of

this (definitions):

5.4.2 Atrial Refractory Period (ARP)

For single chamber atrial modes, the Atrial Refractory Period (ARP) shall be the programmed time interval following an atrial event during which time atrial events shall not inhibit nor trigger pacing.

5.4.3 Post Ventricular Atrial Refractory Period (PVARP)

The Post Ventricular Atrial Refractory Period shall be available in modes with ventricular pacing and atrial sensing. The Post Ventricular Atrial Refractory Period shall be the programmable time interval following a ventricular event when an atrial cardiac event shall not 1. Inhibit an atrial pace. 2. Trigger a ventricular pace.

The specification: and lots of

this (configuration data):

3.6.3 Pace-Now State

Commanded emergency bradycardia pacing (Pace-Now) shall be available.
The Pace-Now Pace parameter values are as follows:

1. The mode Pace-Now pace parameter shall have a value of VVI.
2. The lower rate limit Pace-Now pace parameter shall have a value of 65 ppm ±8 ms.
3. The amplitude Pace-Now pace parameter s
V.
4. The pulse width Pace-Now pace parameter
±0.02 ms.

Parameter	AAT	VVT	AOO	AAI	VOO	VVI	VDD	DOO	DDI	DDD	A C C R
Lower Rate Limit	X	X	X	X	X	X	X	X	X	X	X
Upper Rate Limit	X	X	X	X	X	X	X	X	X	X	X
Maximum Sensor Rate											X
Fixed AV Delay							X	X	X	X	
Dynamic AV Delay							X			X	
Sensed AV Delay Offset										X	
Atrial Amplitude	X		X	X				X	X	X	X
Ventricular Amplitude		X			X	X	X	X	X	X	
Atrial Pulse Width	X		X	X				X	X	X	X
Ventricular Pulse Width		X			X	X	X	X	X	X	
Atrial Sensitivity	X			X					X	X	
Ventricular Sensitivity		X				X	X		X	X	
VRP		X				X	X		X	X	
ARP	X			X					X	X	
PVARP	X			X					X	X	

… ie very low-level, and little behaviour is specified!

sensed or paced ventricular event initiates a VRP. (A VRP is always part of the timing cycle of any pacing system with ventricular pacing and sensing.) The VRP prevents sensing of the evoked potential and the resultant T wave on the ventricular channel of the pacemaker. A sensed or paced ventricular event also initiates a refractory period on the atrial channel (PVARP).[13,14] The PVARP may prevent atrial sensing of a retrograde P wave (see "Endless-Loop Tachycardia," below), but the PVARP alone may not prevent sensing of far-field ventricular events in devices with an automatic mode-switching algorithm. The combination of the PVARP and the AVI forms the total atrial refractory period (TARP). The TARP, in turn, is the limiting factor for the maximum sensed atrial rate that the pacemaker can sense and, hence, track. For example, if the AVI is 150 milliseconds and the PVARP is 250 milliseconds, the TARP is 400 milliseconds, or 150 ppm. In this case, a paced ventricular event initiates the 250-millisecond PVARP, and only after this interval has ended can an atrial event be sensed. If an atrial event is sensed immediately after the termination of the PVARP, the sensed atrial event initiates the AVI of 150 milliseconds. On termination of the AVI, in the absence of an intrinsic R wave, a paced ventricular event occurs, resulting in a VV cycle length of 400 milliseconds, or 150 ppm. Programming a long PVARP limits the upper rate by limiting the maximum sensed atrial rate (Fig. 6.15).[15,16] If the native atrial rate were 151 bpm, every other P wave would coincide with the PVARP, not be sensed, and hence not

[Cardiac Pacing and ICDs, Kenneth A Ellenbogen and Mark A Wood]

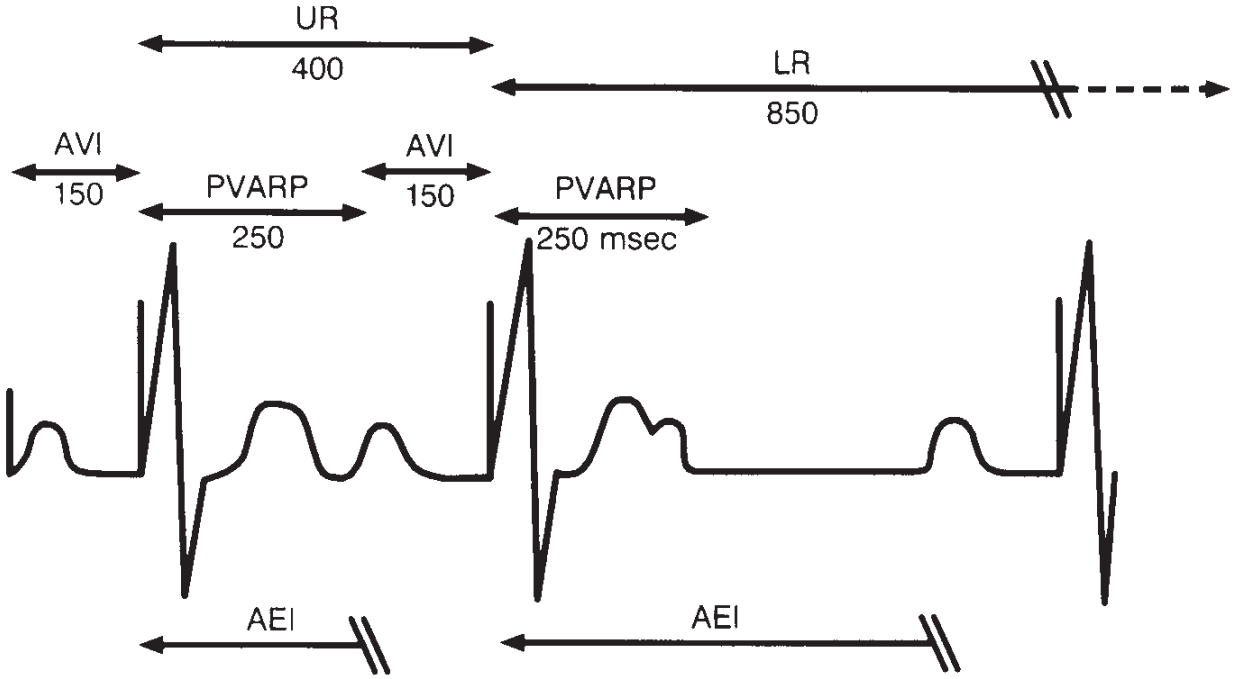


Figure 6.15. In the DDD pacing mode, the upper rate (UR) is limited by the AVI and the PVARP. In this example, the AVI is 150 milliseconds and the PVARP is 250 milliseconds, for a TARP of 400 milliseconds (this is equal to 150 ppm). As shown, after the first paced ventricular complex, a P wave occurs just after the completion of the PVARP. This P wave is sensed, initiates the AVI, and is followed by another paced ventricular complex. The subsequent P wave occurs within the PVARP and is therefore not sensed. The DDD response is to wait for the next intrinsic P wave to occur, as in this example, or for the AEI to be completed, whereupon AV sequential pacing occurs.

282

[Cardiac Pacing and ICDs, Kenneth A Ellenbogen and Mark A Wood]

Related work etc

References

[1] S.S. Barold, R.X. Stroobandt, and A.F. Sinnaeve. *Cardiac Pacemakers and Resynchronization Step-by-Step*. Wiley Blackwell, 2nd edition, 2010.

[2] K.A. Ellenbogen and M.A. Wood. *Cardiac Pacing and ICDs*. Blackwell Science, 3rd edition, 2002.

[3] Z. Jiang, M. Pajic, and R. Mangharam. Cyber-physical modeling of implantable cardiac medical devices. *Proceedings of the IEEE*, 100(1):122–137, 2012.

[4] H.D. Macedo, P.G. Larsen, and J. Fitzgerald. Incremental development of a distributed real-time model of a cardiac pacing system using vdm. In *FM2008*, volume 5014 of *LNCS*. Springer, May 2008.

[5] D. Méry and N.K. Singh. Functional Behavior of a Cardiac Pacing System. *International Journal of Discrete Event Control Systems (IJDECS)*, December 2010.

[6] D. Méry and N.K. Singh. Trustable formal specification for software certification. In Tiziana Margaria and Bernhard Steffen, editors, *ISoLA (2)*, volume 6416 of *Lecture Notes in Computer Science*, pages 312–326. Springer, 2010.

[7] Boston Scientific. Pacemaker system specification. Technical report, 2007. http://sqrl.mcmaster.ca/pacemaker_spec.htm.

Motivation: related work

- A number of teams have modelled the pacemaker, some down to code/ install on test devices
- Not much evidence of refinement
- Not much evidence of abstraction; cue taken from concrete style of requirement doc ?
- No evidence of systematic reuse
- At least 2 teams(Méry/Singh, Jiang et al) have modelled the heart (SUC) in conjunction with the pacemaker (CTL)
  - Méry/Singh have used Event-B, refinement and proof (but in a concrete style)
  - Jiang et al have used hybrid models and simulation

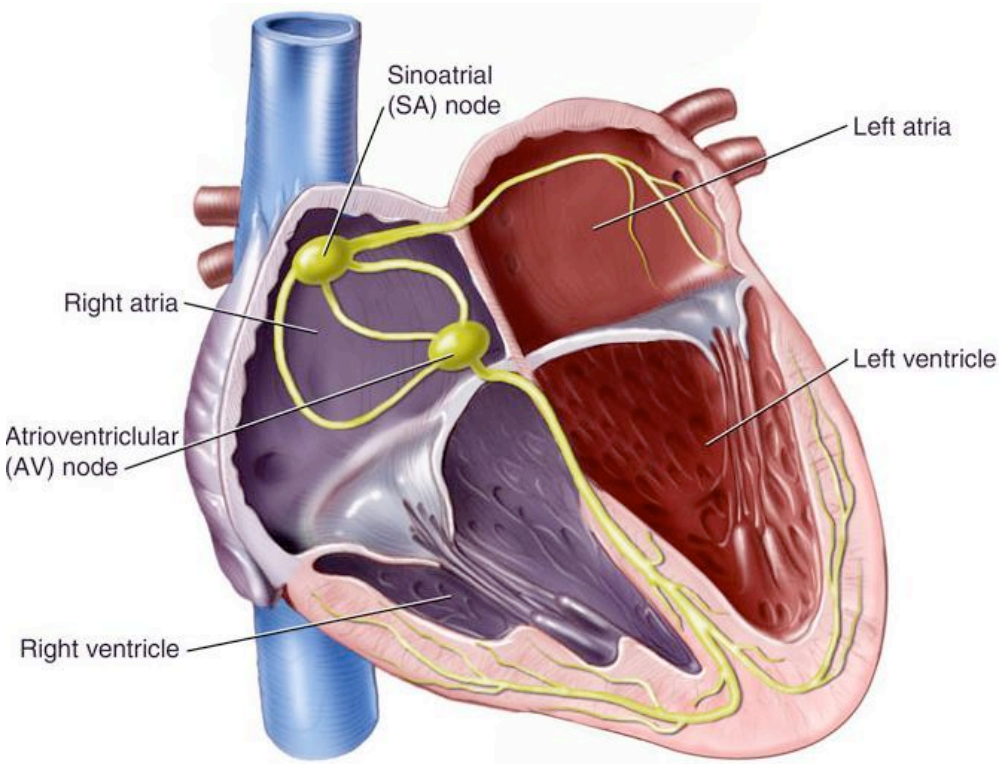
Motivation: ours

- Opportunities for systematic reuse (23 operational modes with common behaviours)
- In search of more explicit repeatable modelling methodology
- State diagram modelling
  - gives elegant process models of requirements
  - which map simply to guide modelling and refinement in Event-B
  - gives a syntactic vehicle for structured reuse
- Exploration of synchronisation modelling
- Investigation of modelling styles for time – starting with Cansell/Mery

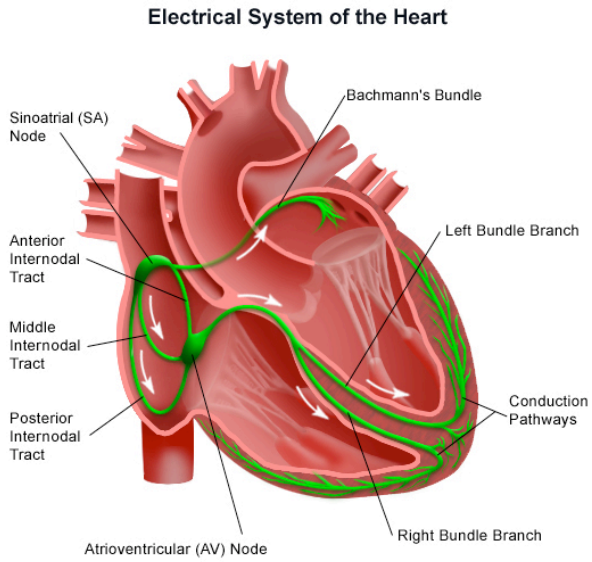
To date we have:

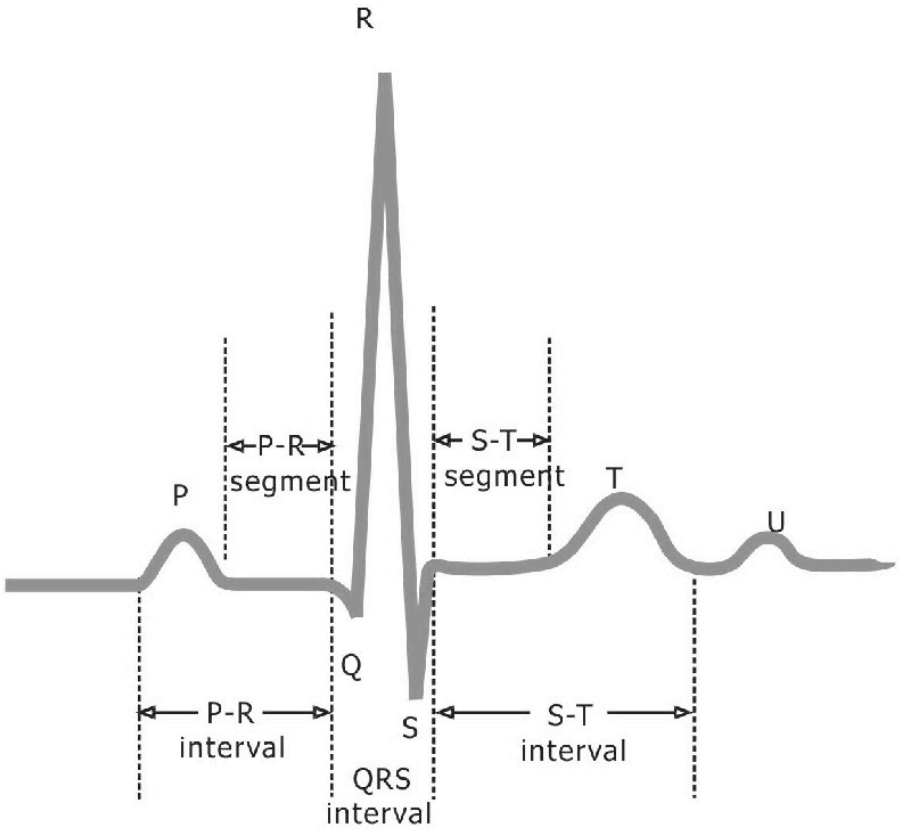
- Modelled 5 modes
  - initially in CSP, then in a *requirements-domain* state diagram notation
  - interpreted these in the *solution domain* of Event-B
  - then refined down to introduction of explicit time intervals
  - Model/verify(proof, model-check)/remodel cycle
- Concerns:
  - no (more) requirements analysis (than anyone else)

Heart System

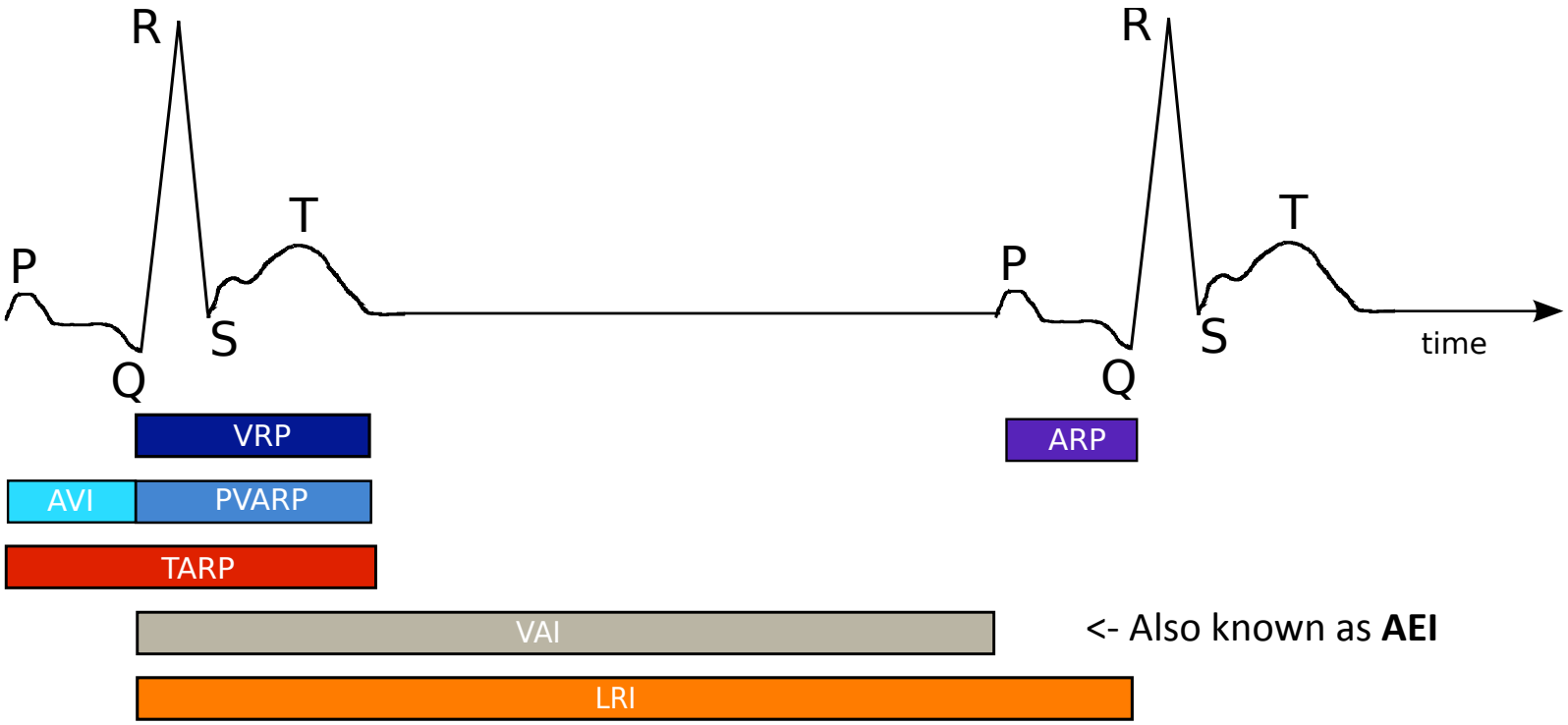


Electrical Signal of the Heart





Timing Cycles



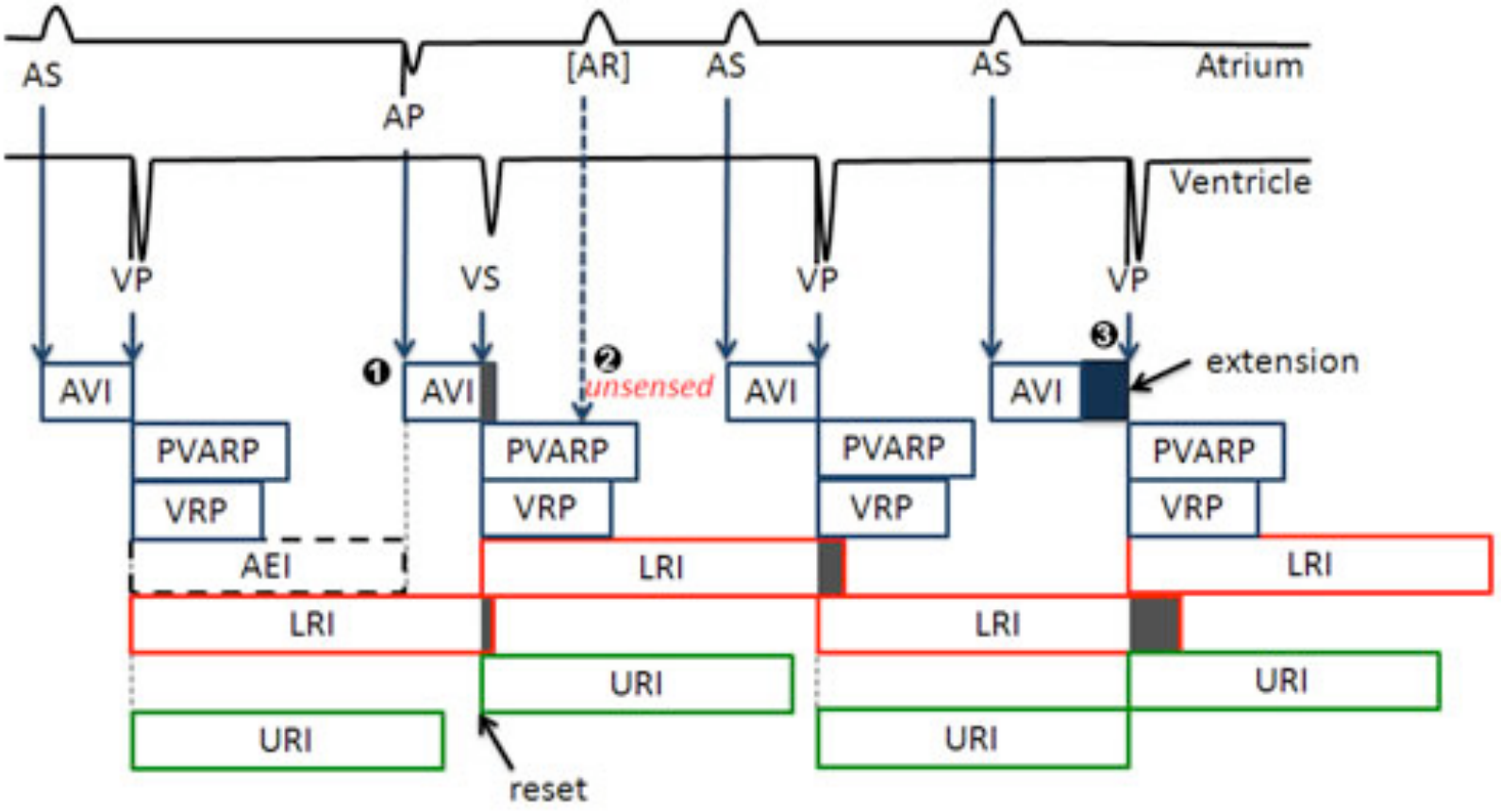
Bradycardia Operating Modes

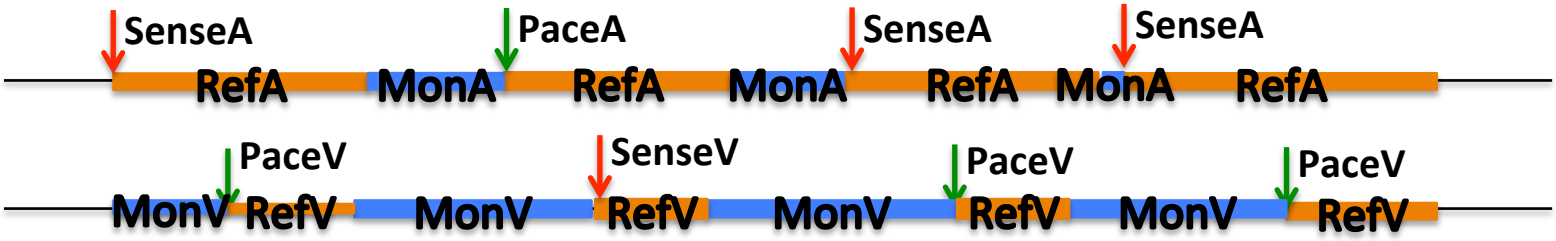
Operating Modes : NASPE/BPEG Generic Code

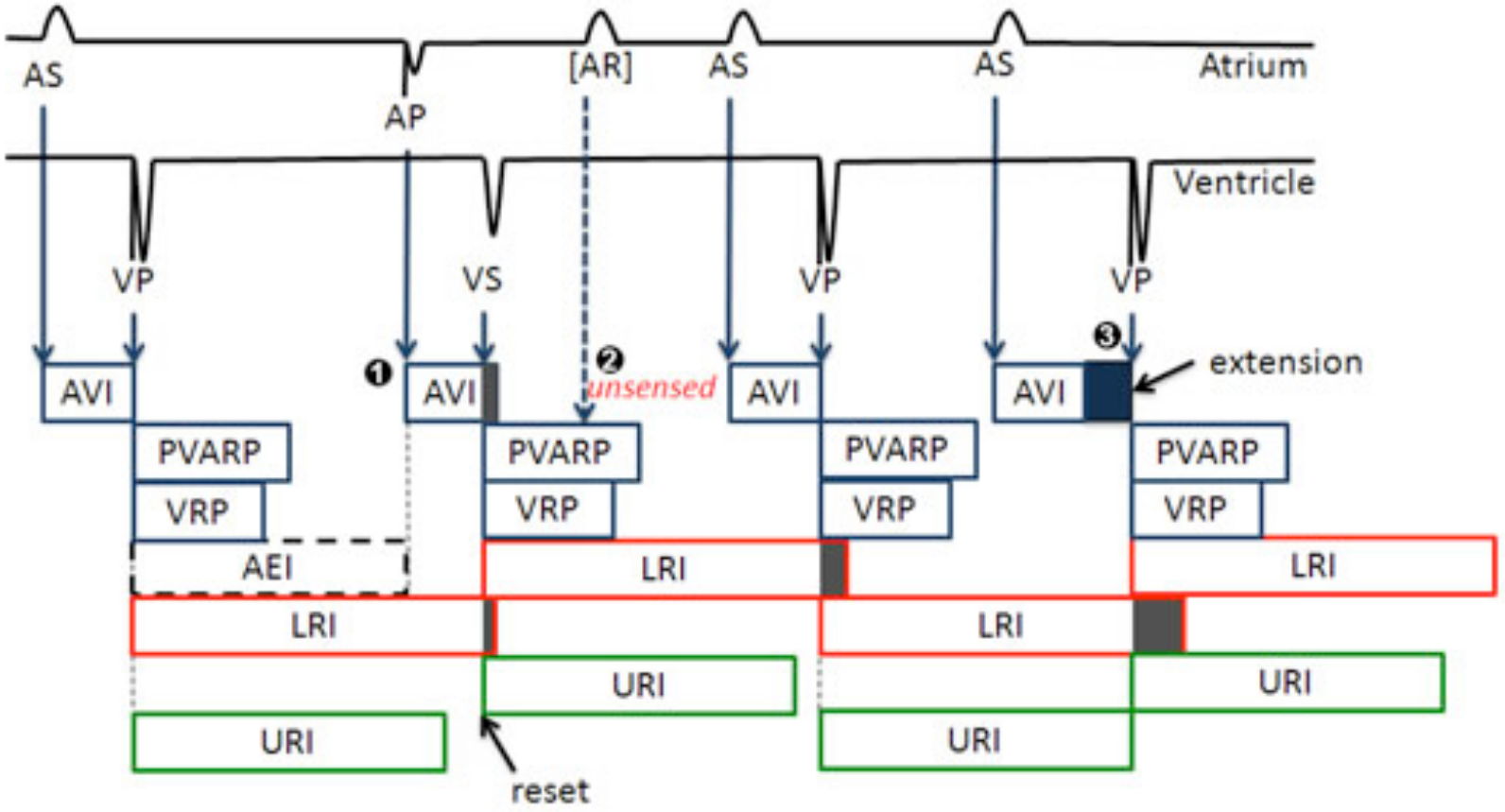
Category	**Chambers Paced**	**Chambers Sensed**	**Response to Sensing**	**Rate Modulation**
Letters	**O**-None **A**-Atrium **V**-Ventricle **D**-Dual(A+V)	**O**-None **A**-Atrium **V**-Ventricle **D**-Dual(A+V)	**O**-None **T**-Triggered **I**-Inhibited **D**-Dual(T+I)	**R**-Rate Modulation

Examples: AOO, VOO, AAI, AAT, VVI, VVT, AATR,VVTR, AOOR etc. . .

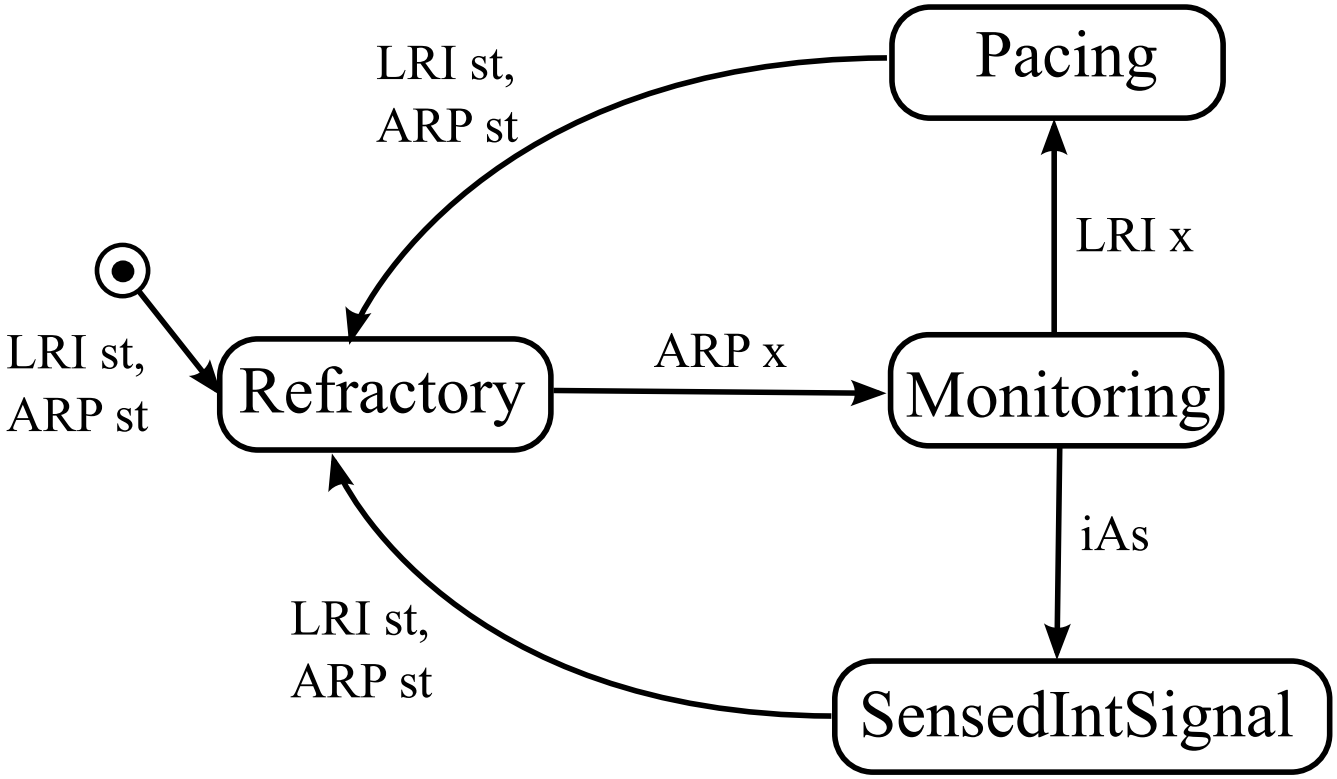
Behaviour: timing-driven sensing and pacing(DDD)



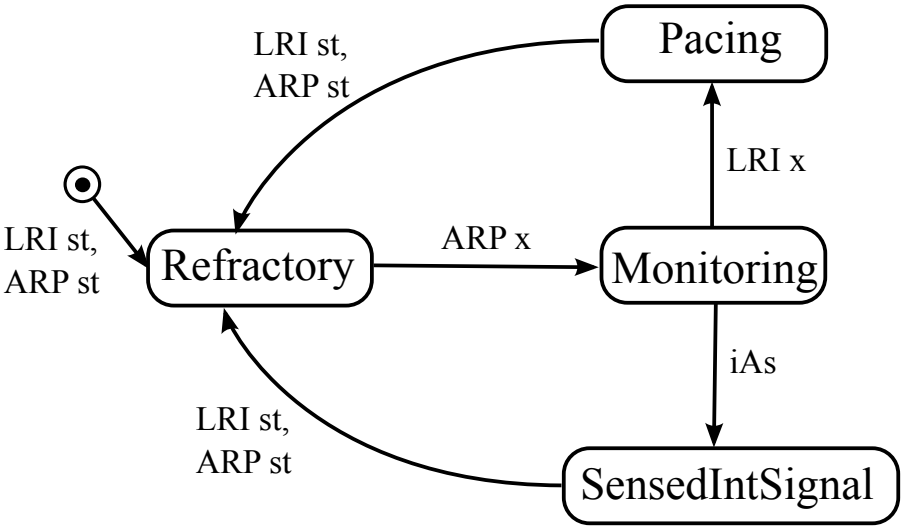

SenseA
PaceA
SenseA
SenseA
RefA
MonA
RefA
MonA
RefA
MonA
RefA
PaceV
SenseV
PaceV
PaceV
MonV
RefV
MonV
RefV
MonV
RefV
MonV
RefV


AS
AP
[AR]
AS
AS
Atrium
Ventricle
VP
VS
VP
VP
❶
❷
unsensed
❸
extension
AVI
AVI
AVI
AVI
PVARP
VRP
PVARP
VRP
PVARP
VRP
PVARP
VRP
AEI
LRI
LRI
LRI
LRI
URI
URI
URI
URI
reset

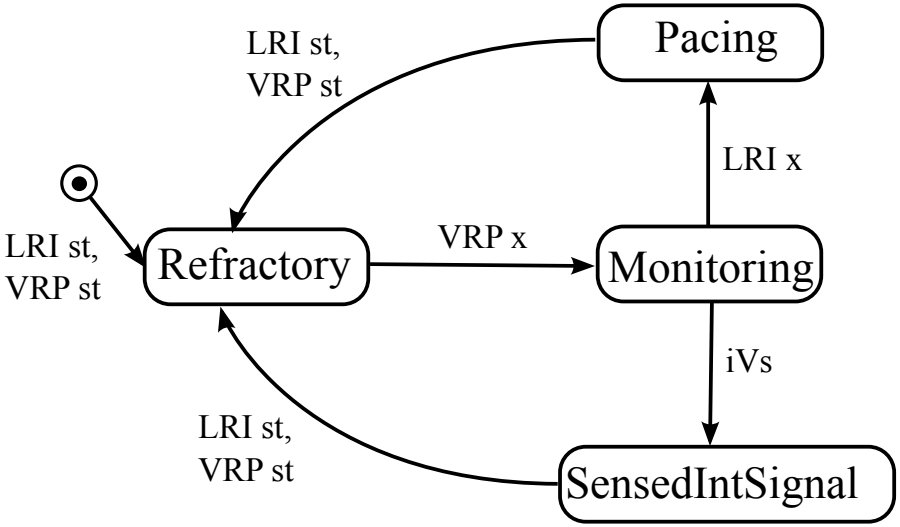
Requirements domain model – AAI



Requirements domain models – AAI, VVI



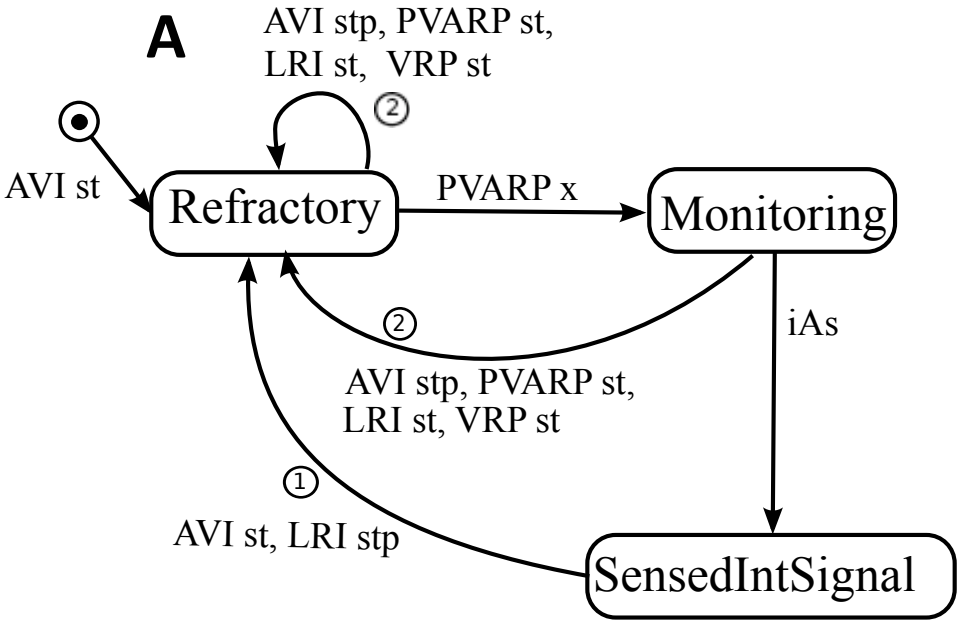
AAI

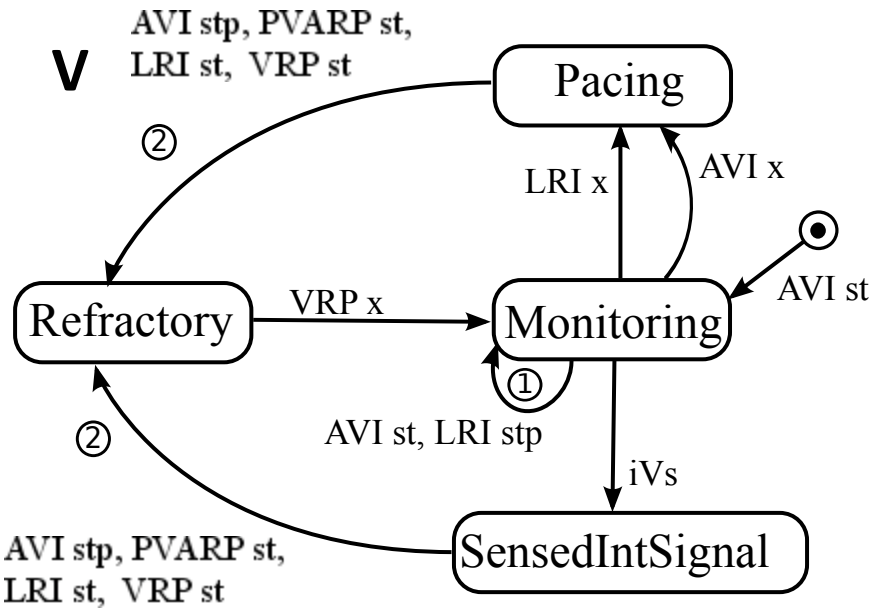


VVI

NB - reuse: AAI -> VVI is simple refactoring by renaming

Requirements domain model - VDD



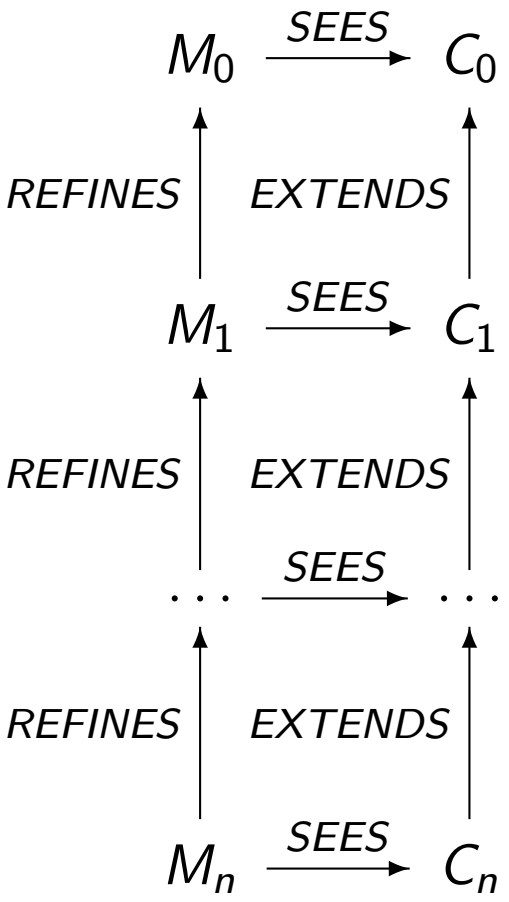


NB: Synchronisation scheme between channels

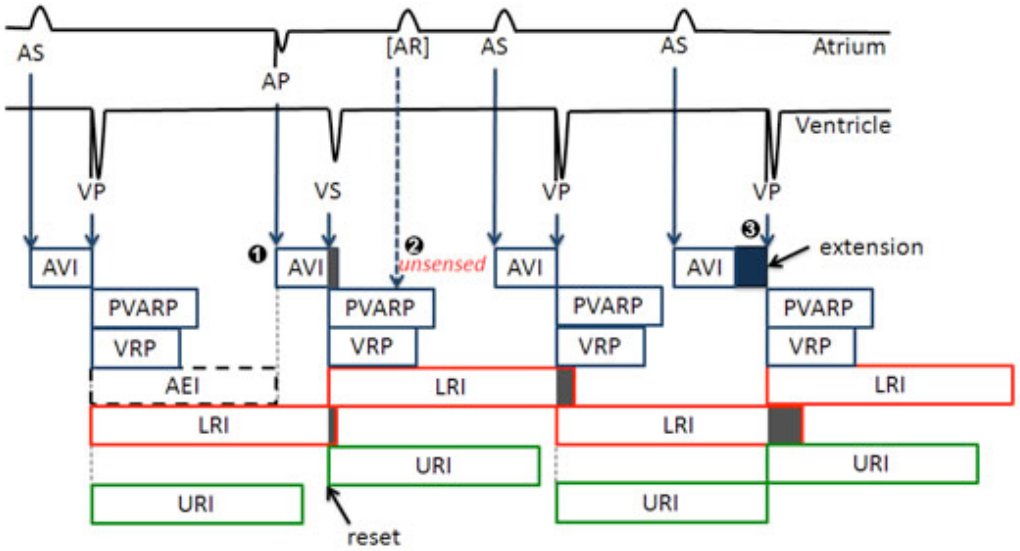
NB – reuse: Each channel is an extended subgraph of AAI modulo renaming

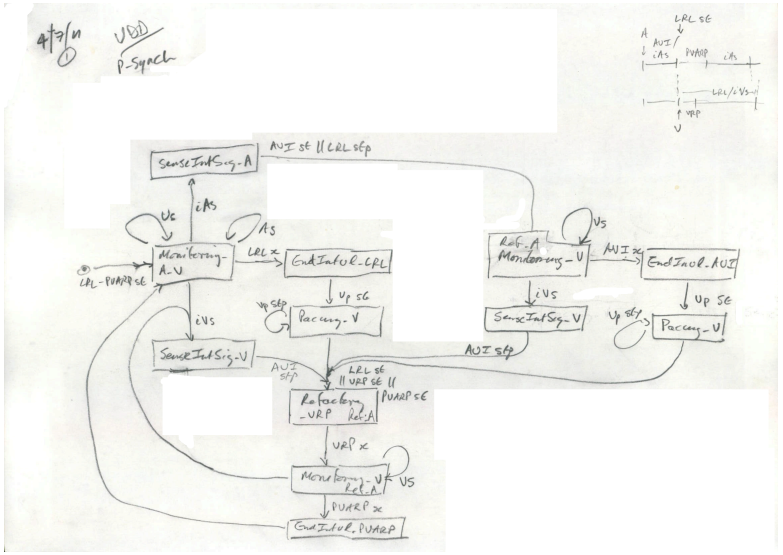
Event-B Overview

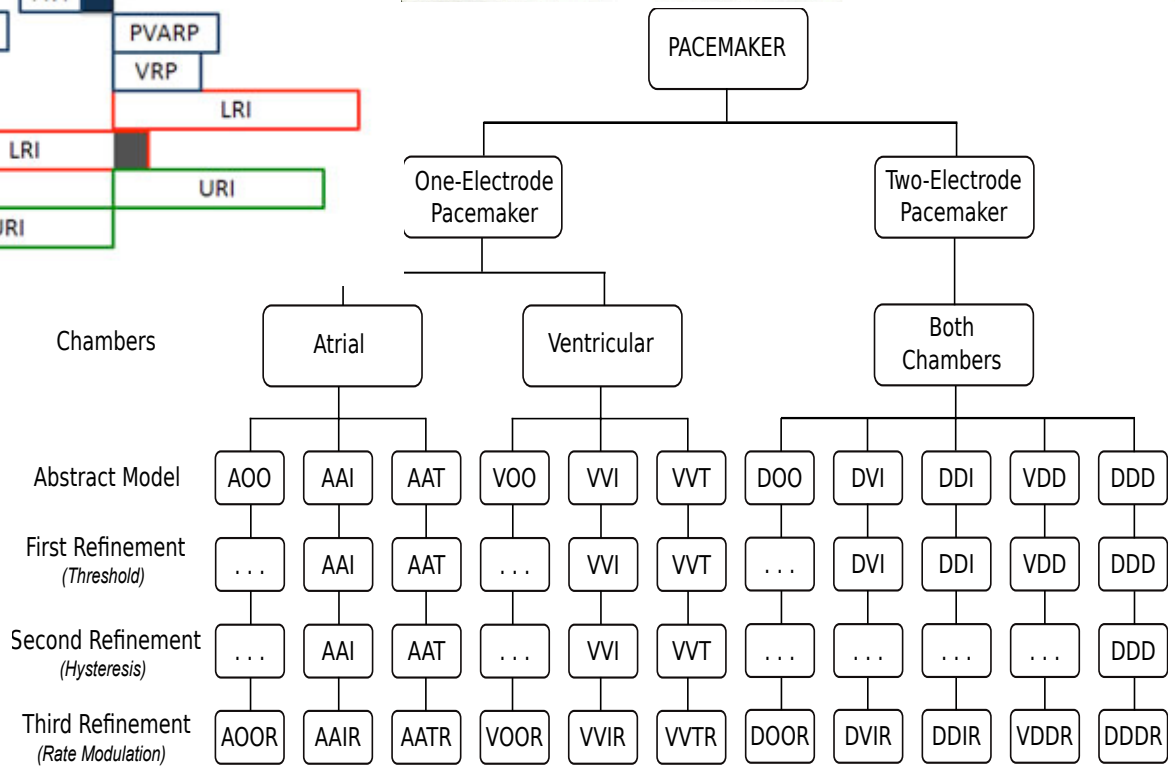
- Two main elements: Contexts & Machines
- Context: static model data and its logical properties
- Machine: dynamic model variables (state), invariants on variables, state-updating events (behaviour)
- Refinement: formal design transformation – layer in requirements, add architectural/ implementation structure, (algorithmic) behaviour



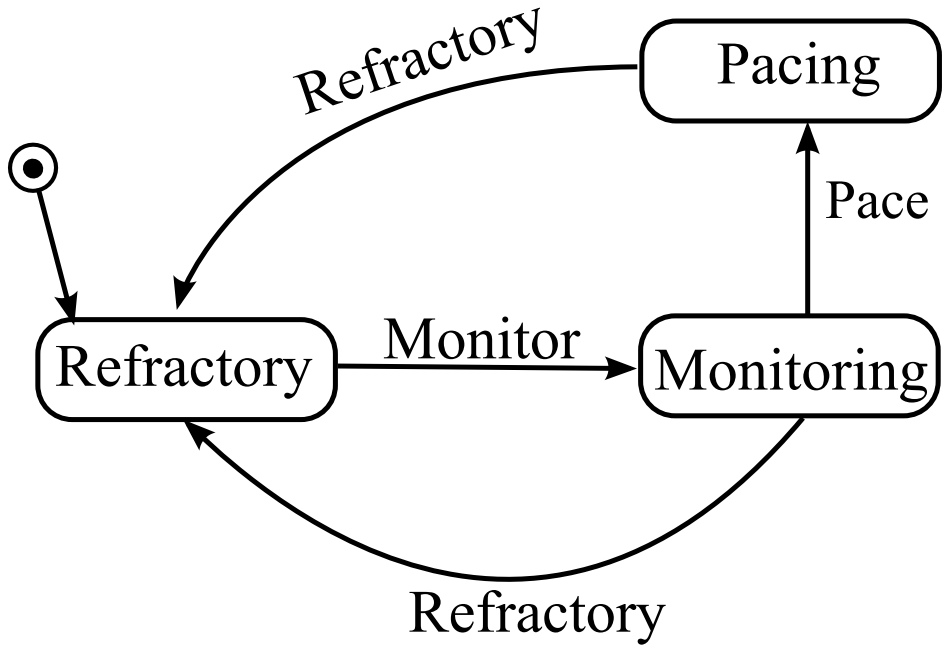
Handling Complexity



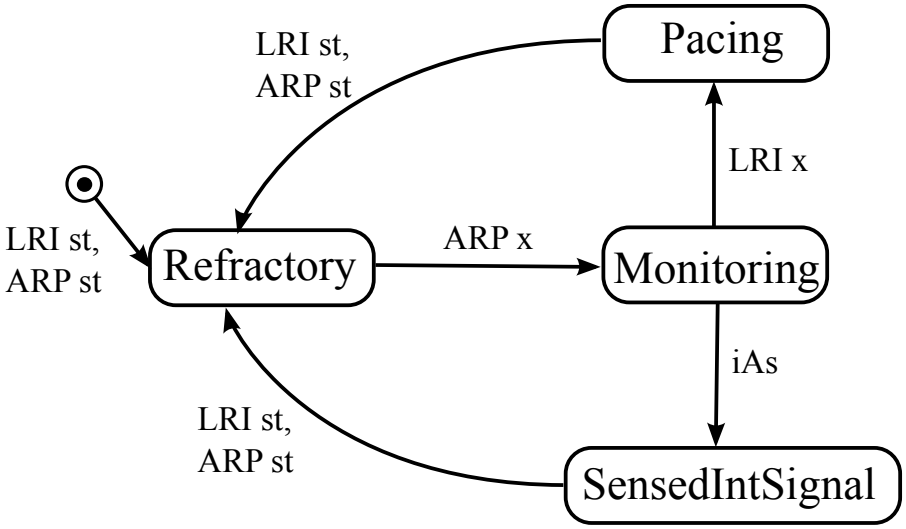


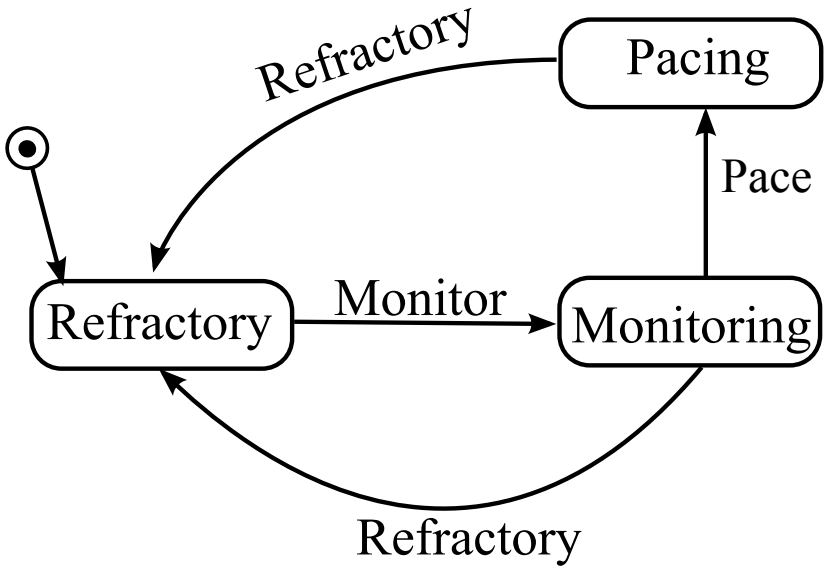


Solution domain AAI : Abstract state machine

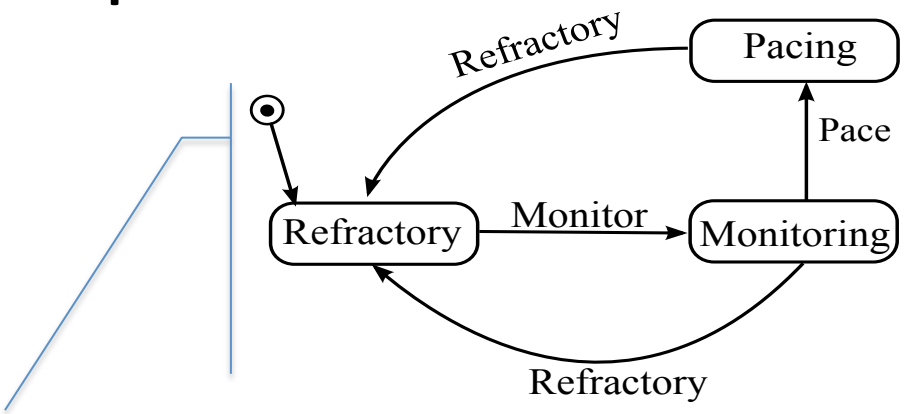


Requirements to solution domain AAI : Abstract state machine





AAI: Event-B Abstract Spec.



INVARIANTS

inv1 : $state \in \mathbb{P}(STATE)$

Initialisation

begin

act1 : $state := \{Refractory\}$

end

Event *Refractory* $\widehat{=}$

when

grd1 : $state = Pacing$

$\lor\ state = Monitoring$

then

act1 : $state := \{Refractory\}$

end

Event *Monitor* $\widehat{=}$

when

grd1 : $state = \{Refractory\}$

then

act1 : $state := Monitoring$

end

Event *Pace* $\widehat{=}$

when

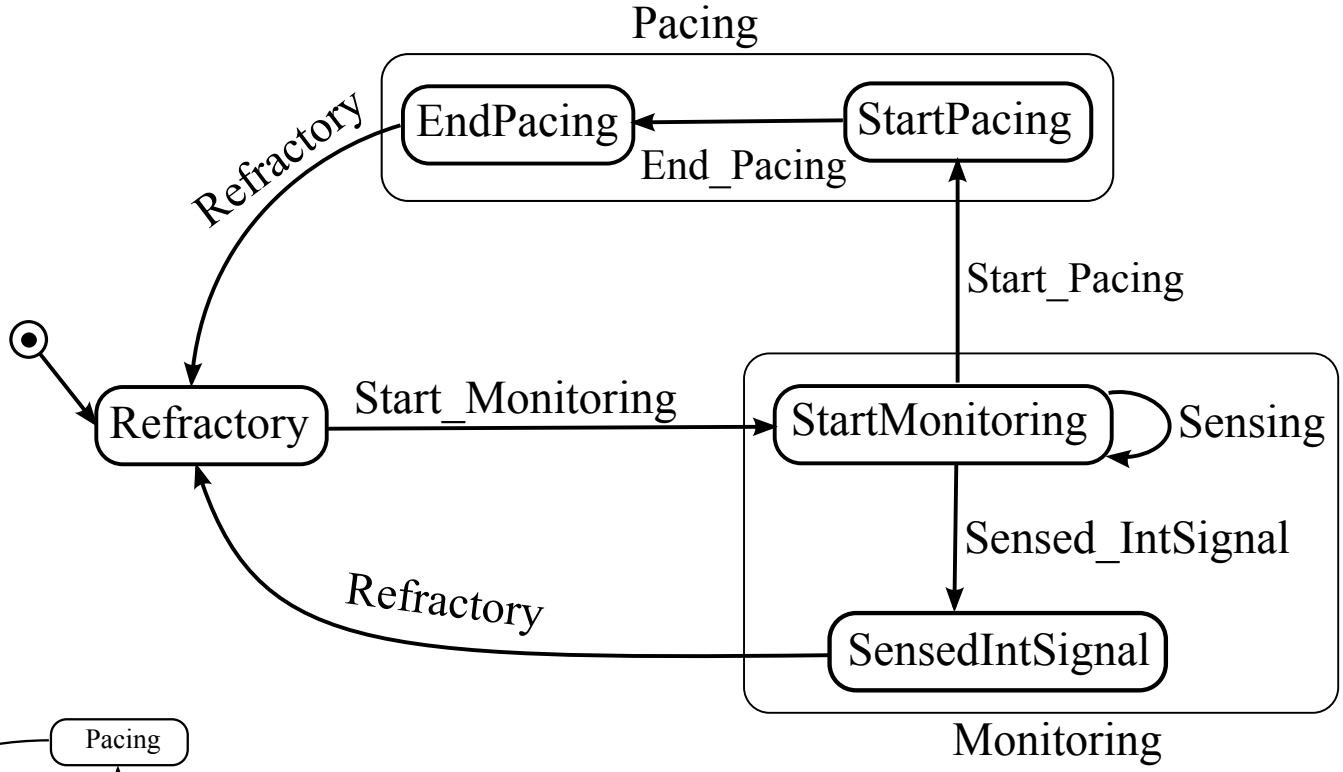
grd1 : $state = Monitoring$

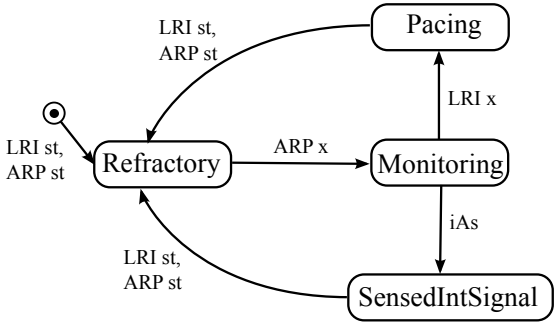
then

act1 : $state := Pacing$

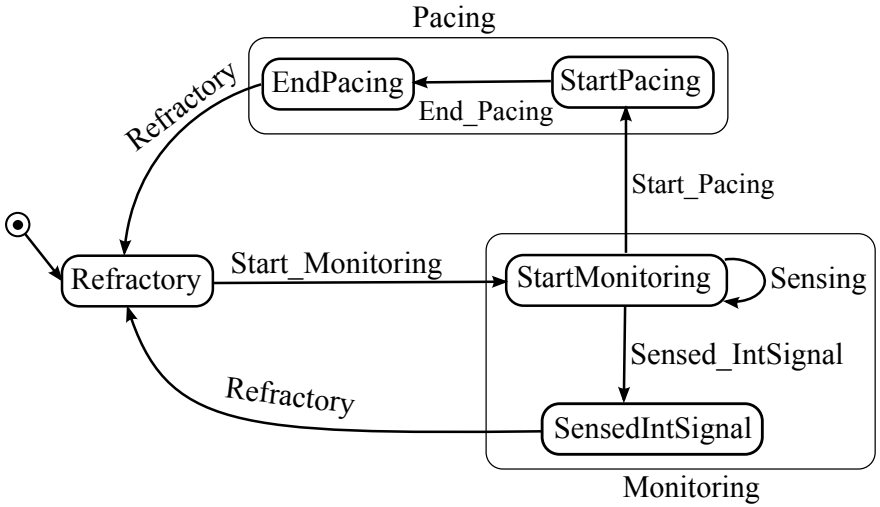
end

AAI – State machine rep. of the first refinement





AAI - First refinement



INVARIANTS

inv01 : $state1 \in STATE$

inv02 : $state1 \in state$

inv08 : $intrinsic\_signal \in BOOL$

inv09 : $pace\_signal \in BOOL$

Initialisation

begin

act1 : $state1 := Refractory$

act02 : $intrinsic\_signal := FALSE$

act03 : $pace\_signal := FALSE$

end

Event *Refractory* $\widehat{=}$
refines *Refractory*
when

grd01 : $state1 = EndPacing \lor state1 = SensedIntSignal$

then

act01 : $state1 := Refractory$

end

Event *Start_Monitoring* $\widehat{=}$
refines *Monitor*
when

grd01 : $state1 = Refractory$

then

act01 : $state1 := StartMonitoring$

act02 : $intrinsic\_signal := FALSE$

end

Event *Sensing* $\widehat{=}$
when

grd01 : $state1 = StartMonitoring$

grd02 : $intrinsic\_signal = FALSE$

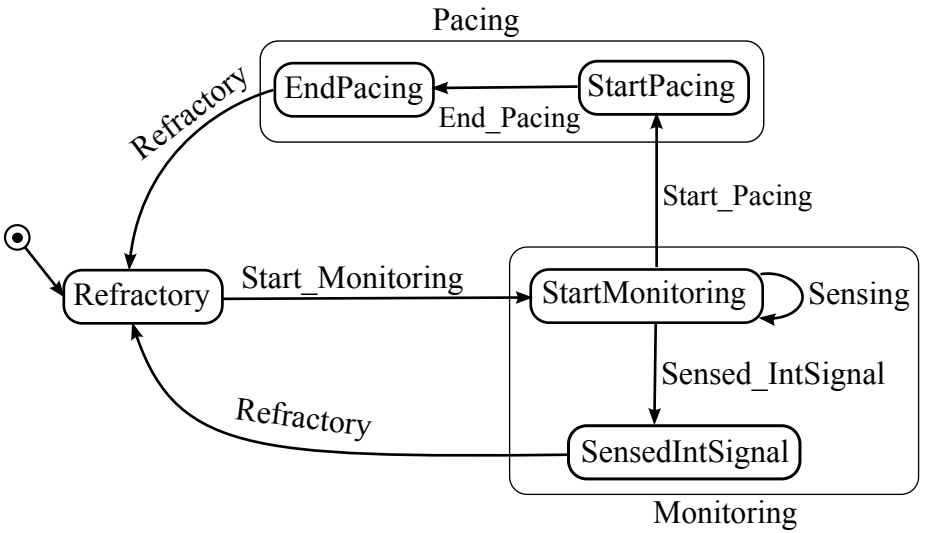
then

act01 : $intrinsic\_signal :\in BOOL$

end

AAI - First refinement(cont.)

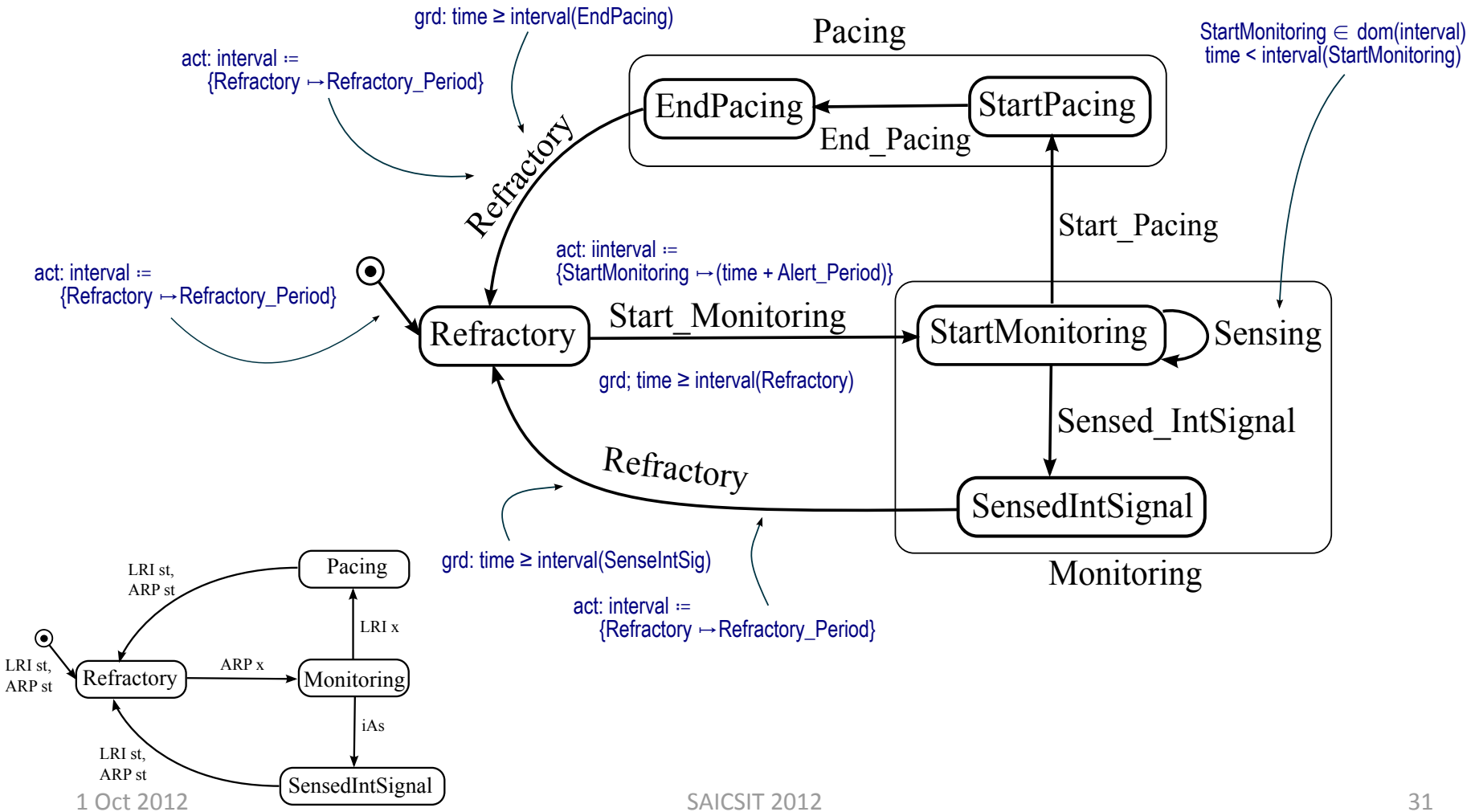
Event *Sensed_IntSignal* $\widehat{=}$
when
grd01 : $state1 = StartMonitoring$
grd02 : $intrinsic\_signal = TRUE$
then
act01 : $state1 := SensedIntSignal$
end



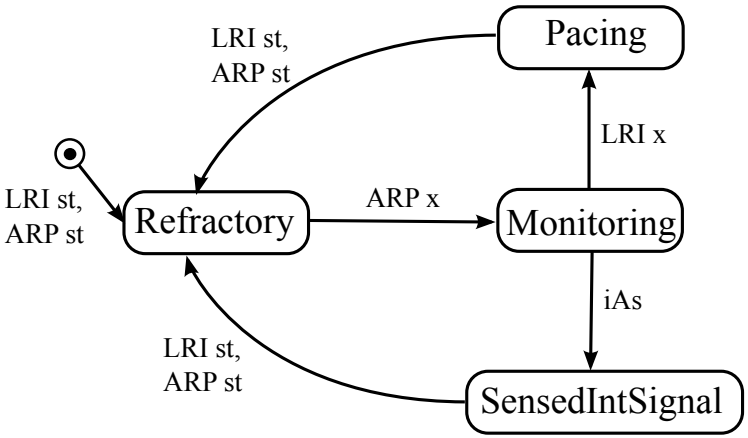
Event *Start_Pacing* $\widehat{=}$
refines *Pace*
when
grd01 : $state1 = StartMonitoring$
grd02 : $intrinsic\_signal = FALSE$
grd03 : $pace\_signal = FALSE$
then
act01 : $state1 := StartPacing$
act02 : $pace\_signal := TRUE$
end

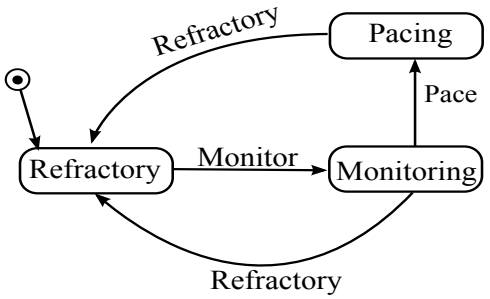
Event *End_Pacing* $\widehat{=}$
when
grd01 : $state1 = StartPacing$
grd02 : $pace\_signal = TRUE$
then
act01 : $pace\_signal := FALSE$
act02 : $state1 := EndPacing$
end

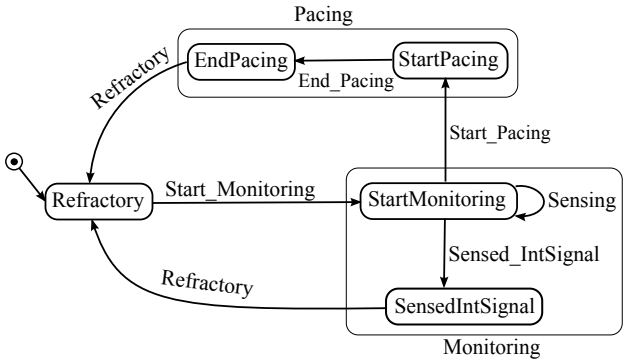
AAI – State machine rep. of the timing refinement

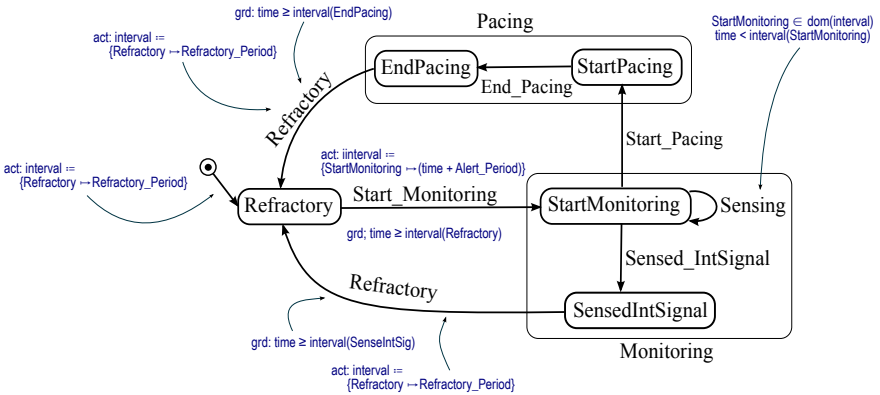


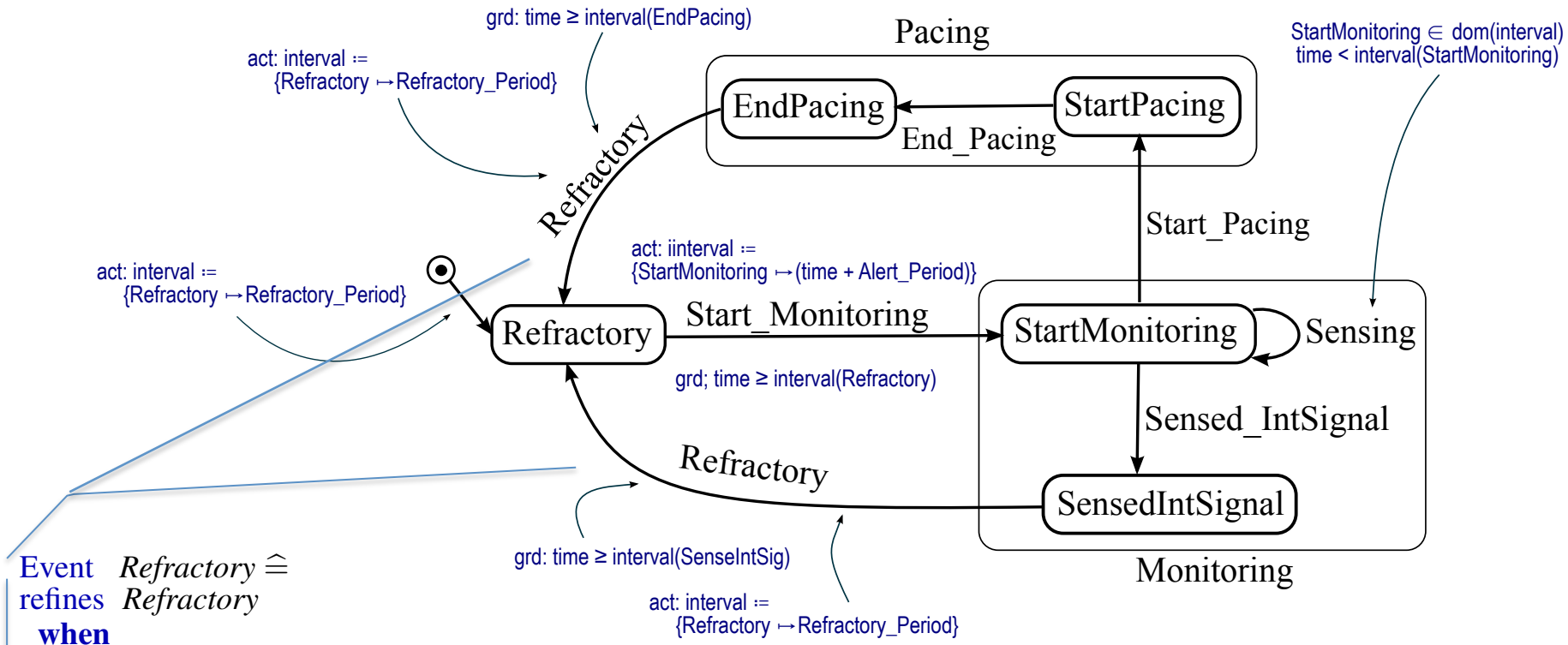
Requirements -> solution domain mapping











Event *Refractory* $\widehat{=}$
refines *Refractory*
when
grd01 : $(SensedIntSignal \in dom(interval)$
$\wedge\ time \geq interval(SensedIntSignal))$
$\vee\ (EndPacing \in dom(interval)$
$\wedge\ time \geq interval(EndPacing))$
then
act01 : $interval := \{Refractory \mapsto$
$(time + Refractory\_Period)\}$
end

Refined

Event *Refractory* $\widehat{=}$
refines *Refractory*
when
grd01 : $state1 = EndPacing$
$\vee\ state1 = SensedIntSignal$
then
act01 : $state1 := Refractory$
end

Abstract

AAI: Second refinement – Introducing Timing properties

INVARIANTS
- inv10 : $time \in \mathbb{N}$
- inv11 : $interval \in STATE \nrightarrow \mathbb{N}$
- inv12 : $dom(interval) = \{state1\}$

Initialisation
begin
- act02 : $intrinsic\_signal := FALSE$
- act03 : $pace\_signal := FALSE$
- act04 : $time := 0$
- act05 : $interval := \{Refractory \mapsto Refractory\_Period\}$

end

Event *Refractory* $\widehat{=}$
refines *Refractory*
when
- grd01 : $(SensedIntSignal \in dom(interval) \land time \geq interval(SensedIntSignal)) \lor (EndPacing \in dom(interval) \land time \geq interval(EndPacing))$

then
- act01 : $interval := \{Refractory \mapsto (time + Refractory\_Period)\}$

end

Event *Start_Monitoring* $\widehat{=}$
refines *Start_Monitoring*
when
- grd01 : $Refractory \in dom(interval)$
- grd02 : $time \geq interval(Refractory)$

then
- act01 : $interval := \{StartMonitoring \mapsto (time + Alert\_Period)\}$
- act02 : $intrinsic\_signal := FALSE$

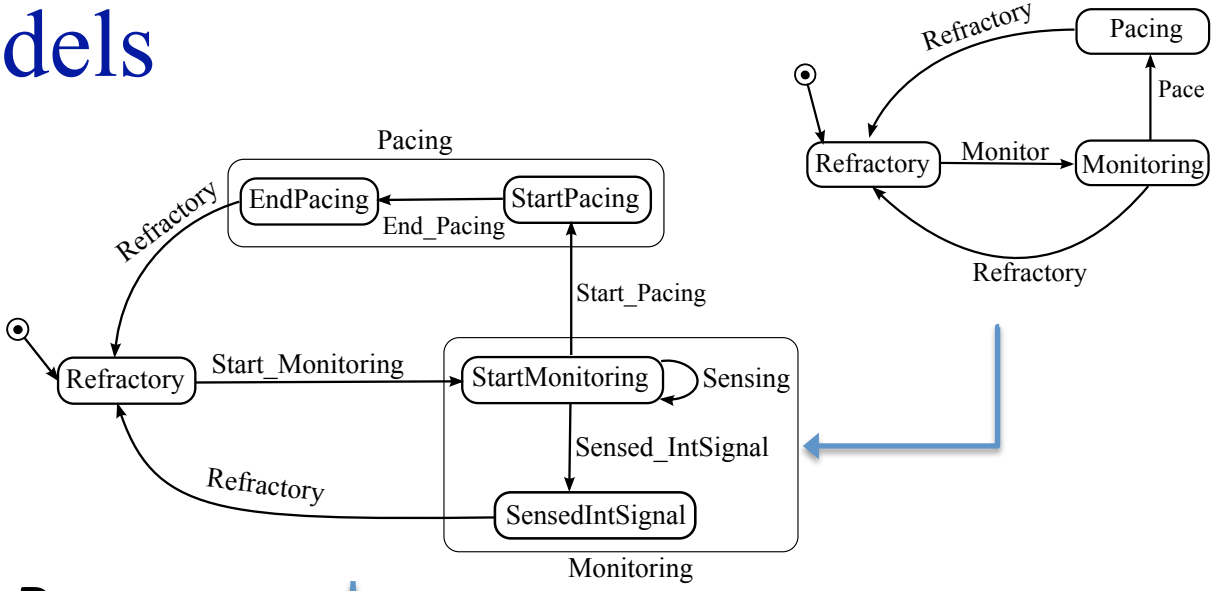
end

Event *Clock* $\widehat{=}$
any
new_time
where
- grd01 : $interval \neq \varnothing$
- grd02 : $new\_time \leq min(ran(interval))$
- grd03 : $new\_time > time$

then
- act01 : $time := new\_time$

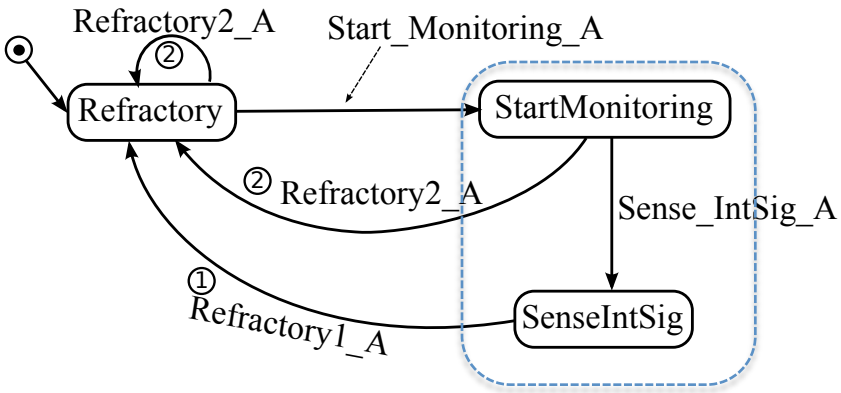
end

Generic Models

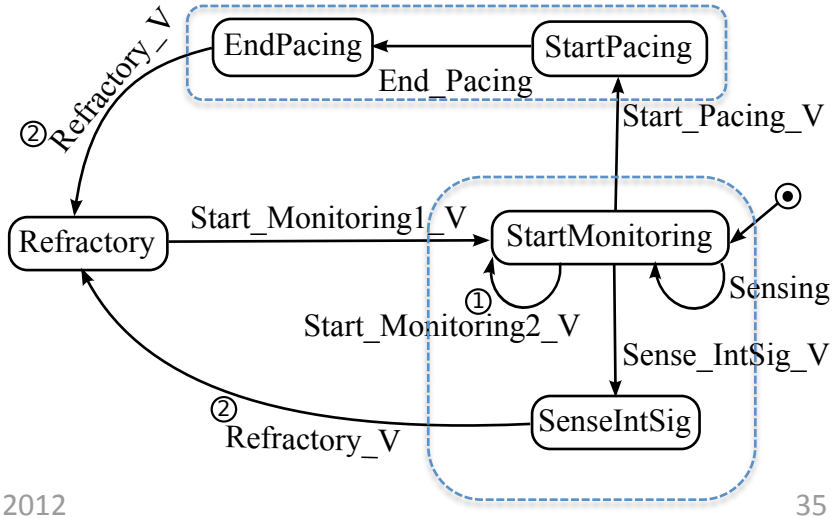


VDD Mode - Reuse

A Channel

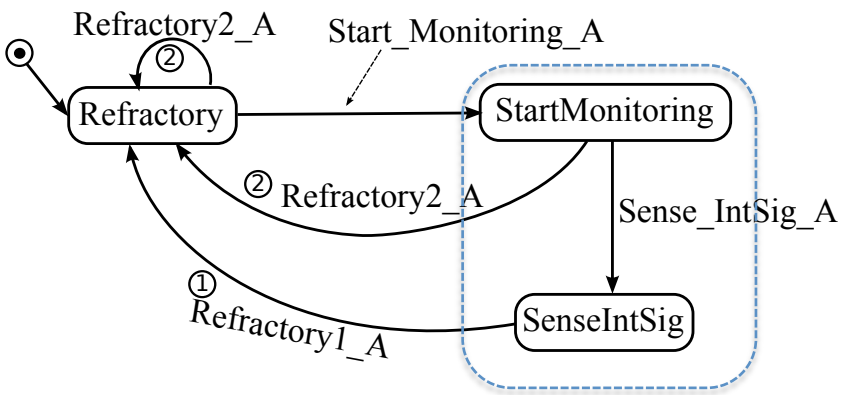


V Channel

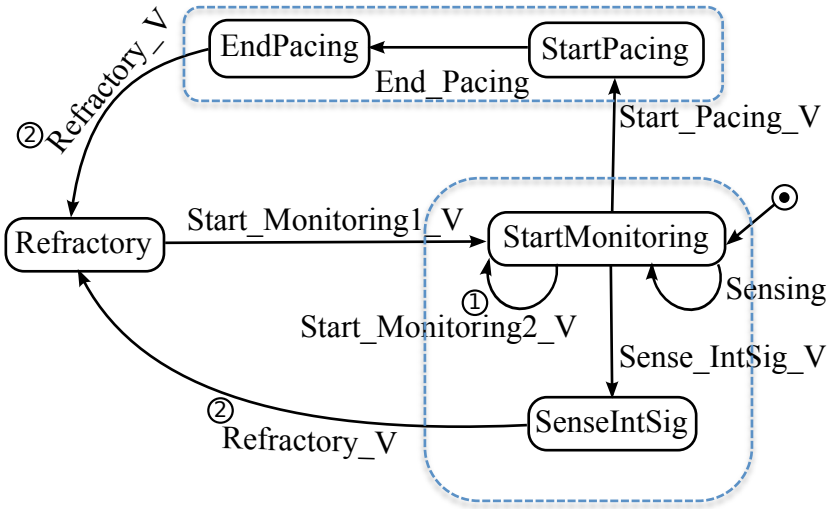


VDD Mode – synchronisation – refinement 1

A Channel



V Channel



Event *Refractory2_A* $\widehat{=}$
refines *Refractory_A*
when
grd01 : $v2a\_signal = TRUE$
grd02 : $state1\_a = Refractory$
$\vee\ state1\_a = StartMonitoring$
then
act01 : $state1\_a := Refractory$
act02 : $v2a\_signal := FALSE$
end

Event *SensedIntSignal_V* $\widehat{=}$
when
grd01 : $state1\_v = StartMonitoring$
grd02 : $intrinsic\_signal\_v = TRUE$
then
act01 : $state1\_v := SensedIntSignal$
act02 : $v2a\_signal := TRUE$
end

Event *Start_Pacing_V* $\widehat{=}$
refines *Pace_V*
when
grd01 : $state1\_v = StartMonitoring$
grd02 : $intrinsic\_signal\_v = FALSE$
grd03 : $pace\_signal\_v = FALSE$
then
act01 : $state1\_v := StartPacing$
act02 : $v2a\_signal := TRUE$
act03 : $pace\_signal\_v := TRUE$
end

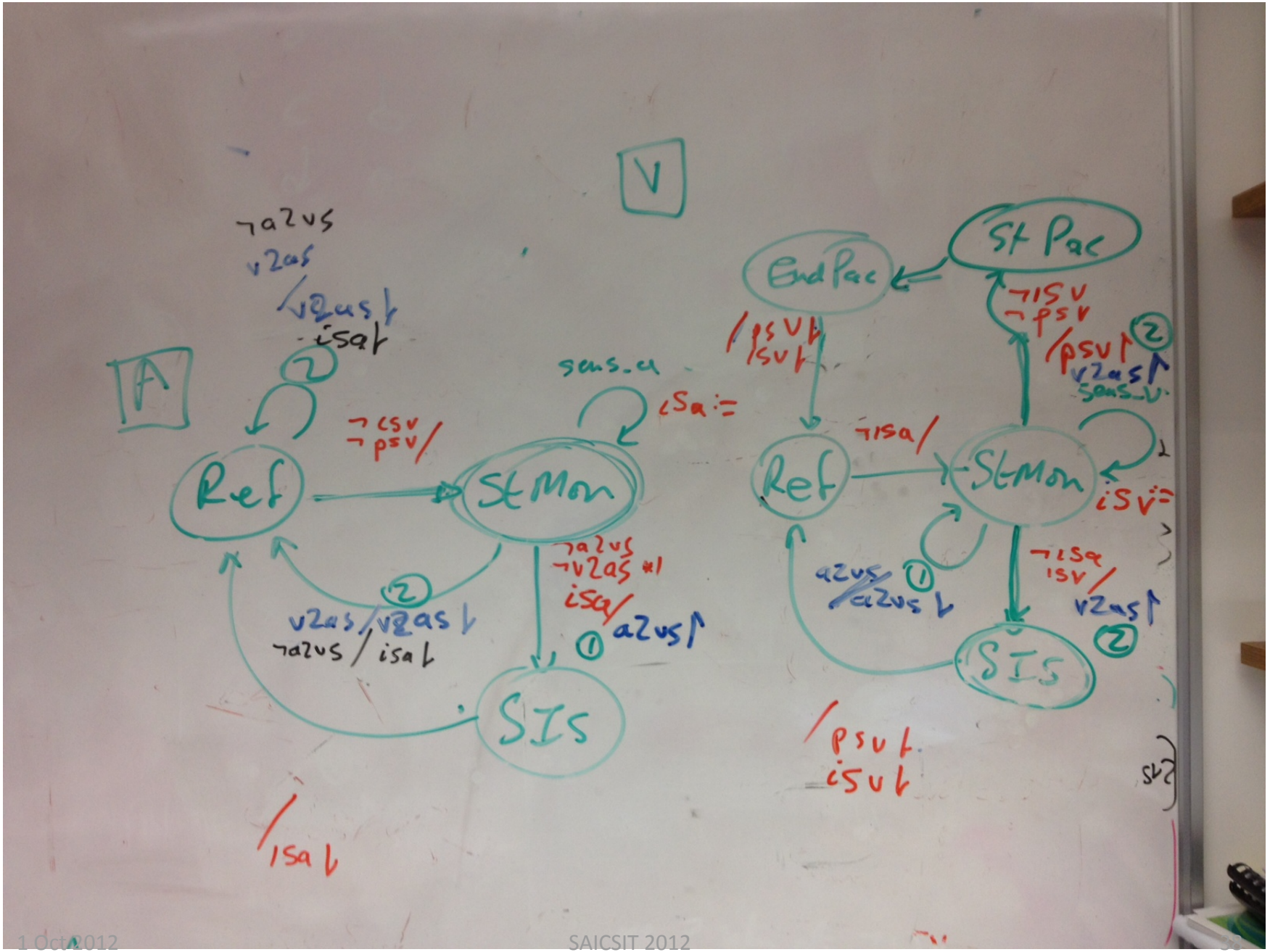
VDD Mode – synchronisation – refinement 1

Requirements elicitation: ProB animation and model-checking of first refinement showed excessive event enablement, i.e. :

- While A int-signal being processed, V pacing and int-signal processing is suspended
- A sensed A int-signal must be (i) reported to V channel for synchronisation/suspension, and (ii) processed in a timely manner. (iii) Completion of processing must be reported to V channel for synchronisation/resumption
- And vice-versa for V int-signal or pace

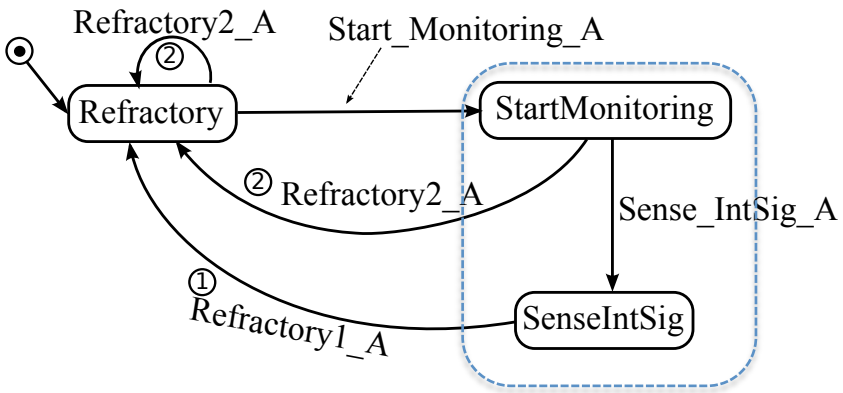
These are requirements on

- signal flags *pace_signal, intrinsic_signal_A, intrinsic_signal_V*
- synch flags *a2vSignal, v2a_signal*

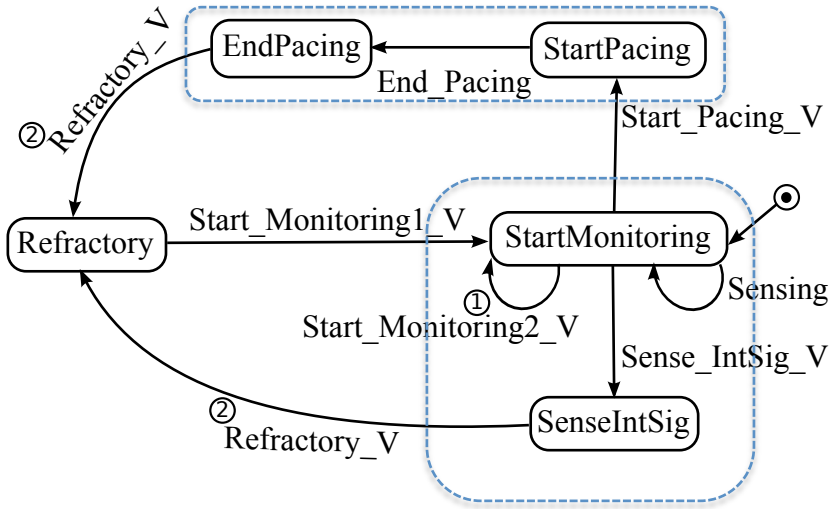

V
A
Ref
StMon
SIs
EndPac
StPac
Ref
StMon
SIs

VDD Mode – synchronisation – refinement 1a

A Channel



V Channel



Statically encode these requirements as safety invariants in 2 refinement steps

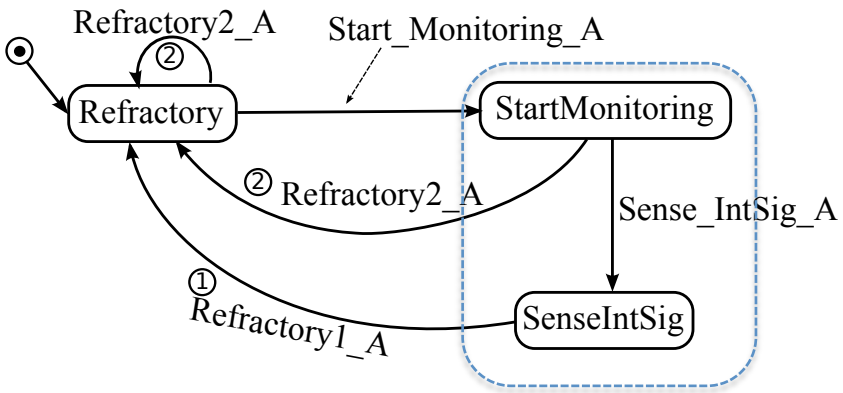
i. Strip out signals

ii. Stop channels from being simultaneously active:

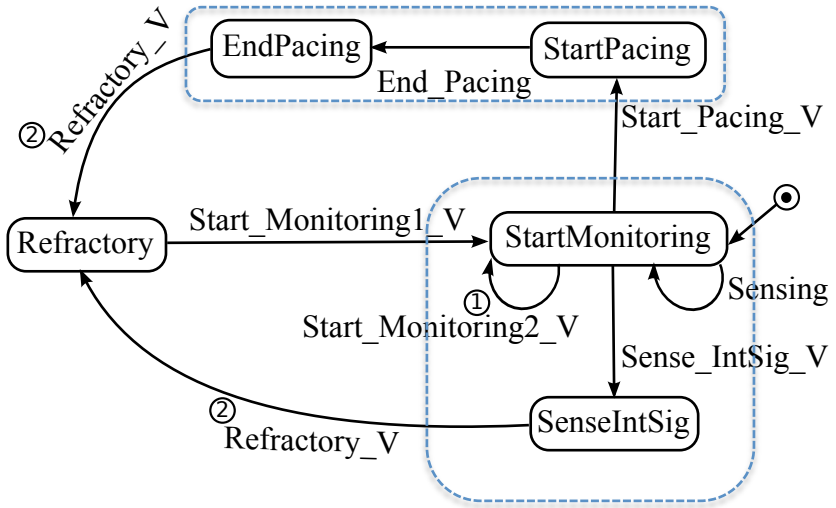
$$\begin{aligned}
&\texttt{inv14} : state1_a = SenseIntSig \Rightarrow state1_v \neq StartPacing \\
&\texttt{inv15} : state1_a = SenseIntSig \Rightarrow state1_v \neq EndPacing \\
&\texttt{inv16} : state1_a = SenseIntSig \Rightarrow state1_v \neq SenseIntSig \\
&\texttt{inv17} : state1_v = StartPacing \Rightarrow state1_a \neq SenseIntSig \\
&\texttt{inv18} : state1_v = EndPacing \Rightarrow state1_a \neq SenseIntSig \\
&\texttt{inv19} : state1_v = SenseIntSig \Rightarrow state1_a \neq SenseIntSig
\end{aligned}$$

VDD Mode – synchronisation – refinement 1b

A Channel



V Channel



i. Replace signals

ii. Replace direct cross-chamber references with signal properties of each state :

$$\texttt{inv51} : state1\_v = SenseIntSig \Rightarrow intrinsic\_signal\_v = TRUE$$
$$\texttt{inv52} : state1\_a = SenseIntSig \Rightarrow intrinsic\_signal\_a = TRUE$$
$$\texttt{inv53} : state1\_v = StartPacing \Rightarrow pace\_signal\_v = TRUE$$
$$\texttt{inv54} : state1\_v = EndPacing \Rightarrow pace\_signal\_v = TRUE$$
$$\texttt{inv55} : state1\_v = StartPacing \Rightarrow intrinsic\_signal\_v = FALSE$$
$$\texttt{inv56} : state1\_v = EndPacing \Rightarrow intrinsic\_signal\_v = FALSE$$
$$\texttt{inv57} : state1\_a = StartMonitoring \wedge state1\_v = SenseIntSig \Rightarrow v2a\_signal = TRUE$$
$$\texttt{inv58} : state1\_a = StartMonitoring \wedge state1\_v = StartPacing \Rightarrow v2a\_signal = TRUE$$
$$\texttt{inv59} : state1\_a = StartMonitoring \wedge state1\_v = EndPacing \Rightarrow v2a\_signal = TRUE$$

VDD Mode – Event-B Rep.

Event $Refractory1\_A \widehat{=}$
refines $Refractory\_A$
when
grd1 : $state1\_a = SensedIntSignal$
then
act1 : $state1\_a := Refractory$
end

Event $Refractory2\_A \widehat{=}$
refines $Refractory\_A$
when
grd01 : $v2a\_signal = TRUE$
grd02 : $state1\_a = Refractory$
$\lor\ state1\_a = StartMonitoring$
then
act01 : $state1\_a := Refractory$
act02 : $v2a\_signal := FALSE$
end

Event $SensedIntSignal\_A \widehat{=}$
when
grd01 : $state1\_a = StartMonitoring$
grd02 : $intrinsic\_signal\_a = TRUE$
then
act01 : $state1\_a := SensedIntSignal$
act02 : $a2v\_signal := TRUE$
end

Event $Start\_Monitoring1\_V \widehat{=}$
refines $Monitor\_V$
when
grd1 : $state1\_v = Refractory$
then
act01 : $state1\_v := StartMonitoring$
act02 : $intrinsic\_signal\_v := FALSE$
end

VDD Mode – Event-B Rep.

Event *SensedIntSignal_V* $\widehat{=}$
when
grd01 : $state1\_v = StartMonitoring$
grd02 : $intrinsic\_signal\_v = TRUE$
then
act01 : $state1\_v := SensedIntSignal$
act02 : $v2a\_signal := TRUE$
end

Event *Start_Pacing_V* $\widehat{=}$
refines *Pace_V*
when
grd01 : $state1\_v = StartMonitoring$
grd02 : $intrinsic\_signal\_v = FALSE$
grd03 : $pace\_signal\_v = FALSE$
then
act01 : $state1\_v := StartPacing$
act02 : $v2a\_signal := TRUE$
act03 : $pace\_signal\_v := TRUE$
end

Event *Start_Monitoring2_V* $\widehat{=}$
refines *Monitor_V*
when
grd01 : $a2v\_signal = TRUE$
grd02 : $state1\_v = StartMonitoring$
then
act01 : $state1\_v := StartMonitoring$
act02 : $a2v\_signal := FALSE$
act03 : $intrinsic\_signal\_v := FALSE$
end

Conclusion

We have:

- modelled AAI, VVI, DVI, VDD, DDD modes
  - in a flat *requirements-domain* state diagram notation
  - mapped this to a refinement chain of state machines in the *solution domain* (of Event-B)
- explored loose synchronisation modelling (for concurrent code)
- sketched a method pattern of dynamic -> static verification loop
- identified statemachine-based reuse patterns
- progressed statemachine design & animation tool support for Event-B
- started investigation of adapted (from Cansell/Mery) modelling style for time

Further work

Continue:

- Synchronisation/ implementation design
- Timing elaboration
- State machine plugin & animation development

- Functional development: hysteresis, rate modulation
- Systematize statemachine-based reuse, develop tool support (building on the feature composition plugin)
- Heart modelling for simulation-based testing of pacemaker
- Utility of process (eg CSP) modelling ?
- Continuous modelling of sensing / pacing mechanics ?

That’s it folks.