

# Discrete Intra-Agent Dynamics: Statecharts & Messaging

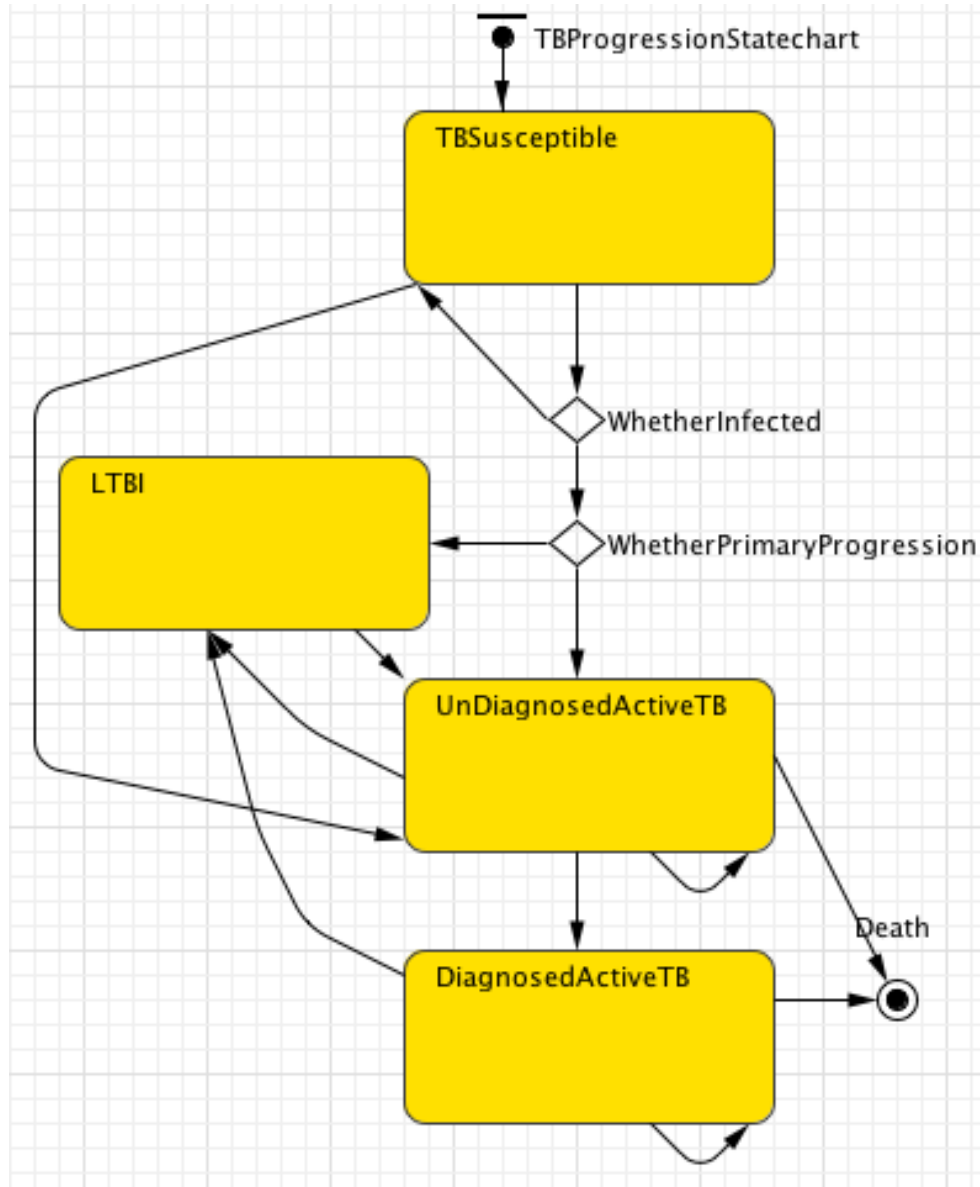
Nathaniel Osgood

February 10, 2011

# Discrete Agent Dynamics

- Frequently we can represent agent behaviour using as transitioning among a set of states in a state chart
- For a given simple statechart, the agent is in exactly one state at a time
- Fixed transitions between states define possible evolution
- The transitions between states occur instantaneously, based on some condition

# Example State Transition Diagram



# Discrete Agent Dynamics: Transitions

- Many transition conditions are possible
  - Timeout: Spending some period of time in the state
  - Fixed rate: Leave state with some fixed change per unit time
    - This is similar to “first order interarrival time”, and is conceptually linked to the operation of first-order delays in stock & flow diagrams
  - Variable rate: If desired, we can change the rate over time – but Anylogic only “notices” changes when eg agent re-enters the state
  - Message received: We can transition when a message (any message or particular type of message) is received
  - Predicate: Only transition when condition becomes true
- These transitions can be conditionally “routed” via branches
  - Conditions can determine to what destination state a particular transition will travel

# Transition Type: Message Triggered

The image shows a statechart editor interface for a TB progression model. The main workspace displays a statechart with the following states and transitions:

- States:** TBSusceptible, LTBI, UnDiagnosedActiveTB, DiagnosedActiveTB, and a final state (represented by a bullseye).
- Transitions:** "WhetherInfected" (diamond) and "WhetherPrimaryProgression" (diamond) are decision points. Transitions include "Aging", "Sex", "Ethnicity", "getDegree", "Death", and "ForcedInitialTBInfection".

The configuration panel for the "ForcedInitialTBInfection" transition is open, showing the following settings:

- Name:** ForcedInitialTBInfection
- Triggered by:** Message
- Message type:** int (selected), boolean, double, String, Other
- Class Name:** (empty)
- Fire transition:** If message equals (selected), Unconditionally, If expression is true (use msg for message)
- Message:** MsgForceInitialTBInfection

The left sidebar shows a project tree with folders for Parameters, Functions, Environments, Embedded Objects, Presentation, Person, Plain Variables, Dynamic Variables, and Statecharts. The bottom status bar indicates "Selection" and "Cursor: X=339, Y=320".

# Transition Type: Fixed Rate

The screenshot displays the AnyLogic Advanced software interface, showing a statechart for TB progression. The statechart is titled "TBProgressionStatechart" and is located within the "Person" environment. The states are represented by yellow rounded rectangles: "TBSusceptible", "LTBI", "UnDiagnosedActiveTB", and "DiagnosedActiveTB". Transitions are represented by arrows, and decision diamonds are used for "WhetherInfected" and "WhetherPrimaryProgression".

The transition "NaturalTBRecovery" is highlighted in the console, showing its configuration:

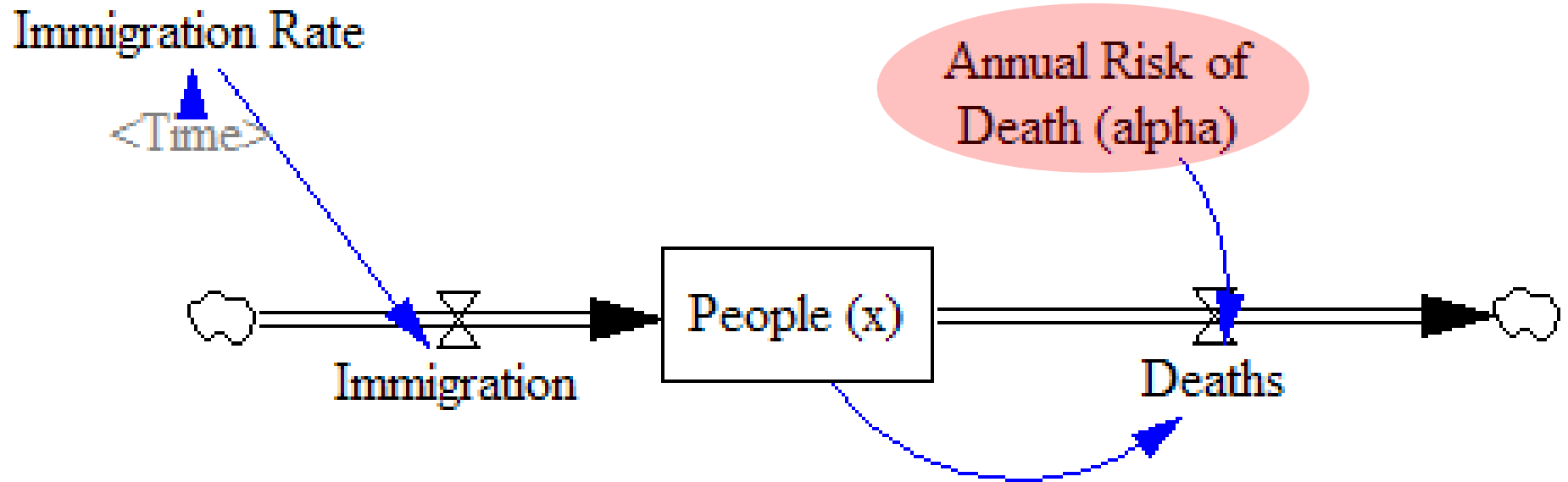
- Name: NaturalTBRecovery
- Triggered by: Rate
- Rate:  $\text{MeanDaysToNaturallyClearInfection} / \text{DaysPerTimeUnit}$
- Action: `traceIn("Naturally recovering to LTBI...")`
- Guard: (empty)

The left sidebar shows the project structure, including parameters, functions, environments, and statecharts. The bottom status bar indicates the cursor position at X=455, Y=420.

# Rates & Flows

- We've seen fixed rates before – in the form of “transition rates” in System Dynamics models
- Within the System Dynamics model, a flow out of a stock was commonly set by the multiplication of the
  - Stock
  - Some rate of transition
- We use different names for these rates
  - “Transition rates”
  - “Likelihood of transition per *Unit Time*”
  - Transition (e.g. “infection”, “mortality”) “hazard”

# Example Fixed Transition Rate/Hazard





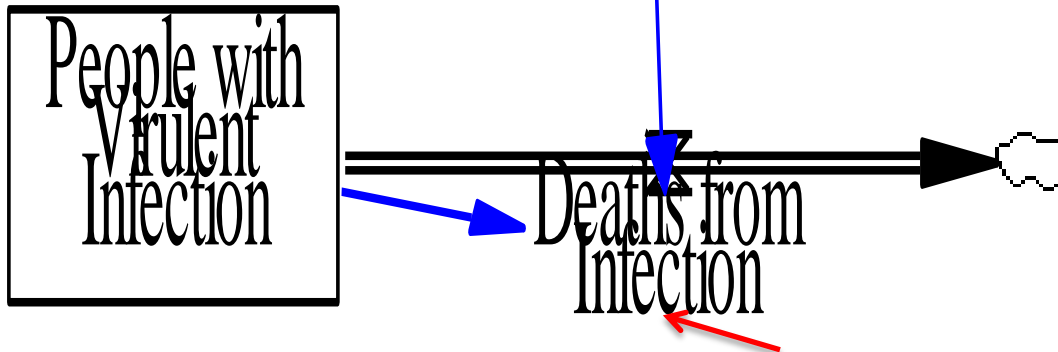
# Example Fixed Transition Rate/Hazard

The transition rate is the *reciprocal* of this number i.e.

1

$\frac{1}{\text{Mean time until Death}}$

Mean time until Death



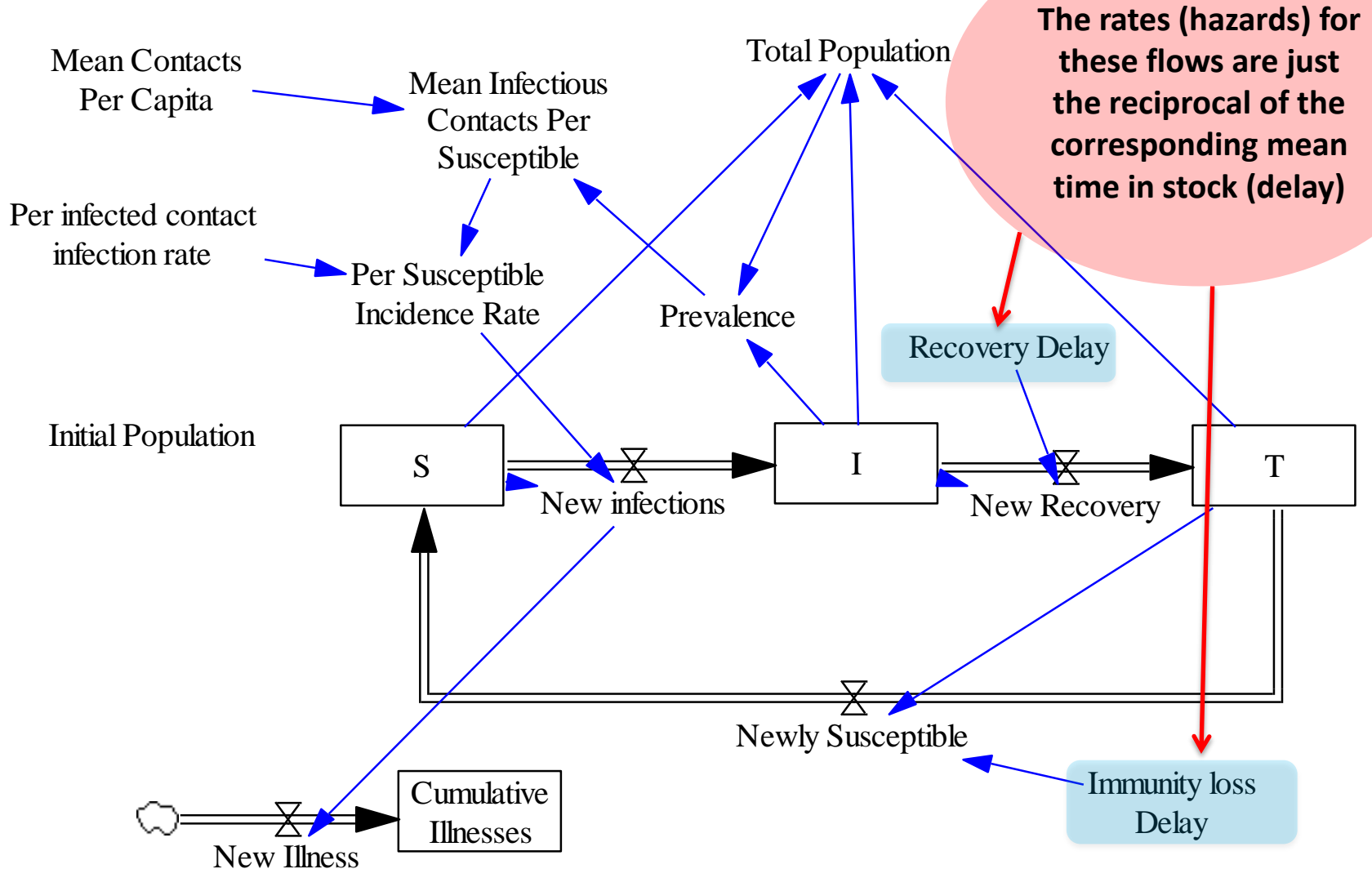
People with Virulent Infection/Mean time until Death

=

People with Virulent Infection\*(1/Mean time until Death)

i.e. People with Virulent Infection\*Rate

# First Order Delays in Action: Simple SIT Model



# Transition Type: Variable Rate

The screenshot displays the AnyLogic Advanced software interface, titled "AnyLogic Advanced [EDUCATIONAL USE ONLY]". The main workspace shows a statechart for a TB progression model. The states are represented by yellow rounded rectangles: "TBSusceptible", "LTBI", "UnDiagnosedActiveTB", and "DiagnosedActiveTB". Transitions are shown as arrows between these states, with decision diamonds for "WhetherInfected" and "WhetherPrimaryProgression". A transition from "UnDiagnosedActiveTB" to "UnDiagnosedActiveTB" is highlighted with a blue arrow, indicating it is the selected transition. The "Reactivation - Transition" properties window is open, showing the transition name "Reactivation", triggered by "Rate", and the rate expression "ReactivationRateForCKDStage()". The action is "println(\"Reactivated\");".

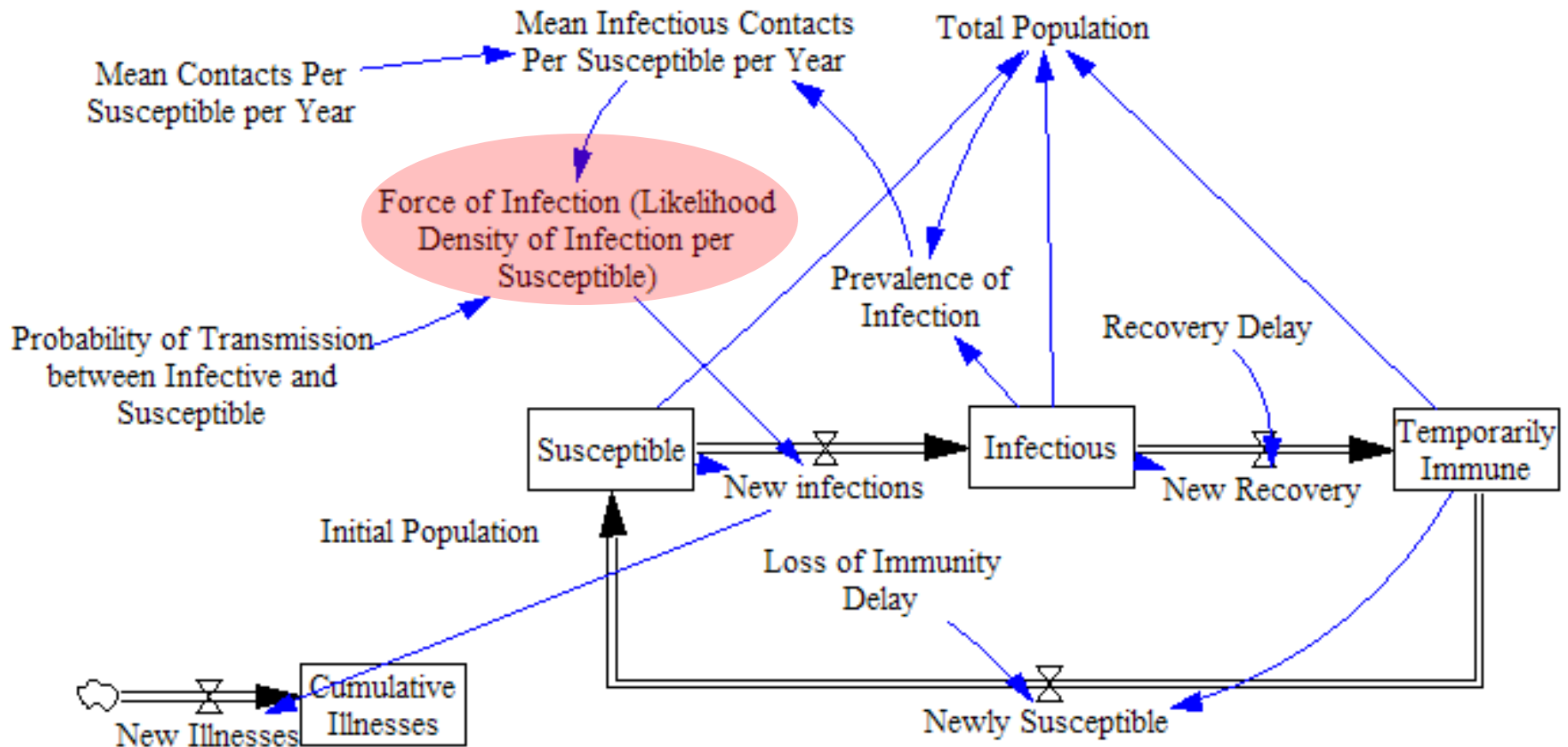
The left sidebar shows the project structure for "TBv1\*" under "Main". The "Person" statechart is selected, showing its parameters, functions, environments, embedded objects, presentation, and plain variables. The "Statecharts" section lists "TBProgressionStatechart" and its sub-elements: "TBSusceptible", "TBIInfectiousContact", "WhetherInfected", "TBTransmission", "WhetherPrimaryProgression", "PrimaryProgression", and "UnDiagnosedActiveTB".

The bottom-left pane shows a list of problems, including "Engine.log cannot be resolved" and "Cannot make a static reference to the non-static method getCurrentState()". The bottom-right pane shows the "Reactivation - Transition" properties window with the following details:

- Name: Reactivation
- Triggered by: Rate
- Rate: ReactivationRateForCKDStage()
- Action: println("Reactivated");
- Guard: (empty)

The status bar at the bottom indicates "Selection" and "Cursor: X=455, Y=420".

# Example Transition Rate/Hazard



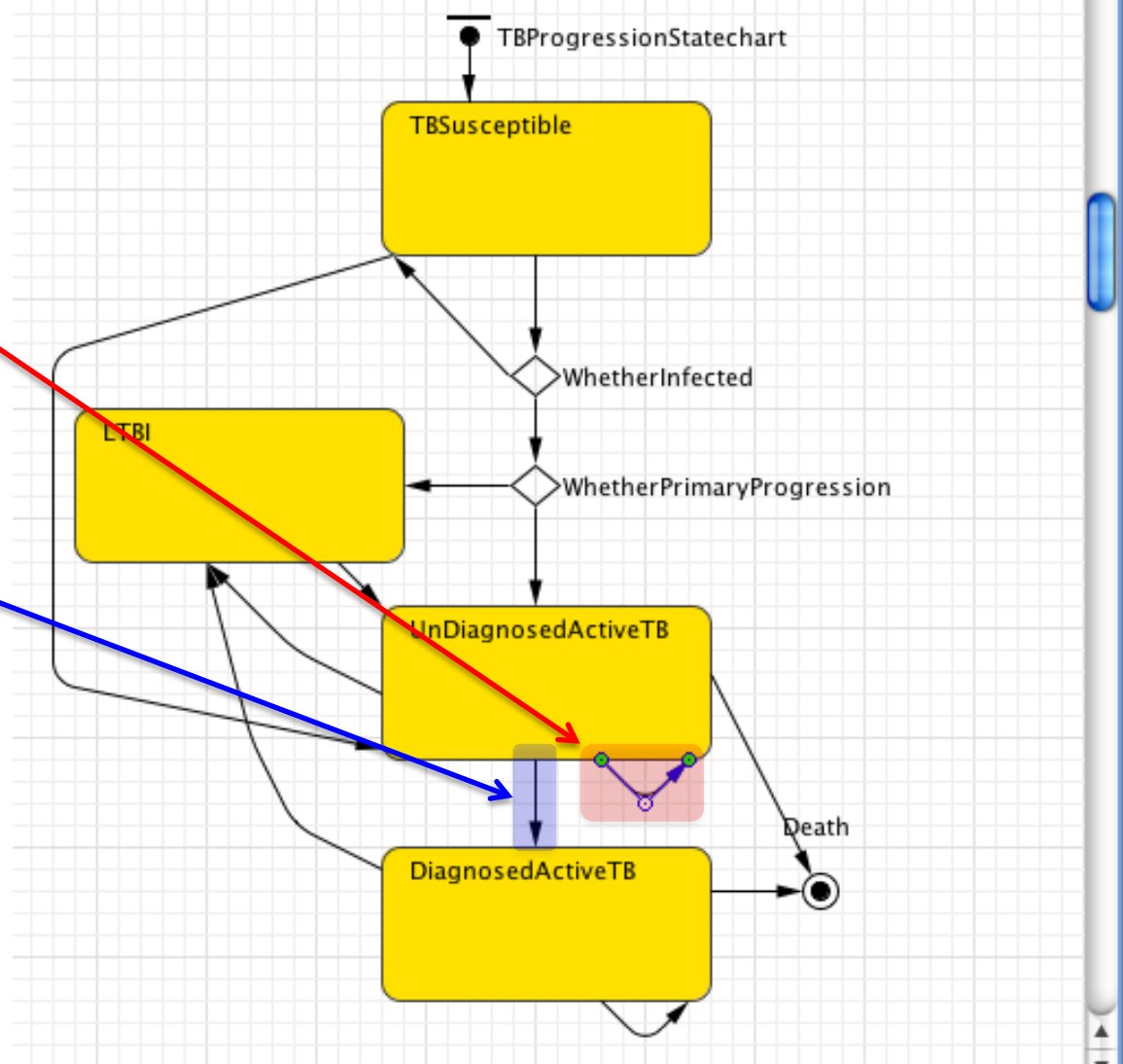
# Fixed Rates: Transition “Hazards”

- With “fixed rates”, we are specifying rates of transitions
- Because we are dealing with the chance that each individual transitions, we don’t need to multiply by the number of people at risk
  - Here, there is just 1 person at risk!
- As in SD models, these rates can change over time, but the statechart needs to be “made aware” of these changes
  - Leave & go back into current state (circular transition)
  - Likely: trigger “change” event in Agent (see manual)

# Special Elements: Self-Transition

(Use if Wish To Have State Register Changing Out-transition rates)

The **self-transition** will “make the state realize” that the rate associated with any out transition (e.g. **this one**) has changed



# Transition Type: Fixed Residence Time (Timeout)

The screenshot displays the AnyLogic Advanced software interface for modeling a Tuberculosis (TB) progression statechart. The main workspace shows a statechart with the following states and transitions:

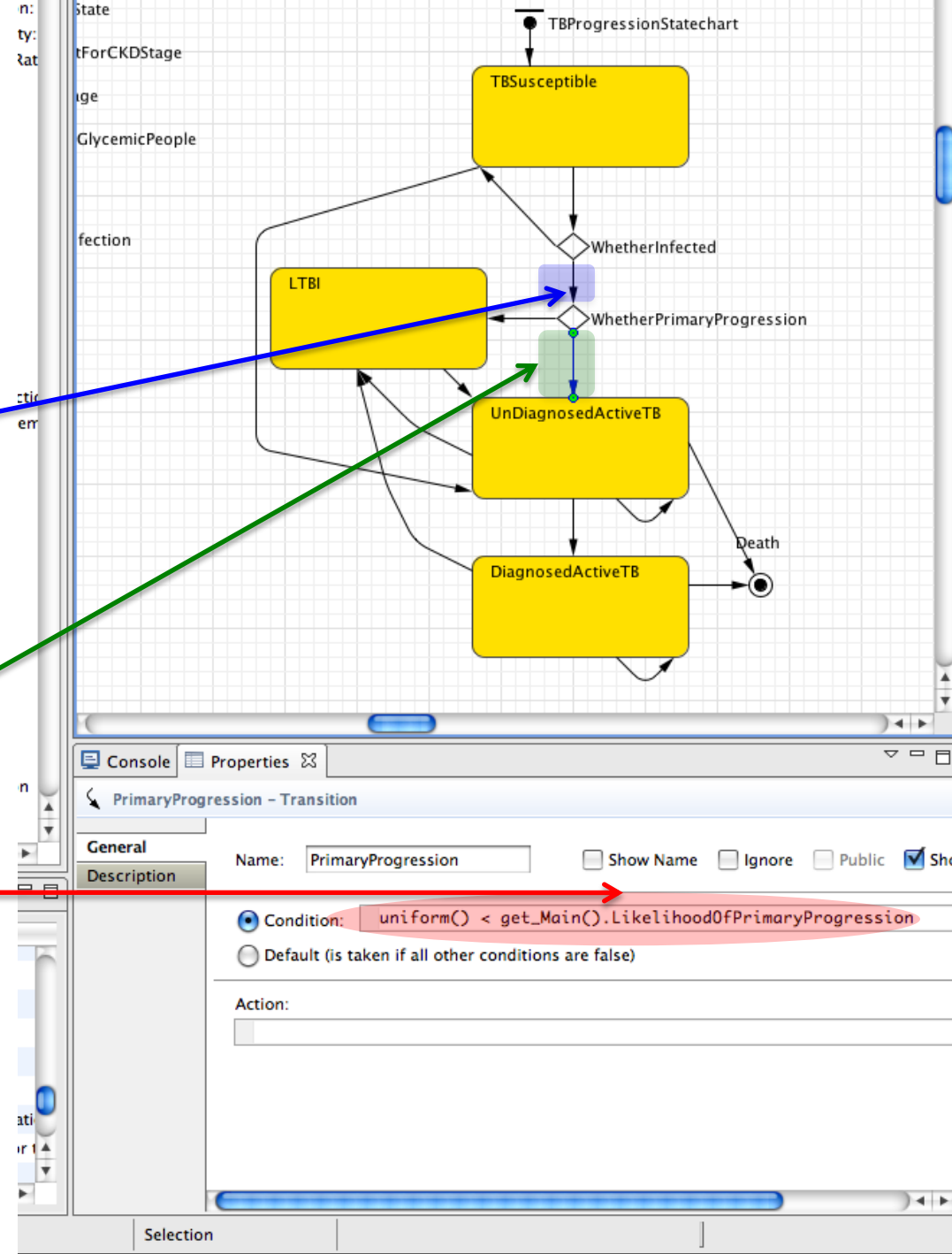
- States:** TBSusceptible, LTBI, UnDiagnosedActiveTB, DiagnosedActiveTB.
- Transitions:** WhetherInfected, WhetherPrimaryProgression, and a 'Diagnosis' transition.
- Flow:** TBSusceptible transitions to LTBI via 'WhetherInfected'. LTBI transitions to UnDiagnosedActiveTB via 'WhetherPrimaryProgression'. UnDiagnosedActiveTB transitions to DiagnosedActiveTB via 'Diagnosis'. Both UnDiagnosedActiveTB and DiagnosedActiveTB have self-loops.

The 'Diagnosis' transition is configured with the following properties:

- Name:** Diagnosis
- Triggered by:** Timeout
- Timeout:** `get_Main().DaysUntilDiagnosis/DaysPerTimeUnit`
- Action:** `traceln("Diagnosis performed");`

The left sidebar shows the project structure for 'Tbv1\*' and 'Person', including parameters, functions, and statecharts. The bottom-left pane shows a list of errors, such as 'Engine.log cannot be resolved' and 'Cannot make a static reference to the non-static method getCurrentState()'. The bottom-right pane contains various tool buttons like 'Action', 'Analysis', and 'Presentati...'.

# Example Conditional Transition



The **incoming** transition into “**WhetherPrimaryProgression**” will be routed to this **outgoing** transition if **this condition** is true

Console Properties

PrimaryProgression - Transition

General

Description

Name: PrimaryProgression  Show Name  Ignore  Public  Show

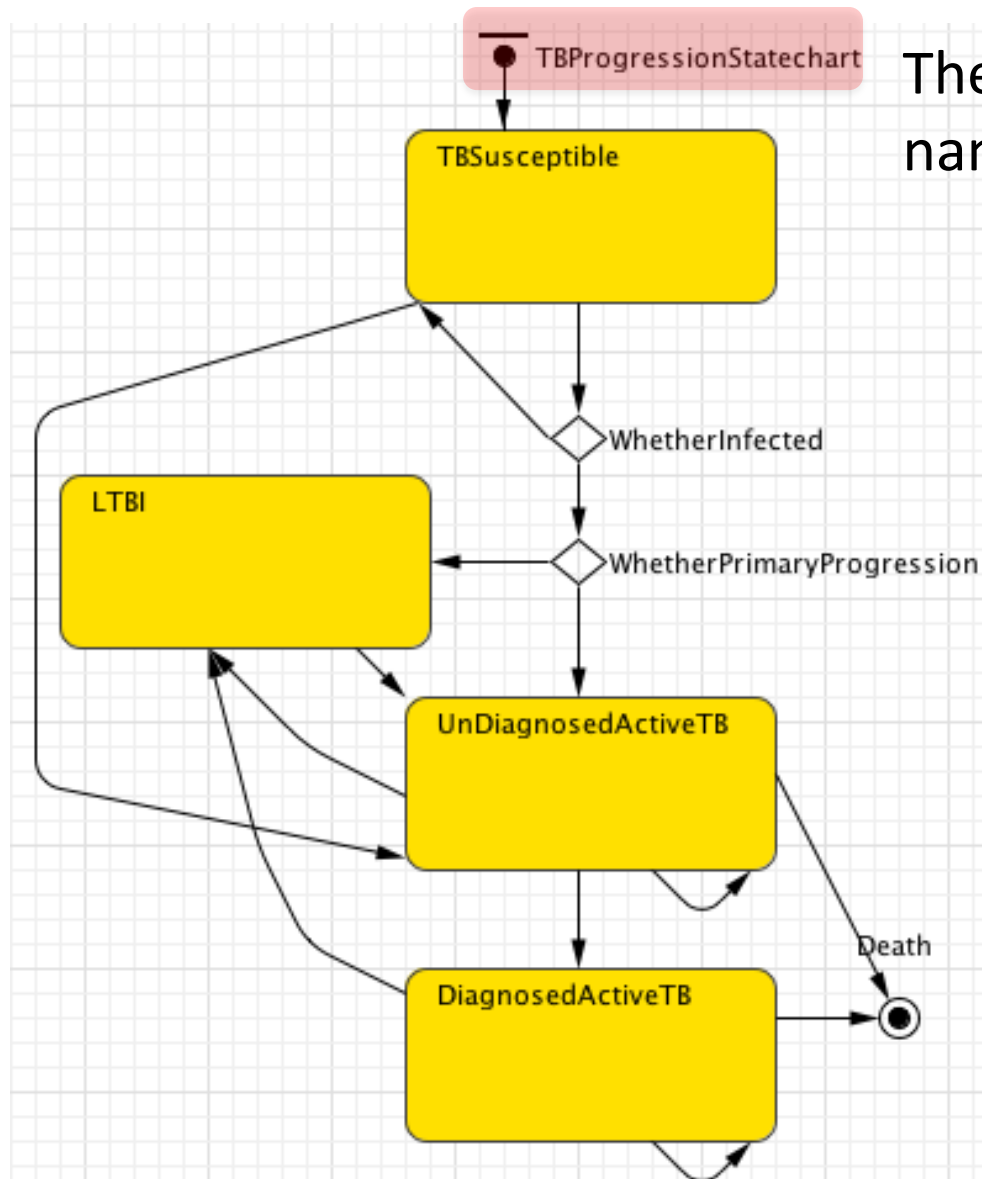
Condition: `uniform() < get_Main().LikelihoodOfPrimaryProgression`

Default (is taken if all other conditions are false)

Action:

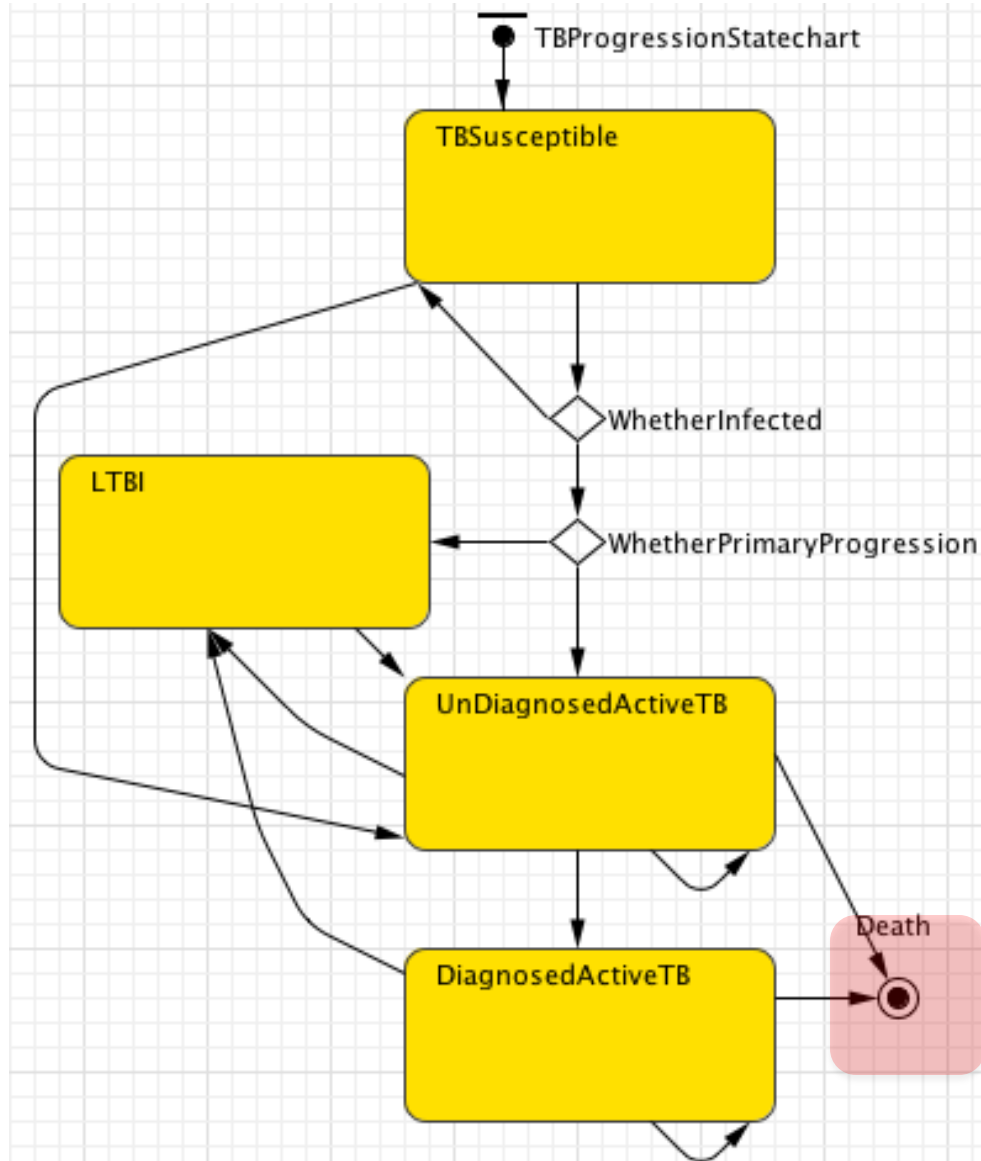


# Special Elements: Entry Point



The associated text is the name of the statechart!

# Special Elements: Exit Point





Hands on Model Use Ahead

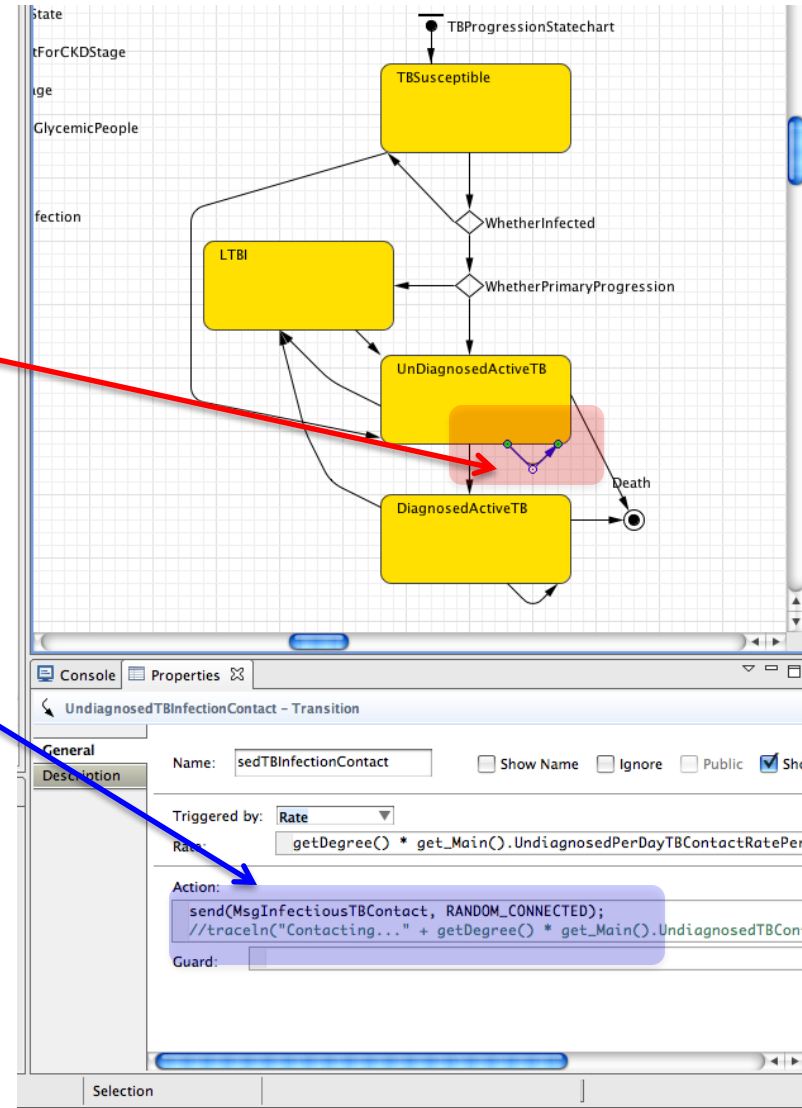


Load model: TBv1.alp

# Special Elements: Self-Transition

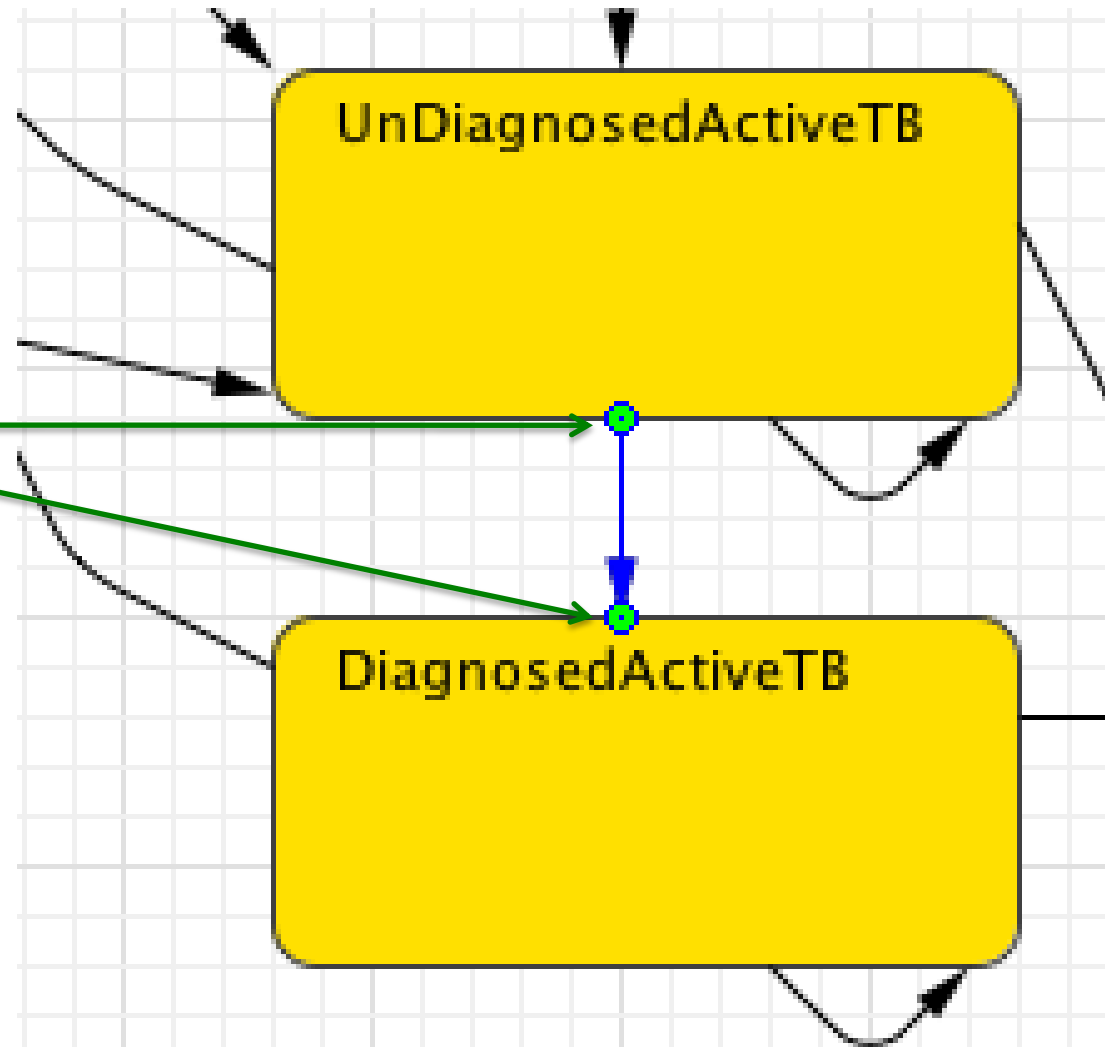
(Use if Wish To Trigger an Action w/o Leaving State)

The **self-transition**  
will invoke **this action**  
when it occurs



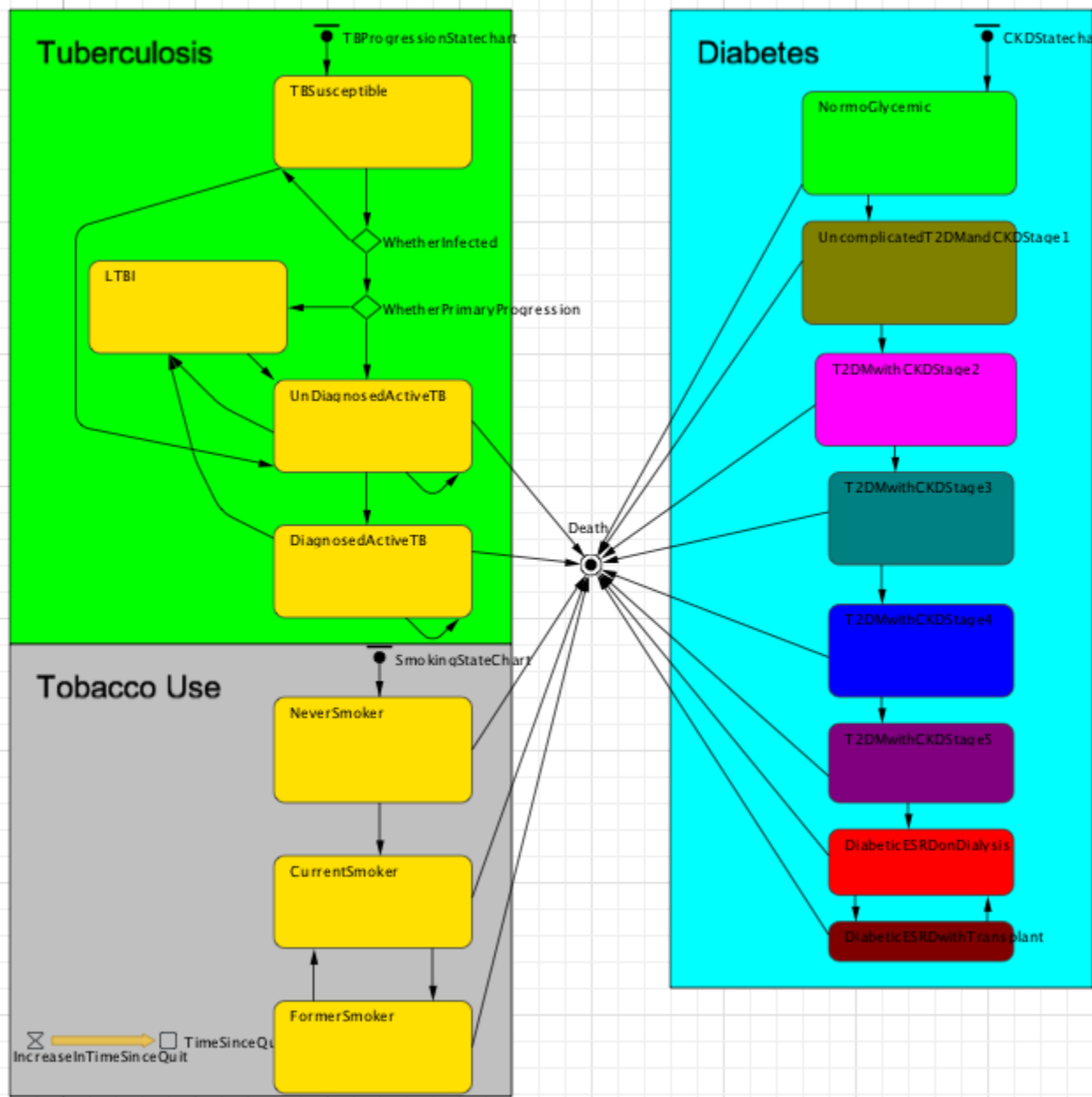
# Tip: Confirming Transition Connectivity

- Ensure that both sides of the transition show **green circles** when connected
  - Otherwise, may appear connected but will actually be disconnected!



# Parallel Statecharts

- By default, each statechart evolves independently.
- If coupling is desired, can make transitions/actions dependent on state of other statecharts

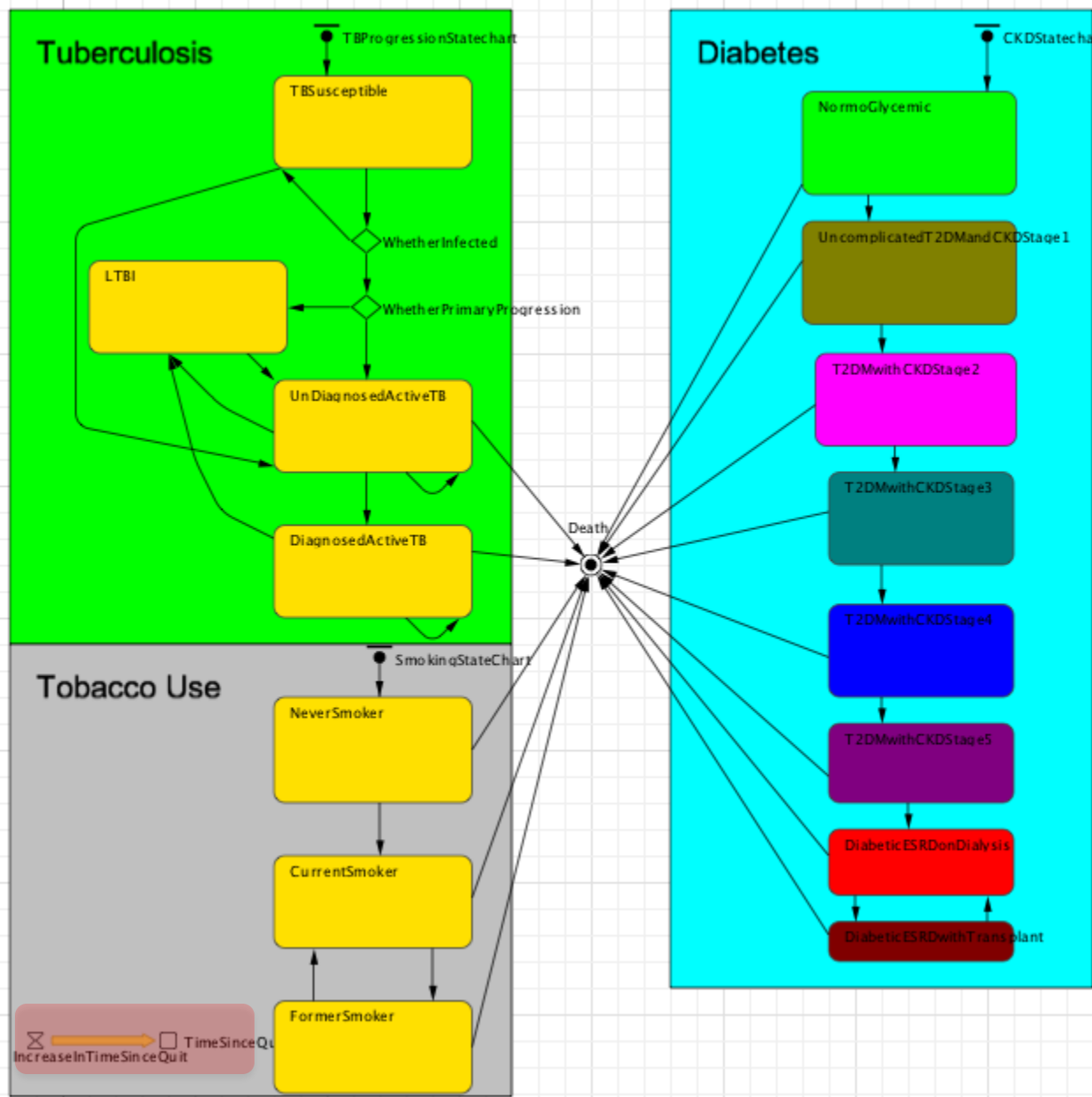


# Comparison with Aggregate Stock & Flows

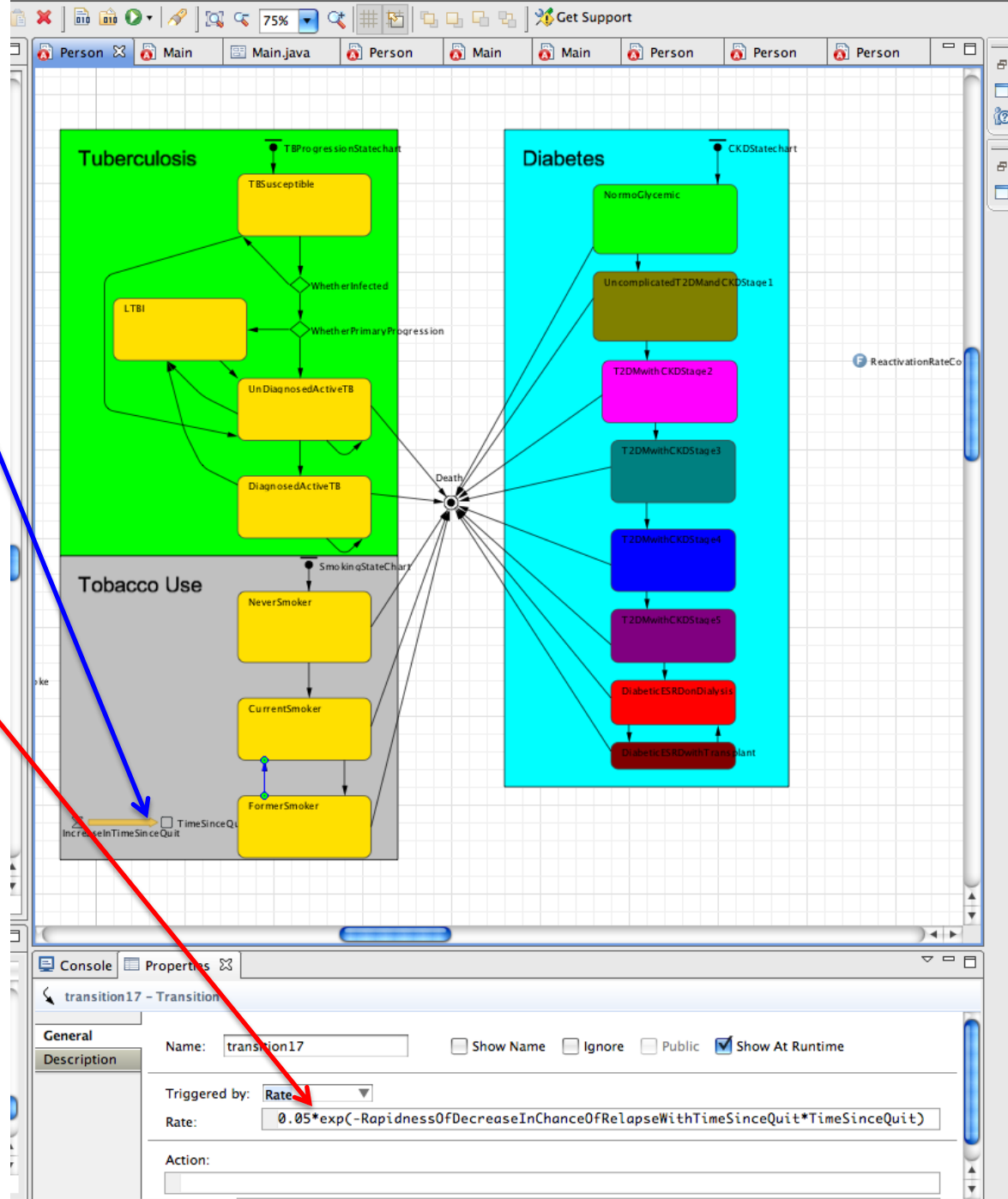
- As for aggregate stocks & flow, individuals' states are discrete
- Unlike aggregate stocks & flows
  - One state within a given statechart is active at a time
  - For parallel flows (e.g. comorbidities), there is no need for considering all combinations of the possible states
  - We can keep track of how long an individual is in a given state & adjust the transition rate accordingly

# Parallel Transitions

- Example recording the residence time in a state (via a stock with unit inflow -- i.e. just accumulates the time present in that state)



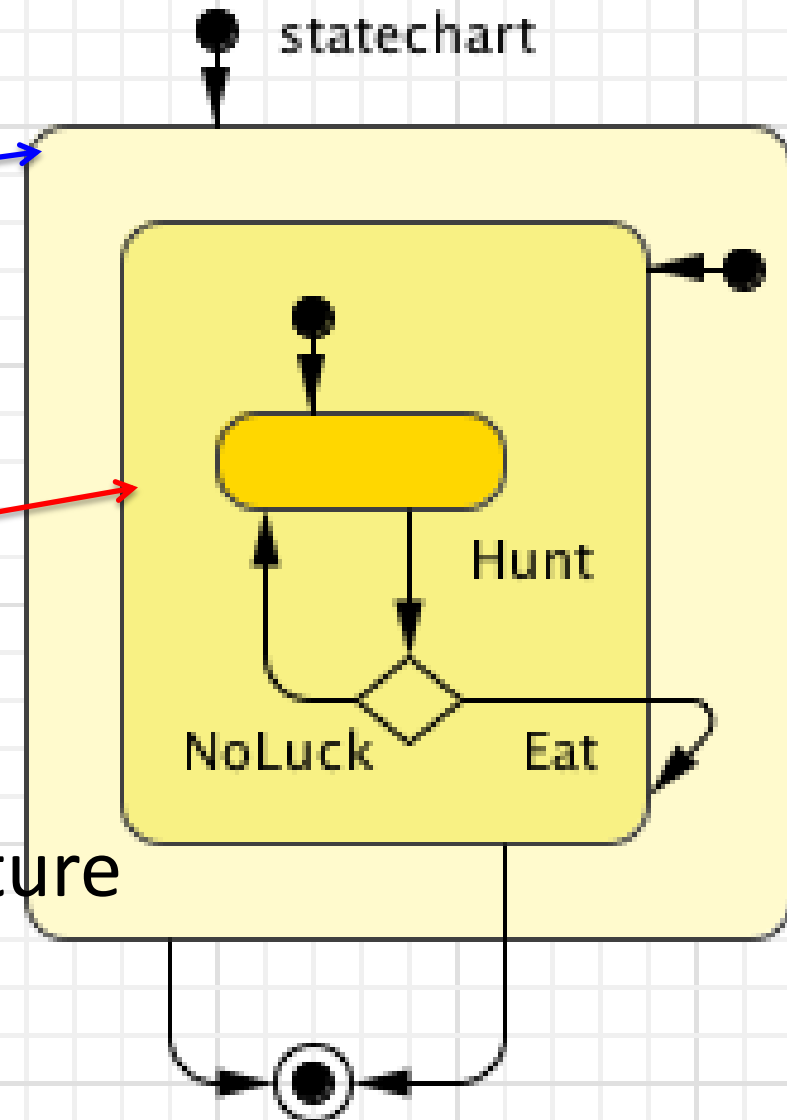




- The **residence time** in the state determines the **transition rate** out of that state.
- Transition rates depending on residence time are generally not possible with aggregate models

# Advanced Element: Hierarchical States (*Predatory Prey Agent Based* by xitek)

- The **outermost state** captures time since born (for natural deaths)
- The **middle-state captures** time since last ate (for deaths by hunger). [Eating reenters]
- The inner state transition capture hunting frequency & success



# Natural Death Transition

The screenshot displays a statechart editor interface. On the left, a project tree shows a hierarchy: Project > Contact > Presentation > Simulation: Main > Presentation > Predator Prey Agent Based > Hare > Statecharts > statechart. The main workspace shows a statechart for a Lynx agent. The statechart includes a state with a start node, a transition labeled 'HaveBabies' (triggered by the event 'Width'), a state transition labeled 'cell', a state transition labeled 'NoLuck', a state transition labeled 'Eat', and a state transition labeled 'Hunt'. A transition named 'tran3' is highlighted in red, with its properties shown in the bottom right panel.

**tran3 - Transition**

General	Name:	tran3	<input type="checkbox"/> Show Name	<input type="checkbox"/> Ig
Description	Triggered by:	Timeout		
	Timeout:	get_Main().LynxLifeExpectancy		
	Action:			
	Guard:			

**Problems**

Description
Engine.log cannot be resolved
Engine.log cannot be resolved
Engine.log cannot be resolved
Engine.log cannot be resolved

# Death By Hunger

(Note that Depends on Time in State – i.e. time Since last ate)

The screenshot shows a statechart editor interface. On the left is a project tree with folders for 'Presentation', 'Simulation: Main', 'Predator Prey Agent Based', 'Hare', and 'Lynx'. The 'Lynx' folder is expanded, showing 'Parameters', 'Plain Variables', and 'Statecharts'. The 'Statecharts' folder is further expanded to show a 'statechart' containing 'Alive', 'DiedBecauseO', 'Dead', and 'WasEatenByAL'. The main workspace displays a statechart diagram. A state named 'statechart' contains a state 'Alive' (yellow circle) and a state 'Dead' (black circle). A transition labeled 'Hunt' leads from 'Alive' to a decision diamond. From the diamond, the 'NoLuck' path loops back to 'Alive', and the 'Eat' path leads to another state. A transition labeled 'tran4' is triggered by a 'Timeout' event and is associated with the 'DiedBecauseO' state. The 'Properties' panel at the bottom right shows the configuration for 'tran4 - Transition': Name: tran4, Triggered by: Timeout, Timeout: get\_Main().LynxHungerDeathThreshold, and Action: (empty). The 'Problems' panel at the bottom left shows four error messages: 'Engine.log cannot be resolved'.



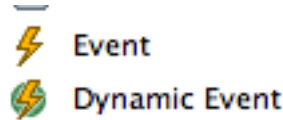
# Tips on Statechart Code

- Each State & Transition has an integer index
  - This is accessed via a (static) constant holding the name of state within the statechart class (*statechart.StateName*)
- To determine length of time spent in state
  - *StateName.getLocalTime(StateIndex)*
- To determine current state
  - *statechart.getActiveSimpleState()*
- To find out if a state (either simple or composite) is currently active
  - *statechart.isStateActive(StateIndex)*

# **EVENTS IN ANYLOGIC**

# Rates & Events

- *Rates* and *Timeouts* are associated with types of events in AnyLogic
- Events can also be declared explicitly from the palette



- Dynamic events can have multiple instances
    - Each instance can be scheduled at the same time
    - The instances disappear after event firing
  - Regular (static) events can be rescheduled, enabled/disabled, but can only have one scheduled firing at a time
- There are some subtleties with events



# Event Times: Options for Event Scheduling

- Manually (via `restart()`) – see following slides)
- When boolean condition changes (depends on *onChange* being called)
- One-time
  - Can go off at a particular time (specified as a calendar time or as a double-precision value)
- At some initial time and then cyclically beyond with set “timeout” period
  - The timeout period is set according to the time unit
  - This goes off after *exactly* the timeout time
- At a specified Poisson Rate
  - Interarrival time is exponentially distributed!
  - Mean time between events is reciprocal of rate (i.e.  $1/\text{rate}$ )

# Event Subtleties

- Be very careful of what you count on for recomputation of rate – may think was recomputed, but hasn't been
- Event rates (and likely event timeout times) are only computed occasionally, not continuously
  - These are computed when
    - Explicitly call event methods
      - start()
      - restart()
      - onChange()
    - When event fires and requires restarting
    - (For outgoing transitions) when enter a state in a statechart

- Calling “reset” will disable a rate until re-enable (e.g. with call to *restart()*)