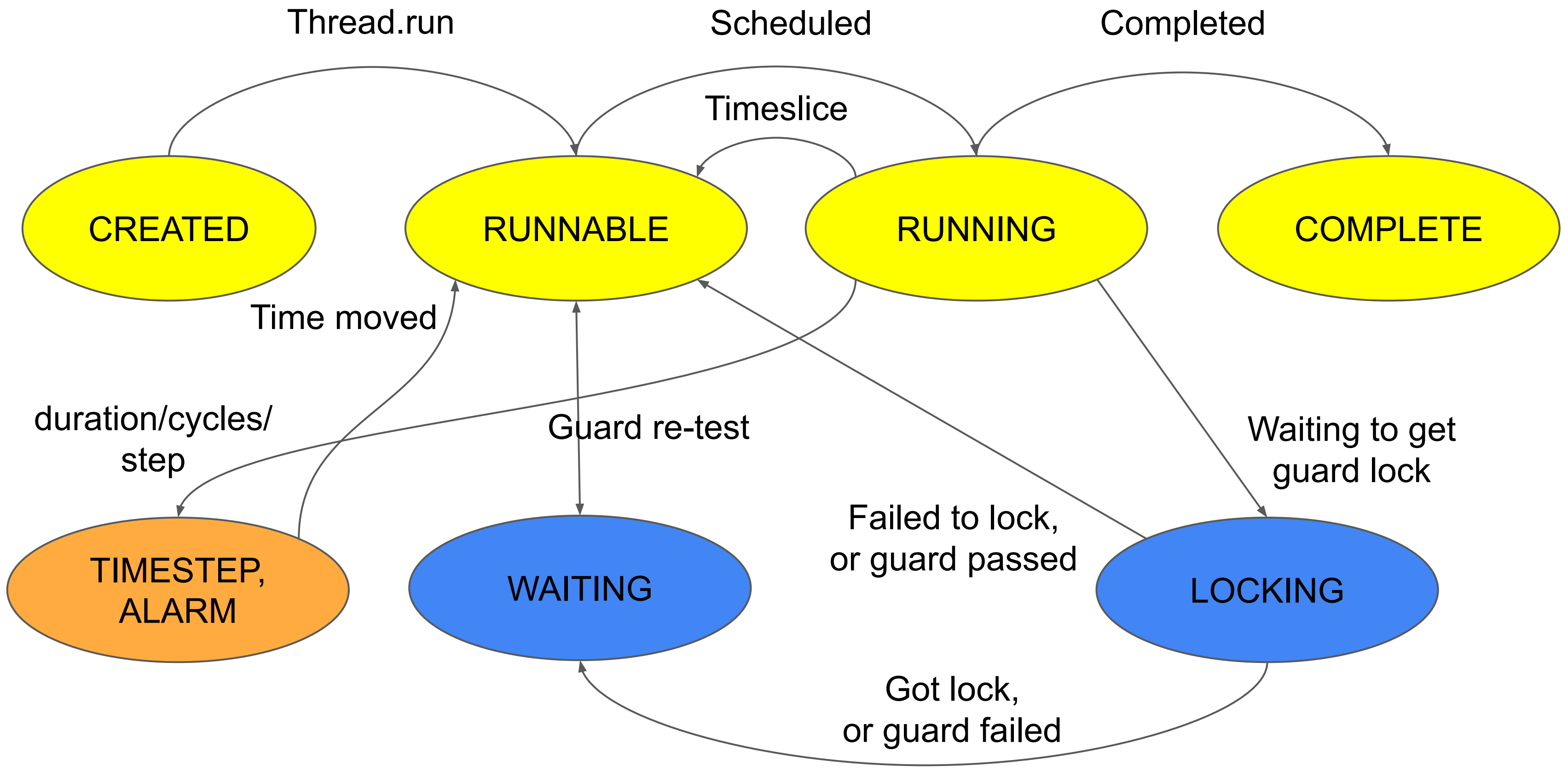
VDMJ
Execution and Scheduling

Nick Battle, July 22

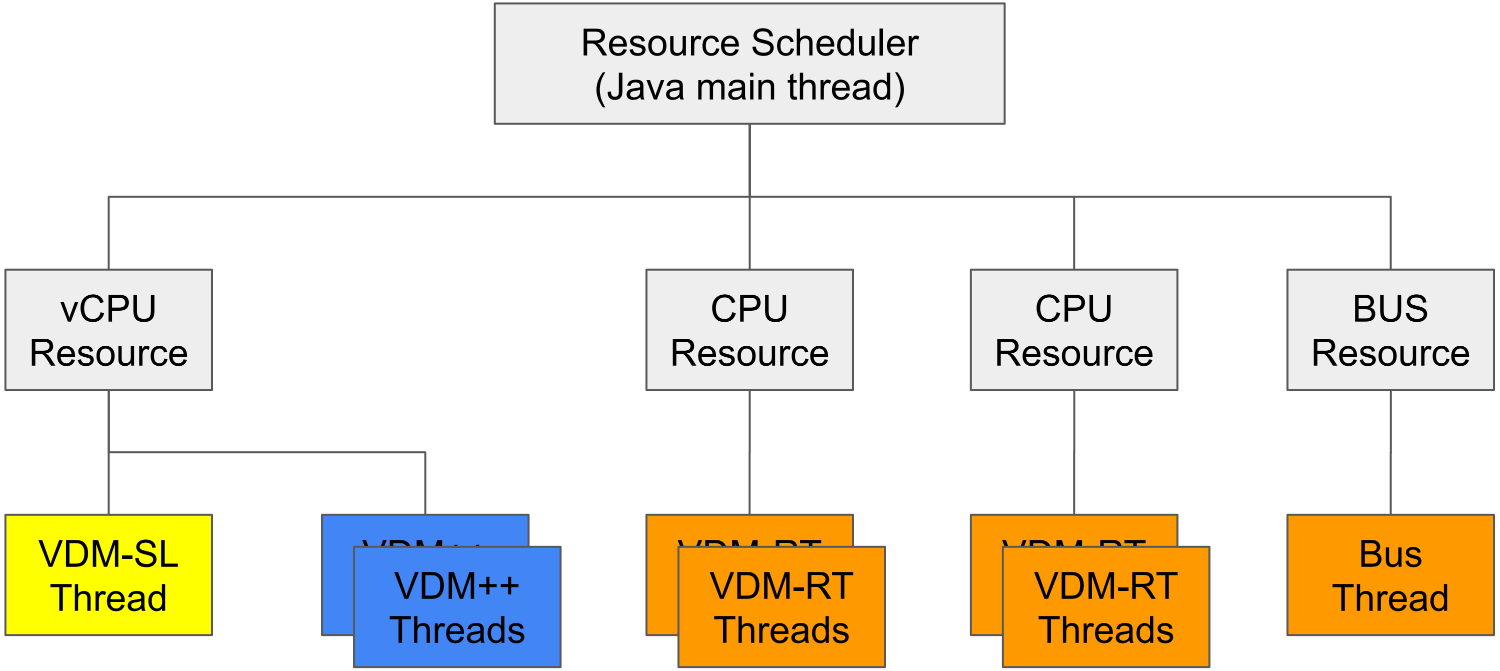
VDMJ Execution and Scheduling

- VDM-SL execution walks over the "IN" tree, calling the *eval* methods:
  - The *eval* is passed a Context = map of TCNameToken to Values
  - Contexts chain together to form a stack
  - A *RootContext* represents the base of a "frame" and refers to module/object data
  - The bottom of the stack is a *RootContext* that refers to global data (top level definitions)
  - VDM threads are *SchedulableThreads extends Thread*
  - *RunState* is CREATED, RUNNABLE, RUNNING, COMPLETE
- VDM++ adds objects with threads and coordination:
  - Every exp/stmt execution starts with a Breakpoint *check*. This calls *SchedulableThread.step*
  - After a fixed timeslice (number of steps) schedules another RUNNABLE thread
  - New LOCKING, WAITING states used to coordinate operation calls
- VDM-RT adds a CPU/BUS architecture and discrete time control:
  - *ResourceScheduler* coordinates *Resources* (CPU/BUS) which own *SchedulableThreads*
  - New TIMESTEP and ALARM states used to coordinate movement of time

VDMJ Execution and Scheduling



VDMJ Execution and Scheduling



VDMJ Execution and Scheduling

- *ResourceScheduler* owns CPU and BUS *Resources*
    - Runs in Java's "main" thread (or async *ExpressionExecutor* thread in DAP)
    - The *start* method controls one entire execution, passed a *MainThread*
    - Spec "init" process not executed under the *ResourceScheduler* (uses *InitThread*)
- *Resources* have a list of *SchedulableThreads*
    - VDM-SL just uses the implicit "vCPU" and a single thread
    - VDM++ also uses the vCPU, but can create multiple threads
    - VDM-RT can declare multiple CPU and BUS resources, with multiple threads
    - Only one *SchedulableThread* active on a CPU at once
    - *ResourceScheduler* allocates time to all resources in a round-robin
    - Coordinates time stepping between all resources in VDM-RT
- *Resources* each have a scheduling policy
    - Policies are <FCFS> (first come first served) or <FP> (fixed priority)
    - vCPU uses FCFS