Conceptional problems of transaction-based modeling and its implementation in SimEvents 4.4

phwt
Private Hochschule
für Wirtschaft und Technik

Workshop of the ASIM/GI-Sections STS and GMMS
HS Ulm, 09. 03. 2017

**Lars Austermann, Peter Junglas,
Jan Schmidt, Christian Tiekmann**

Content

Introduction

Discrete systems

- many different modeling paradigms
- graphical methods
  - abstract (Petri nets, state graphs)
  - medium level of abstraction (entities running through components)
  - concrete (material flow applications, e.g. PlantSimulation)
- medium level widely used
  - sufficiently abstract to be universally applicable
  - concrete enough to be comprehensible by users
  - process based (active components, e.g. Arena) or transaction based (active entities, e.g. SimEvents)

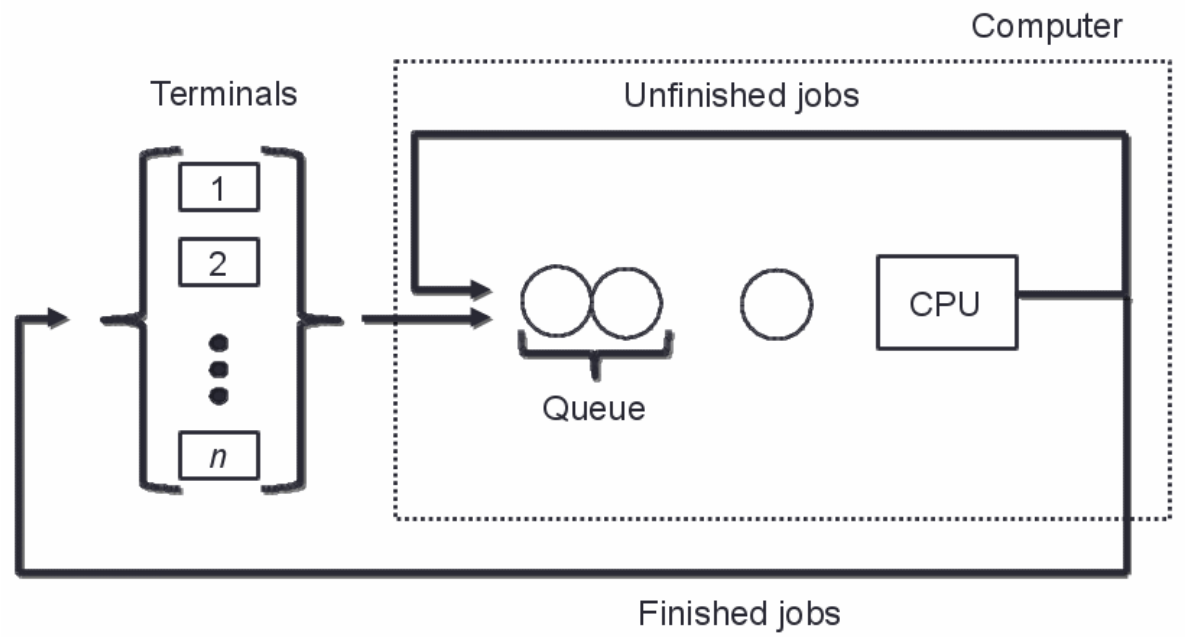
Closer look at transaction based modeling

- no well-established set of basic features and components (cf. SimEvents 4.4 → 5.0)
- open issues
  - shortcomings of current implementations
  - missing concepts or components
  - reasonable set of components
- strategy
  - implement many different applications
  - concrete: SimEvents 4.4 (Mathworks)

timeshared

Time-shared computer model

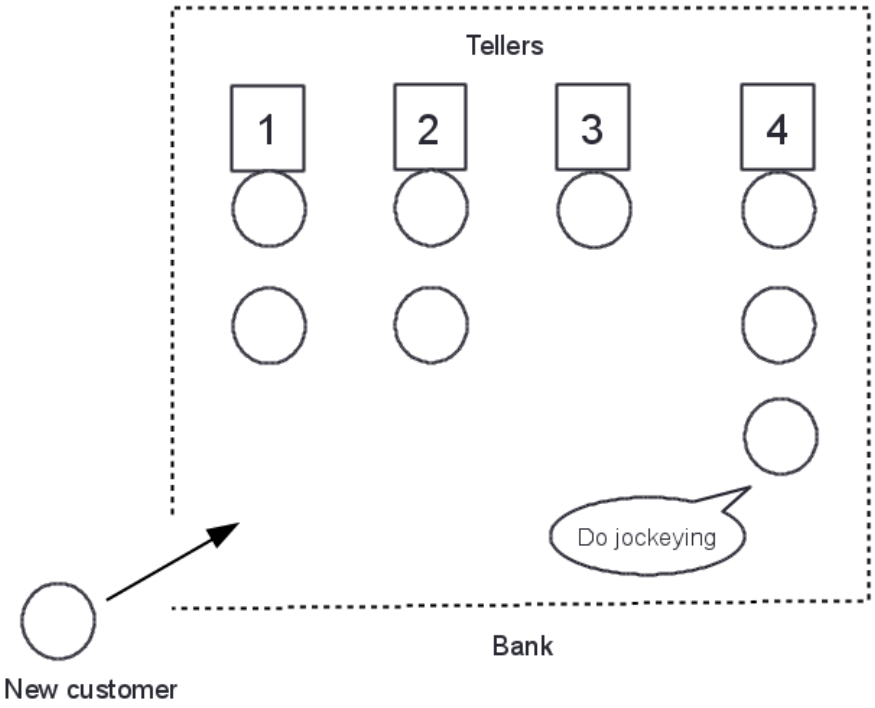
- from Law's textbook
- several terminals send jobs of varying computing time demands
- processed in time slices using a round-robin scheduler



multiteller

Multiteller bank with jockeying

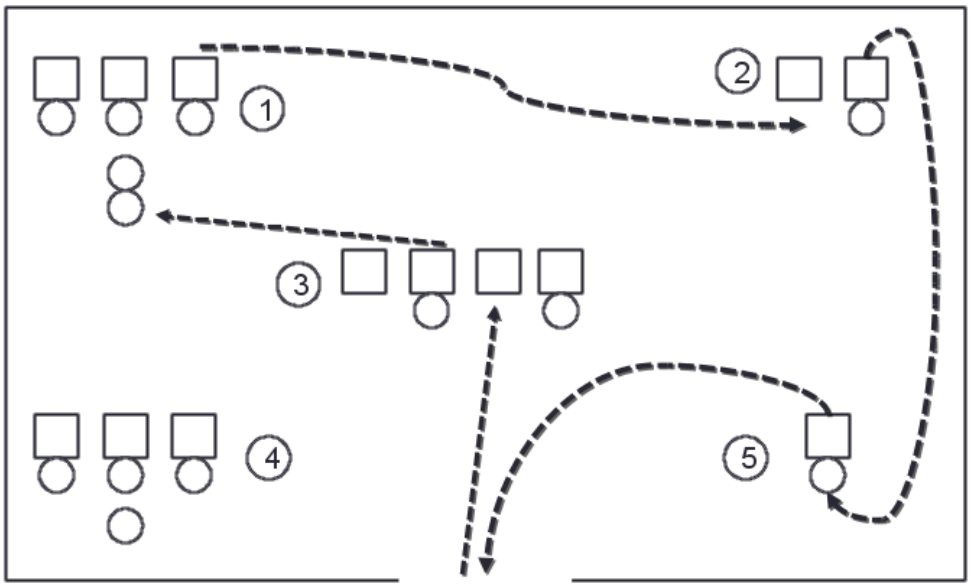
- from Law's textbook
- bank with several teller queues
- customers are allowed to change to a shorter queue (jockeying)



jobshop

Job-shop model

- from Law's textbook
- factory with five workstations
- variable kinds of jobs with different paths through the stations

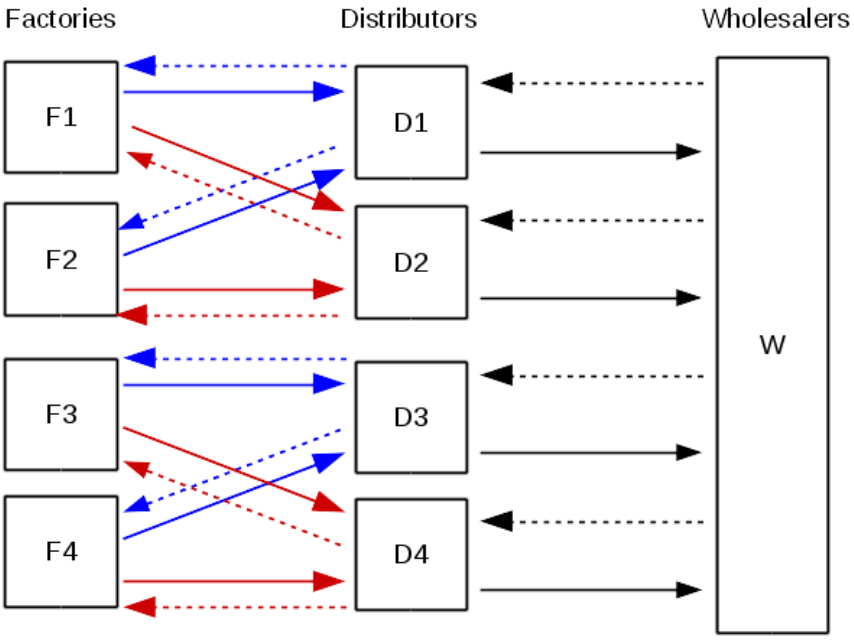


Jobtype 1

supplychain

Supply chain model

- from Argesim benchmark C14
- supply chain with factories, distributors and wholesalers
- wholesalers order different products from distributors
- distributors use special strategies to comply with the demand

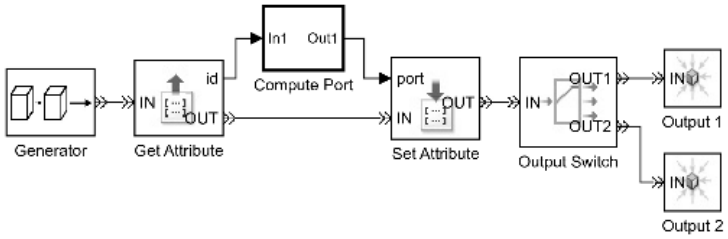


Handling of concurrent events

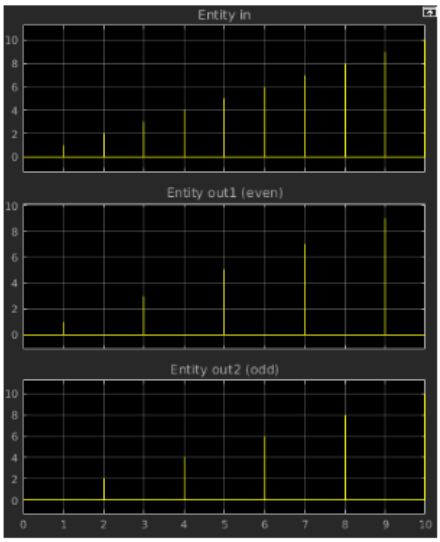
Who defines the order of concurrent events?

- often: a global event queue
- in TBM: events are defined by blocks → order not well-defined

Example testNullserver



- leads to error message
- undefined, what comes first at `Set Attribute`: new entity or new attribute value



Solutions

- ignore message → entities leave at the wrong port!
- null server after `Get Attribute` → model works

Old problem

- BUFFER command of GPSS

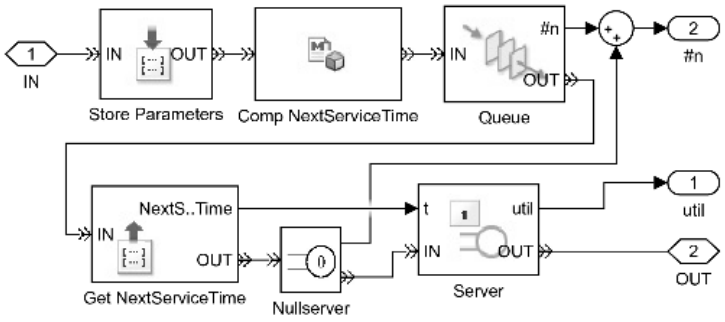
Handling of concurrent events

Null server - no universal solution

- its output may be blocked → null server stores entity
- no problem, if an (unlimited) queue follows

Careful with null server between queue and server

- example timeshared/CPU



- server busy → next entity sits in the null server
- has to be taken into account, e.g. for computation of queue length!

Null server workaround conceptually wrong

- its purpose is not related to storage → easily forgotten
- creates unexpected problems

Separation of entities from a queue

Several applications, e.g.

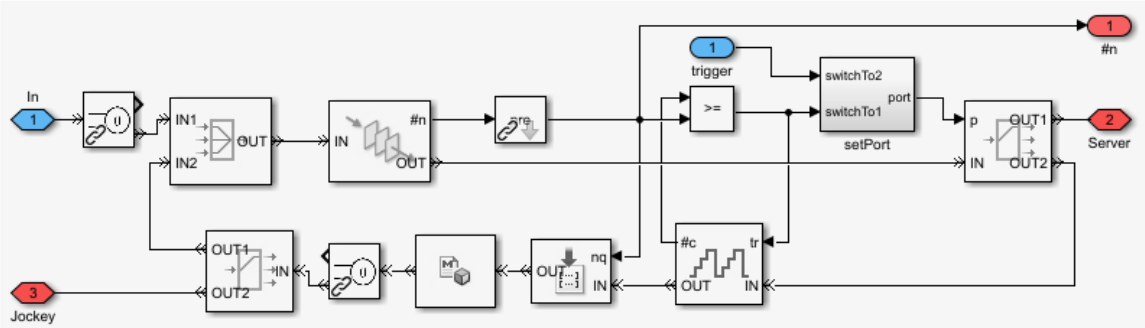
- leaving a queue after a maximal waiting time (**reneging**)
- changing to shorter queue (**jockeying**)

In SimEvents

- reneging is implemented
- jockeying is not

Shuffle queue

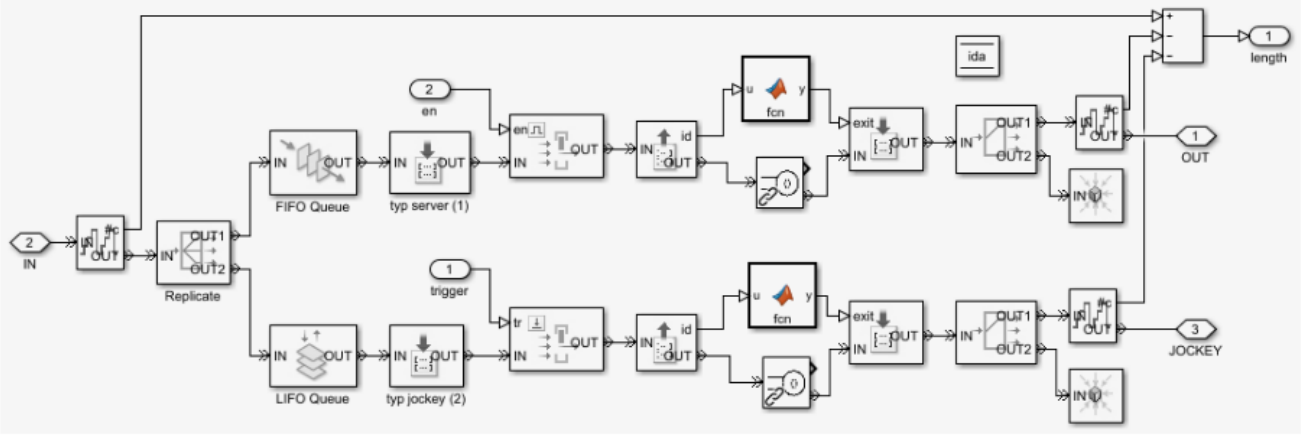
- jockey event leads to cyclic walk of queue entities
- last one gets separated



Separation of entities from a queue

Clone queue

- incoming entities are duplicated and routed to a FIFO and a LIFO queue
- jockey event → entity is taken from the LIFO queue
- bookkeeping device destroys clones, whose partner has already left



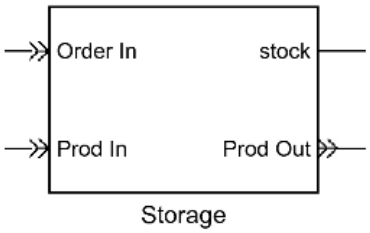
Result

- complex timing problems → cumbersome and error prone
- one needs generic solution
- maybe similar to user chains in GPSS

Storing entities

Storage component for supplychain example

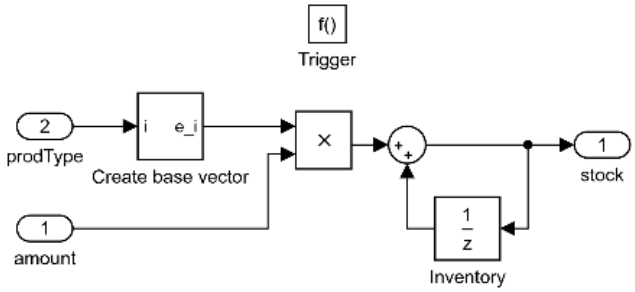
- input for product entities (with type and amount)
- input for order entities (with type and amount)
- output for product entities
- signal output for inventory vector
- parameter for initial stock



Possible storage components in SimEvents

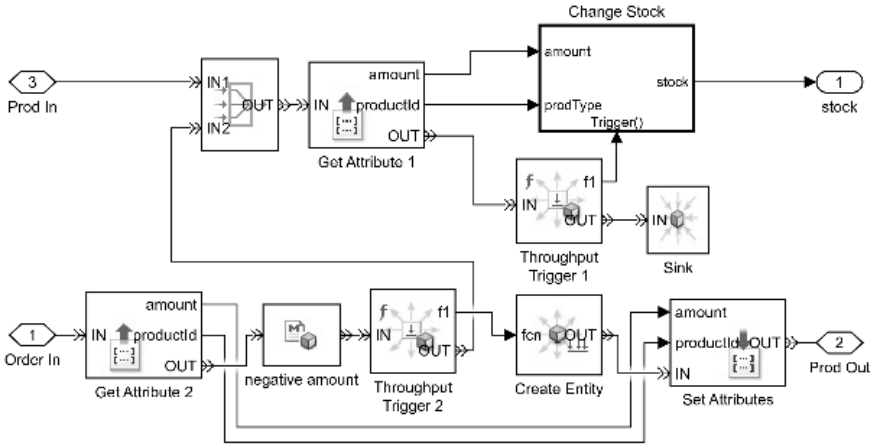
- N-server: how to get an entity of given type?
- resources: fixed amount!
- queues: one necessary for every product type!
- → do it yourself with Simulink

Implementation of inventory



Storing entities

Implementation of storage block



Result

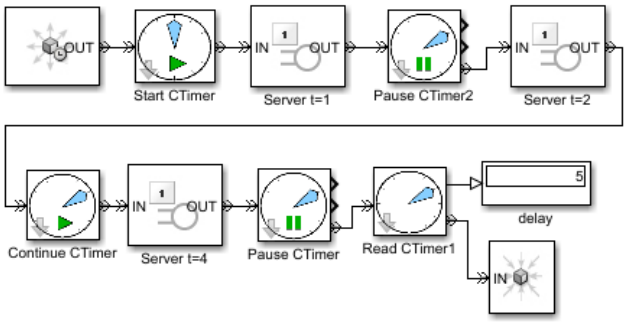
- works - but not really a "storage"
- design of a generic storage component ?
- special queue with internal advancement (user chains) ?

Time measurements across several blocks

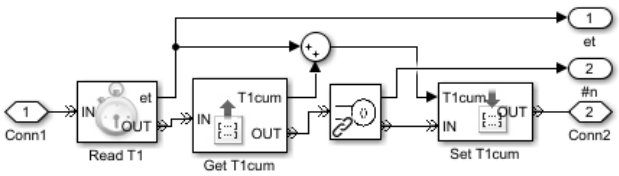
Total waiting time in several queues

- in SimEvents: only time between two points
- solution: accumulation of several waiting times in an attribute

New timer components for Start/Pause/Continue/Read



- submodell `PauseCTimer`



Time measurement across several queues

- problem with null server when accumulating between queue and server

Workaroud

- measure t_{Q+S} and t_S
- accumulate behind server (= before next queue) $t_Q = t_{Q+S} - t_S$

Statistical analysis

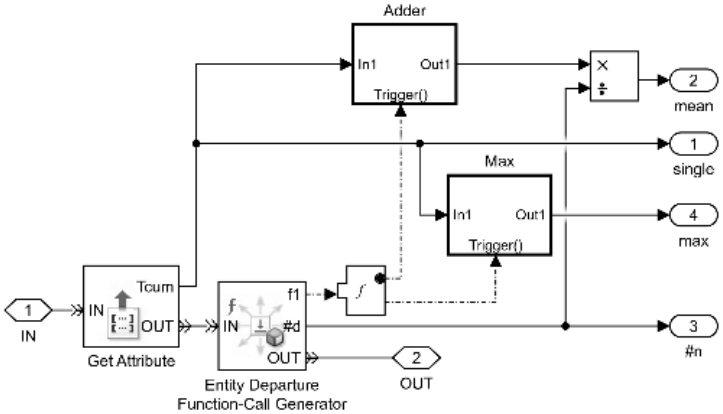
Decentral concept of TBM

- → no central collection instance
- block statistical data often not sufficient

Workaround

- collect data in entities
- final statistics component at the end of the entity path

Statistics components

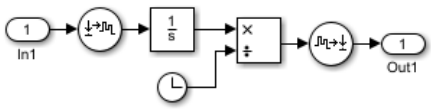


- atomic subsystems with 1/z block as accumulator

If necessary

- create report using Matlab

Computing time averages



- using continuous time domain

Conclusions

Some basic problems of TBM/SimEvents

- timing of concurrent events
  - null server is the wrong solution!
- implementation of alternative queueing policies
  - a new (or old) abstraction is needed
- storing and retrieving of entities
  - again: a basic concept is missing
- gathering and processing of statistical data
  - a few supplementing blocks would be handy

Proposal: new Argesim benchmark with

- modeling of jockeying queues
- statistical data over several queues
- additional complication: **many** queues

SimEvents 5

- complete redesign of the library
- based on a unifying theoretical description (but not DEVS)
- migration from SimEvents 4 difficult!
- is it better? We'll see ...

Conclusions

For the advancement of TBM we need

- fundamental abstractions
- theoretical analysis
- stable designs

Or else we have

- tricky workarounds
- no real understanding of our models
- redesign every other year