Generating Filmstrip Models from Actor-Based Systems

Tony Clark[1] Vinay Kulkarni[2] Souvik Barat[2] Balbir Barn[3]

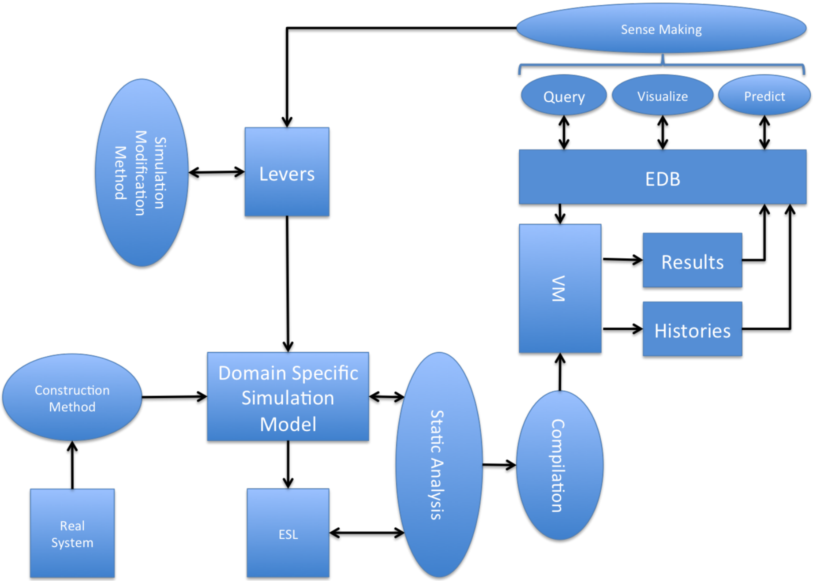
Sheffield Hallam University, UK

Tata Consultancy Services Research, India

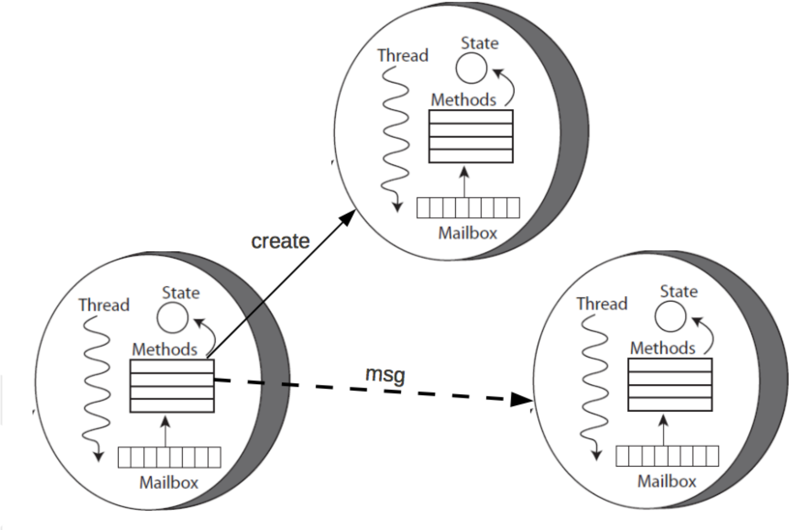
Middlesex University, UK

September 20, 2017

ESL Architecture



Actors


Thread
State
Methods
Mailbox
create
msg
Thread
State
Methods
Mailbox
Thread
State
Methods
Mailbox

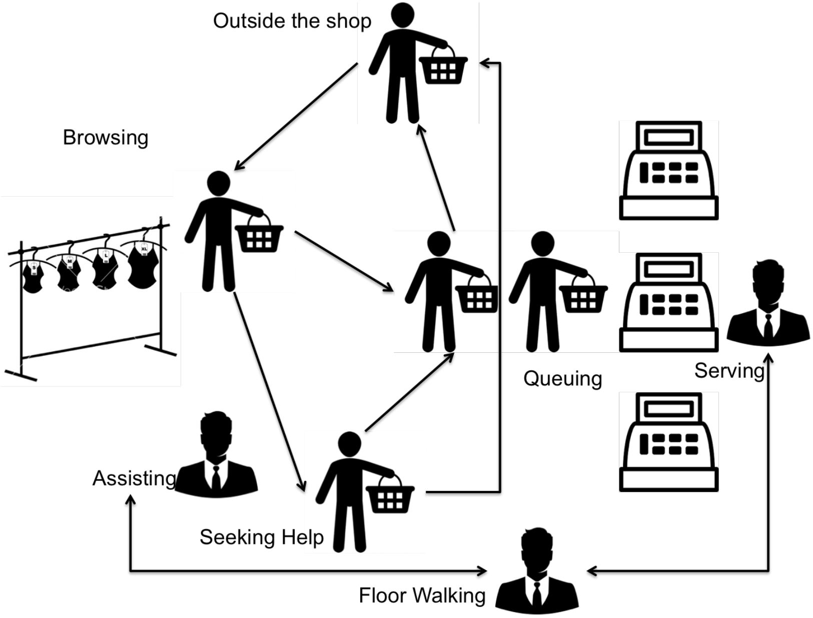
Hello World: Definition

```
export main;

type Main   = Act { Time(Int) };
type Worker = Act { Work() };

act worker(id::Int)::Worker {
  Work → {
    print[Str]('Hello World: ' + id);
    wait(random(10))
  }
};

range::[Int]        = 0..10;
workers::[Worker] = [new worker(i) | i::Int ← range];
limit::Int          = 100;

act main::Main {
  Time(n::Int) when n < limit → {
    for w::Worker in workers do {
      w ← Work
    }
  };
  Time(n::Int) → stopAll()
}
```

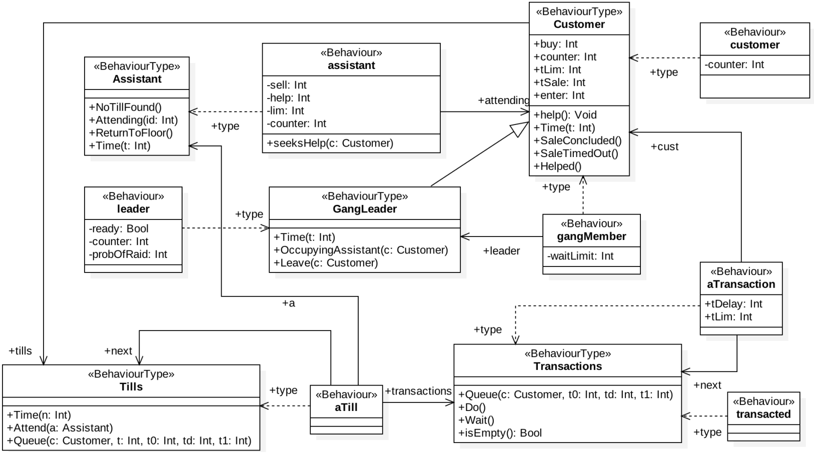
Hello World: Output

```
Hello World: 0
Hello World: 7
Hello World: 2
Hello World: 10
Hello World: 3
Hello World: 8
Hello World: 9
Hello World: 5
Hello World: 6
Hello World: 4
Hello World: 1
Hello World: 10
Hello World: 4
Hello World: 5
Hello World: 4
Hello World: 8
Hello World: 3
Hello World: 7
Hello World: 9
Hello World: 2
```

Shop


Outside the shop
Browsing
S
M
L
XL
Queuing
Serving
Assisting
Seeking Help
Floor Walking

Structure



«BehaviourType»
Customer
+buy: Int
+counter: Int
+tLim: Int
+tSale: Int
+enter: Int
+help(): Void
+Time(t: Int)
+SaleConcluded()
+SaleTimedOut()
+Helped()

«Behaviour»
customer
-counter: Int

«BehaviourType»
Assistant
+NoTillFound()
+Attending(id: Int)
+ReturnToFloor()
+Time(t: Int)

«Behaviour»
assistant
-sell: Int
-help: Int
-lim: Int
-counter: Int
+seeksHelp(c: Customer)

«Behaviour»
leader
-ready: Bool
-counter: Int
-probOfRaid: Int

«BehaviourType»
GangLeader
+Time(t: Int)
+OccupyingAssistant(c: Customer)
+Leave(c: Customer)

«Behaviour»
gangMember
-waitLimit: Int

«Behaviour»
aTransaction
+tDelay: Int
+tLim: Int

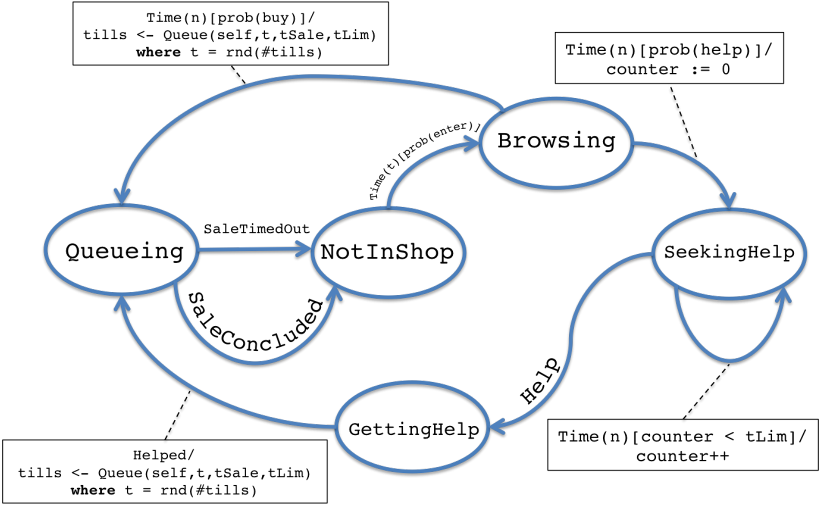
«BehaviourType»
Tills
+Time(n: Int)
+Attend(a: Assistant)
+Queue(c: Customer, t: Int, t0: Int, td: Int, t1: Int)

«Behaviour»
aTill

«BehaviourType»
Transactions
+Queue(c: Customer, t0: Int, td: Int, t1: Int)
+Do()
+Wait()
+isEmpty(): Bool

«Behaviour»
transacted

Behaviour



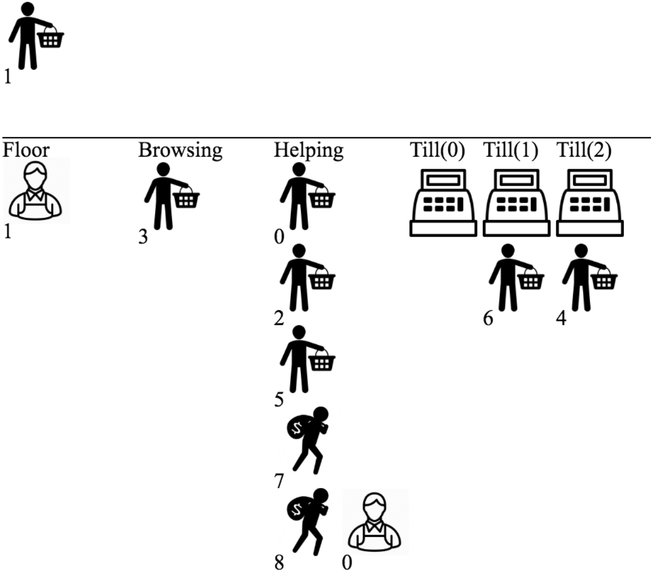
```
Time(n)[prob(buy)]/
tills <- Queue(self,t,tSale,tLim)
where t = rnd(#tills)
```

```
Time(n)[prob(help)]/
counter := 0
```

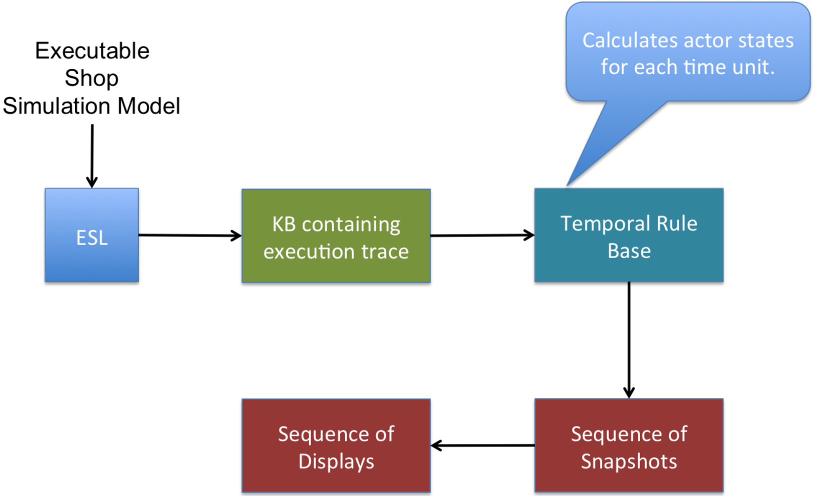
```
Helped/
tills <- Queue(self,t,tSale,tLim)
where t = rnd(#tills)
```

```
Time(n)[counter < tLim]/
counter++
```

Filmstrip Snapshots



Filmstrip Production



History Pattern

$$\begin{tikzcd}
{([E], i : E \to \mathrm{Int})} \arrow[r, "\sigma"] \arrow[d, "\omega"'] & {([[E]], i : E \to \mathrm{Int})} \arrow[d, "\mu^*"] \\
{[D]} & {[S]} \arrow[l, "\gamma^*"]
\end{tikzcd}$$

E	Events
$i : E \to \mathrm{Int}$	Event Owner
$([E], i : E \to \mathrm{Int})$	Event Histories
$[[E]]$	Snapshots
$(S, \oplus, \epsilon)$	Semantics Monoid
$\phi : M \to S$	Semantic Mapping
$\mu = \mathrm{foldr}(\phi, \oplus, \epsilon)$	Snapshot Mapping
D	Displays
$[D]$	Filmstrips