Located Demos2k - Towards a Tool for Modelling Processes and Distributed Resources

Matthew Collinson, Brian Monahan, David Pym
HP Laboratories
HPL-2008-76

Keyword(s):
Demos2k, location, resources, processes, systems modelling

Abstract:
We describe the background to, and the current state of the development of, Located Demos2k, an executable modelling language which reconstructs the Demos2k language starting from an explicit model of location. The version of Located Demos2k described herein is the first useful stage in its development, and provides convenient a point of departure of discussing its further development.

External Posting Date: June 21, 2008 [Fulltext] Approved for External Publication

Internal Posting Date: June 21, 2008 [Fulltext]

Located Demos2k

Towards a Tool for Modelling Processes and Distributed Resources

Matthew Collinson[1] Brian Monahan[2] David Pym[3]

Systems Security Lab, HP Laboratories, Bristol, UK

Abstract

We describe the background to, and the current state of the development of, Located Demos2k, an executable modelling language which reconstructs the Demos2k language starting from an explicit model of location. The version of Located Demos2k described herein is the first useful stage in its development, and provides convenient a point of departure of discussing its further development.

1 Introduction

We have begun an exploration of the application of modelling technology based upon Demos2000 [Dem] (henceforth Demos2k) into the domain of IT security and issues concerning economic and service-level trade-offs [GMP07, MP06, MY05, YMP06a, YMP06]. These examples and case studies show how managing IT security for large organizations is concerned with the distribution of resources over the organization and not merely with its allocation within the organization. IT security is inevitably concerned also with environmental factors such as the presence of vulnerabilities within third party software and the potential for disruption caused by exploits.

In this paper, we introduce a prototype that we call 'Located Demos2k' (or 'LD2k' for short) for a practical modelling language, whose mathematical semantics is under development using the Synchronous Calculus of Resource and Processes (SCRP), as introduced in [PT06, PT07, CPT07a] and recently consolidated and extended in [CP08a, CP08b] and, in particular, [CMP08, CMP08-TR].

The LD2k language will take Demos2k as its starting point, primarily extending Demos2k with a notion of located resource. We believe this notion will assist in developing the models of distributed resources that arise in IT security. The development of any well-honed, useful tool takes time and, in particular, experience of application. In this spirit, we anticipate that we will make enhancements and improvements to our design as our experience extends, mainly as a result of case studies being developed by our colleagues and ourselves in a range of applied modelling engagements. We anticipate further reports and papers as these case studies and mathematical developments unfold in due course.

The next section highlights the modelling philosophy and approach we have adopted, followed by a section describing Demos2k and how LD2k extends it. In § 4, we introduce a short example involving ferries and submarines (which is given in full in the Appendix) and, in § 5, we discuss what the challenges are (in the semantics) and briefly what we hope to produce in the coming months in terms of software support.

2 A systems modelling philosophy

We are concerned with modelling systems and their interaction with their environments at levels of abstraction that are appropriate for the properties of interest and do not represent unnecessary detail. To this end,

[1]Systems Security Lab, HP Labs, Bristol matthew.collinson@hp.com
[2]Systems Security Lab, HP Labs, Bristol brian.monahan@hp.com
[3]Systems Security Lab, HP Labs, Bristol and University of Bath david.pym@hp.com

based on both a basic consideration of the ontology of processes and a good deal of modelling experience, we take the view that we need mechanisms and structures within our models that represent the following [PT06, PT07, MP06, YMP06]:

Environment We consider that the environment is the source of events that are incident upon the system's (logical or spatial) boundary. The events considered include not only the intended interaction of the system with its environment, but also unintended interactions, such as security incidents. Mathematically, we consider these events to be represented stochastically;

Locations In general, a system is not confined to a single (logical or spatial) place. Rather, it is distributed over a collection of interconnected places. Mathematically, we consider a model of location to include set of *places*, a set of (directed) *connections* between places, a notion of *sub-location*, a notion of *substitution* (preserving connectivity), and (perhaps) a *product* of locations;

Resources Resources are the essentially passive components of the system, required for processes to be able to execute, and are manipulated by process execution. Resource elements can be combined and compared. Mathematically, we require resources to carry the structure of an ordered partial monoid;

Processes The processes that execute relative to the resources available around the distributed system describe the dynamics of the system and the service it provides. Mathematically, we describe processes as terms of a process algebra. More specifically, we use a synchronous calculus of processes which execute relative to available resources.

The mathematics of our systems modelling is described in [PT06, PT07, CPT07a, CP08a], in which we present a synchronous calculus (within which asynchrony can be represented) of resources and processes (SCRP), and in [PT07, CP08a], where we also discuss the basic idea of location. Properties of models described using SCRP can be described using its associated modal logic, MBI [PT06, PT07, CPT07a, CP08a, CP08b]. Our theory of location is developed more fully in [CMP08, CMP08-TR].

3 From Demos2k to Located Demos2k (LD2k)

As we have mentioned, our starting point is Demos2k [Dem], which is an executable modelling language, derived from the classic simulation language Demos [Bir79]. It is essentially a Concurrent Guarded Command style imperative language in which independent processes are launched and then communicate via shared resources (e.g., queues, semaphores). The underpinning semantics is synchronous (using SCCS [Mil83]) resulting in a timed process model. Demos2k models have been extensively applied to performance modelling of large-scale systems and services in commercial engagements [PTT$^+$06], and have been embodied in tools (freely downloadable from [Dem]).

Demos2k (partially) realizes our modelling philosophy to the following extent:

Environment Demos2k is essentially synchronous, so models can accurately represent timed stochastic behaviours, since the semantics permits an exact treatment of process timing. This allows a rigorous treatment of randomly distributed timings, according to well-known choices of probability distribution (e.g., geometric, negative exponential, Poisson, normal, and others);

Locations Demos2k focusses on aggregated performance-modelling abstractions such as queues and so on; it is harder to represent problem situations that involve the allocation of individually separated resources and how they are (re)distributed. In short, Demos2k most easily deals with aggregated resources, and adding location will widen the range of problems that can be treated directly.

Demos2k eschews structured objects of any kind — the only data supported are numbers and number arrays. This works well for performance modelling style models, but is less useful in cases where

the models need to embody both structural features and the actions that depend on them, such as IT security models;

Resources Demos2k supports resource abstractions such as *bins* (i.e., queues) for communication between processes. It also supports *shareable*, non-consumable resources where processes can lay claim to a number of such elements, use them in some way and then return those claimed before the process completes;

Processes Demos2k models represent activities consisting of interacting concurrent processes, where statements are executed sequentially within each process. Naturally, processes can launch other processes as needed. Synchronized call of one process by another (cf. Remote Procedure Call) is also supported.

Demos2k can manipulate queues of arbitrary size, and so is Turing complete. Denotational models of Demos2k, based on CCS and SCCS [Mil83] have been given in [BT01a, BT01b]. Denotational models of Demos2k in SCRP would give more direct representation of Demos2k's resources and their manipulation by entities.

Demos2k already provides a strong foundation for investigating performance-style models of systems and services, but is weakest perhaps when it comes to dealing with abstractions that necessarily involve the distribution and mobility of structured elements, such as systems security models. To counter this, we introduce a broad extension of Demos2k (called here Located Demos2k or LD2k), that involves the notion of location described in § 2, and which captures resource distribution and allocation properties. The appropriate theory for these abstractions is embedded inside an extension of SCRP here called Located SCRP (see [PT07, CMP08, CMP08-TR]).

The implementation that embodies our location abstractions involves re-developing and re-engineering the Demos2k language and its software support tools — the resulting system is called Located Demos2k and retains the essential features of Demos2k (i.e., no expressiveness is lost). Our extension amounts to introducing resources that are associated with particular locations. Once we can have resources at locations, it then makes sense to redistribute them to other locations dynamically within the model. Thus, we also introduce operations to *move* (claimed) resources from one location to another.

4 An Example: Docking Ferries and Submarines

We illustrate LD2k, as implemented so far, using a brief example concerning boats and the activity of *docking* them. In particular, we speak of the *secure docking* of submarines on the one hand and the standard docking of ferry boats on the other. This is a variation of the standard elementary Demos2k 'boats' example [Dem] which describes the activity of docking boats by using an appropriate number of tugs, unloading the contents of each docked boat and then dispatching them, eventually.

In our example, we have four locations: *OpenSea, Harbour, FerryDock*, and *SubmarineDock* which are connected as follows:

- *OpenSea* to *Harbour*, and *Harbour* to each of *FerryDock* and *SubmarineDock*.

The two docks do *not* interact directly. This is described in LD2k as follows:

```
location openSea, harbour, ferryDock, submarineDock;

connect openSea <-> harbour; connect harbour <-> ferryDock;
connect harbour <-> submarineDock;
```

As resources, we have *Tugs, SecureTugs, Ferries, Submarines* and *Jetties*. The Jetties are directly associated with the two docks — *FerryDock* and *SubmarineDock* — and so are fixed. These are described in LD2k as follows:

```
newR(tug@harbour, 3);       newR(secureTug@harbour, 2);
newR(ferry@openSea, 3);     newR(submarine@openSea, 2);
newR(jetty@ferryDock, 2);   newR(jetty@submarineDock, 3);
```

The activity of docking the ferries and submarines is represented by a couple of concurrent processes, *dockFerry*, describing the way that tugs are used to dock each ferry, and *dockSub*. Owing to space limitations, we only show a short fragment of dockSub:

```
process dockSub = {
  req[ getR(submarine@harbour, 1); getR(tug@harbour, 1); getR(secureTug@harbour, 1) ];
  moveR(submarine, harbour -> submarineDock, 1);    // move the ferry to the ferry dock, and
  moveR(secureTug, harbour -> submarineDock, 1);    // move the secureTug as well
  moveR(tug, harbour -> submarineDock, 1);          // move the tug as well
  getR(jetty@submarineDock, 1);                     // grab a jetty to dock at
  hold(dockingTime);                                // dock the submarine
  ...
  try [ getR(tug@harbour, 2) ] then {               // grab 2 tugs in the harbour
    ...
    putR(jetty@submarineDock, 1);                   // now release the jetty
    ...
  }
  etry [ getR(tug@harbour, 1); getR(secureTug@harbour, 1) ] then {
    ...
    putR(jetty@submarineDock, 1);                   // now release the jetty
    ... } ... }
```

The purpose of our model is to describe the activities, obtain illustrative traces and to gather overall statistics from repeated runs. Here is a very short extract from one example trace produced by our current software support tools:

```
  ...
  75 ! dockSub_11 Launching entity 'dockSub' (dockSub_14) at time 75
  78 ! genSubs_13 MoveR: moved resource 'submarine' amount 1 from location 'OpenSe ...
  78 ! genSubs_13 PutR: released 1 units for resource 'submarine' at location 'Har ...
  78 ! genSubs_13 Launching entity 'genSubs' (genSubs_14) at time 81
  78 ! dockSub_13 MoveR: moved resource 'submarine' amount 1 from location 'Harbou ...
  78 ! dockSub_13 MoveR: moved resource 'secureTug' amount 1 from location 'Harbou ...
  78 ! dockSub_13 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to  ...
  78 ! dockSub_13 GetR: claimed 1 units for resource 'jetty' at location 'Submarin ...
  78 ! dockSub_13 Hold issued for 3
  ...
```

5 Directions and challenges

The logic MBI can essentially provide a generalized account of Separation Logic [Rey02, CP08a] and Concurrent Separation Logic [O'H07, CP08a]. In these logics, location is treated as resource (via partial composition). Demos2k, however, does not provide for partial composition. MBI with locations, integrated with Located Demos2k is intended to provide a unified and generalized account of these ideas. Alternative views of location are given in [CG00, JM04, BG07, CGZ05], among others.

The theory of location in Located SCRP and its embodiment in Located Demos2k is under (very active) development. We are working towards supporting:

- Short term: Completion of basic software support tools (i.e., full syntax parsing, more stochastic simulation and statistical analysis tools);
- Near term: A fully developed notion of location, with the ability to 'recall' and 'forget' location substructure (to give substitutivity for abstraction and refinement);
- Longer term: Fully developed model checking, properly integrated with LD2k.

5.1 Semantic challenges posed by LD2k

As LD2k is substantially based upon the modeling language Demos2k, it naturally inherits a number of characteristics such as:

- the real-valued passage of time;
- imperative actions/effects for a guarded-command like language;
- shared variable; and
- both sequential and synchronously concurrent processes.

Ideally, a mathematical semantics of a modelling language like LD2k would naturally be expressed as a straightforward homomorphism from an appropriate term algebra representing the language itself into an algebra expressed in terms of SCRP. However, the particular mix of language features proposed for LD2k represents an interesting challenge to capture directly in this way.

Our purpose here is to mention briefly some of these challenges, and then to point towards how we hope to resolve them.

5.1.1 Sequencing and timed actions

Each LD2k process is internally sequential, consisting of both simple and compound statements. Simple statements for example consist of simple assignments and and 'holds' in which an explicit amount of (real-valued) time passes. Other processes can however be launched after an explicit (real-timed) time delay. Resources can be claimed and released. Compound statements include scoping block statements, looping constructs such as repeat and while and finally the try conditional statement.

The 'try' statement inherited from Demos2k is particularly useful since it effectively provides multi-way synchronisation of resource access and introduces both blocked waiting and non-blocking tests. Because we have both simple and compound statements in LD2k, this means that sequencing — the semi-colon operator — would not directly correspond to an action prefix (or even the asynchronous action prefix) in SCRP.

Now, each LD2k process moves forward in timed synchrony - that is, real-valued time passes simultaneously in each process. The upshot of this is that each statement may concurrently *consume* different amounts of time. We can illustrate this as follows:

```
process A = { hold(1.4); print("A - 1");
              hold(3.4); print("A - 2"); }

process B = { hold(0.8); print("B - 1"); hold(14.0);
              print("B - 2"); hold(2.0); }

launch("process-A", A, 1.0); // Launches instance of process
                             // A after 1.0 unit of time.
launch("process-B", B, 0.3); // Launches instance of process
                             // B after 0.3 units of time.

hold(37.4);
close;
```

Executing this fragment in LD2k results in an explicit timed trace:

```
 0.0   :  simulation begins
 0.3   :  "process-B-1" starts
 1.0   :  "process-A-1" starts
 1.1   :  "process-B-1" prints "B - 1"
 2.4   :  "process-A-1" prints "A - 1"
 5.8   :  "process-A-1" prints "A - 2"
 5.8   :  "process-A-1" exits
15.1   :  "process-B-1" prints "B - 2"
17.1   :  "process-B-1" exits
37.4   :  simulation closes
```

Notice that both the instances of process A and process B make progress, even though each action they perform happens at a different moment in time.

The form of synchrony covered here in SCRP assumes that each action is atomic and synchronization take place at the completion of each action. In the case of LD2k, we need to mark the flow of time even though some of the actions in each component process may not have completed.

5.1.2 Starting and stopping processes synchronously

Processes in LD2k can take different amounts of elapsed time to complete. Thus, LD2k processes can start and terminate at different times. Obviously this is awkward to do with a synchronised product that proceeds in lock-step. We need to be able to introduce and eliminate LD2k processes cleanly without unduely disrupting the execution of other, independent processes.

Fortunately, termination can be cleanly handled of in a translation of each LD2k process into SCRP so that the translated process always ends in 1 where:

$$1 \times P = P = P \times 1$$

Dually, starting a fresh process instance can exploit the same idea (but in reverse) to *introduce* process terms into a product. In practice, we may also need to have appropriate process creation and termination rules to expand and contract synchronous product terms.

5.1.3 Tracking ownership of sharable resources

Like its predecessor Demos2k, LD2k keeps track of how much sharable resource that each process has claimed. The idea is to guarantee that, when a process terminates, it has exactly returned all of this resource or to abort the simulation when that is not the case. This is a highly useful integrity property for LD2k processes to have automatically and systematically checked. This tracking of resource ownership is not naturally incorporated within SCRP, but it could be handled as part of a semantic translation from LD2k terms into SCRP terms. What is required here is suitable mechanism to maintain a per-process dynamic mapping of how much resource each process has claimed and to check at process termination that this mapping is empty.

5.1.4 Partiality of actions

The concept of location in SCRP naturally introduces more ways of expressing distinctions into models. Consequently, this means that LD2k can also introduce more ways of expressing and introducing distinctions. This also means, however, that functions, actions and decisions can depend upon such distinctions; in particular, they may also fail. For example, access-control decisions typically depend upon identity/role of the claimant, their location, and for what purpose the resource is being claimed.

The upshot here is that incorporating location into LD2k introduces the potential for partially defined actions. This is a strict extension of the Demos2k execution model — if a resource is not currently available in Demos2k, then it is assumed that it may be available at a later stage — there is no explicit *refusal* to provide resource in Demos2k.

This is of course different in LD2k and thus we need to represent how to manage such refusals when they arise. Incorporating this will require both syntactic and semantic extensions to the LD2k framework.

5.1.5 Priority and choice

As mentioned earlier, a significant feature of both Demos2k and LD2k is the ‘try’ statement. Here is an example:

```
try [ getR(ports@server42, 1), count < 100 ] with {

  ... do statements (block 1) ...

}
etry [ getR(fileShares@server67, 4) ] with {

  ... do other statements (block 2) ...

}
etry [ getR(account@client35, 4),  getR(writePrivilege@server56, 1)] with {

  ... do yet more statements (block 3) ...

}
```

The 'try' statement provides priority for the guarded alternatives. Essentially, each guard is tested and the alternative taken is the first that can be accepted. If none can be accepted immediately then the process blocks (i.e., waits) until there is an alternative that can be accepted. If there is more than one accepting alternative, then the earliest one is taken. A pleasing consequence of this approach is that by adding a 'default' alternative with empty acquisition condition (which can always be satisfied), we can obtain the traditional non-blocking form of conditional, allowing us to express immediate tests upon resource availability.

A simple-minded translation of 'try' from LD2k into process sum in SCRP will not work because of this need to encode priority. To do this, we need to incorporate 'weighting' into the underlying process calculus, probably building on Tofts' WSCCS [Tof94].

5.1.6 An approach to resolving these challenges

A promising approach to attacking these issues is to proceed on the following two fronts:

1. Subsume time as a particular kind of resource in SCRP, thus introducing time as a 'cost' (with the obvious monoid structure).

2. Assuming an explicit granularity of atomic action - thus, hold statements would consume time as integral multiples of some minimal time (e.g., 10**-9). This would (technically) allow LD2k processes to have timed synchrony, since each process would naturally decompose into sequences of atomic actions.

   Note that a side effect is a pleasing simplification of the semantics of sequencing; we can regain the use of action prefix to do this.

References

[BG07] N. Biri and D. Galmiche. Models and separation logics for resource trees. *Journal of Logic and Computation*, 17(4):687–726, 2007.

[Bir79] G. Birtwistle. *Demos — discrete event modelling on Simula.* Macmillan, 1979.

[BT01a] G. Birtwistle and C. Tofts. Getting Demos Models Right — Part I Practice. *Simulation Practice and Theory*, 8(6-7):377–393, 2001.

[BT01b] G. Birtwistle and C. Tofts. Getting Demos Models Right — Part II ... and Theory. *Simulation Practice and Theory*, 8(6-7):395–414, 2001.

[CGZ05] C. Calcagno, P. Gardner and U. Zarfaty. Context Logic and Tree Update. *Proc. POPL 05*. ACM.

[CG00] L. Cardelli and A. Gordon. Mobile ambients. *Theoret. Comp. Sci.*, (240):177–213, 2000.

[CP08a] M. Collinson and D. Pym. Algebra and logic for access control. Submitted: http://www.cs.bath.ac.uk/ pym/alacds.pdf, 2008. To be available as a Technical Report, HP Labs, 2008.

[CP08b] M. Collinson and D. Pym. Algebra and logic for resource-based systems modelling. Submitted: http://www.cs.bath.ac.uk/ pym/mbi.pdf, 2008.

[CMP08] M. Collinson, B. Monahan, and D. Pym. A Logical and Computational Theory of Located Resource. Submitted, 2008.

[CMP08-TR] M. Collinson, B. Monahan, and D. Pym. A Logical and Computational Theory of Located Resource. Forthcoming Technical Report, HP Labs, 2008.

[CPT07a] M. Collinson, D. Pym, and C. Tofts. Errata for Formal Aspects of Computing (2006) 18:495–517 and their consequences *Formal Aspects of Computing* 19(4):551–554, 2007.

[Dem] Demos2k. http://www.demos2k.org, 2000

[GMP07] J. Griffin, B. Monahan, D.J. Pym, M. Wonham, and M. Yearworth. Assessing the Value of Investments in Network Security Operations:A Systems Analytics Approach. HP Labs Technical Report, HPL-2007-89, 2007, http://library.hp.com/techpubs/2007/HPL-2007-89.pdf

[JM04] O. Jensen and R. Milner. Bigraphs and mobile processes (revised). Technical Report 580, Computer Laboratory, University of Cambridge, 2004.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoret. Comp. Sci.*, 25(3):267–310, 1983.

[MP06] B. Monahan and D.J. Pym. A Structural and Stochastic Modelling Philosophy for Systems Integrity. HP Labs Technical Report, HPL-2006-35, 2006 http://library.hp.com/techpubs/2006/HPL-2006-35.pdf

[MY05] B. Monahan, and M. Yearworth. Meaningful Security SLAs. HP Labs Technical Report, HPL-2005-218, 2005, http://library.hp.com/techpubs/2005/HPL-2005-218.pdf

[O'H07] P.W. O'Hearn. Resources, concurrency and local reasoning. *Theoret. Comp. Sci.*, 375(1–3):271–307, 2007.

[PT06] D. Pym and C. Tofts. A calculus and logic of resources and processes. *Formal Aspects of Computing*, 18(4):495–517, 2006. Erratum (with Collinson, M.) *Formal Aspects of Computing* (2007) 19: 551-554.

[PT07] D. Pym and C. Tofts. Systems Modelling via Resources and Processes: Philosophy, Calculus, Semantics, and Logic. In L. Cardelli, M. Fiore, and G. Winskel, editors, *Electronic Notes in Theoretical Computer Science (Computation, Meaning, and Logic: Articles dedicated to Gordon Plotkin)*, volume 107, pages 545–587, 2007. Erratum (with Collinson, M.) *Formal Aspects of Computing* (2007) 19: 551-554.

[PTT+06] D. Pym, R. Taylor, C. Tofts, M. Yearworth, and B. Monahan. Systems and services sciences: A rationale and a research agenda. Technical Report 112, Hewlett-Packard Laboratories, 2006.

[Rey02] J.C. Reynolds. Separation logic: a logic for shared mutable data structures. *Proc. LICS'02*, 55–74. IEEE.

[Tof94] C. Tofts. Processes with Probability, Priority and Time. *Formal Aspects of Computing* 6(5):536-564, 1994.

[YMP06a] M. Yearworth, B. Monahan, and D.J. Pym. Predictive Modelling for Meaningful Security SLAs. HP Labs Technical Report, HPL-2006-50, 2006
http://library.hp.com/techpubs/2006/HPL-2006-50.pdf

[YMP06] M. Yearworth, B. Monahan, and D. Pym. Predictive modelling for security operations economics (extended abstract). In *Proc. I3P Workshop on the Economics of Securing the Information Infrastructure*, 2006. Proceedings at http://wesii.econinfosec.org/workshop/.

A Full Example

Here is the full text of the example we described above in § 4.

We should point out that our current prototype LD2k tools do not as yet include a concrete syntax parser. However, we have hand-translated the text into an appropriate and convenient term language that can nevertheless be used to both check the well-formedness of the example and also animate it.

Please note that we do not as yet support stochastic behaviour; fortunately, this form of randomisation is easily added, given the architecture of our implementation so far.

```
(* Ferries and Submarines *)

location openSea, harbour, ferryDock, submarineDock;

connect openSea <-> harbour;
connect harbour <-> ferryDock;
connect harbour <-> submarineDock;

const delay  =  negexp(3);                const dockingTime  =  negexp(0.5);
const unloadingTime  =  normal(4, 0.7);   const loadingTime  =  unloadingTime;
const processingTime =  loadingTime;

newR(tug@harbour, 3);      newR(secureTug@harbour, 2);
newR(ferry@openSea, 3);    newR(submarine@openSea, 2);
newR(jetty@ferryDock, 2);  newR(jetty@submarineDock, 3);

process genFerries = {
  req [ getR(ferry@openSea, 1) ];
  hold(delay);
  moveR(ferry, openSea -> harbour, 1);
  putR(ferry@harbour, 1);                    // release the ferry
  launch(genFerries, delay)                  // relaunch this process
}

process genSubs = {
  req [ getR(submarine@openSea, 1) ];
  hold(delay);
  moveR(submarine, openSea -> harbour, 1);
  putR(submarine@harbour, 1);                // release the submarine
  launch(genSubs, delay)                     // relaunch this process
}

process dockFerry = {
  req[ getR(ferry@harbour, 1); getR(tug@harbour, 2) ];

  moveR(ferry, harbour -> ferryDock, 1);  // move the ferry to the ferry dock, and
  moveR(tug, harbour -> ferryDock, 2);    // move the two tugs as well

  getR(jetty@ferryDock, 1);               // grab a jetty to unload at

  hold(dockingTime);                      // dock the ferry
  moveR(tug, ferryDock -> harbour, 2);    // release tugs back to the harbour

  hold(unloadingTime);                    // unload the ferry
  hold(loadingTime);                      // load the ferry

  getR(tug@harbour, 2);                   // grab 2 tugs in the harbour
  moveR(tug, harbour -> ferryDock, 2);    // get the tugs to the ferryDock

  putR(jetty@ferryDock, 1);               // now release the jetty

  moveR(ferry, ferryDock -> harbour, 1);  // move the ferry to the harbour, and
```

```
  moveR(tug, ferryDock -> harbour, 2);    // move the tugs to the harbour

  putR(tug@harbour, 2);                   // release the tugs

  moveR(ferry, harbour -> openSea, 1);    // ferry goes out to openSea
  putR(ferry@openSea, 1);                 // release the ferry

  launch(dockFerry, 0)                    // relaunch this process
}


process dockSub = {
  req[ getR(submarine@harbour, 1); getR(tug@harbour, 1); getR(secureTug@harbour, 1) ];

  moveR(submarine, harbour -> submarineDock, 1);    // move the ferry to the ferry dock, and
  moveR(secureTug, harbour -> submarineDock, 1);    // move the secureTug as well
  moveR(tug, harbour -> submarineDock, 1);          // move the tug as well

  getR(jetty@submarineDock, 1);                     // grab a jetty to dock at

  hold(dockingTime);                                // dock the submarine

  moveR(tug, submarineDock -> harbour, 1);          // move tug back to the harbour
  putR(tug@harbour, 1)                              // release the tug

  moveR(secureTug, submarineDock -> harbour, 1);    // move secureTug back to the harbour
  putR(secureTug@harbour, 1)                        // release the secureTug

  hold(processingTime);                             // process the submarine

  try [ getR(tug@harbour, 2) ] then {               // grab 2 tugs in the harbour
    moveR(tug, harbour -> submarineDock, 2);        // get the tugs to the submarineDock

    putR(jetty@submarineDock, 1);                   // now release the jetty

    moveR(submarine, submarineDock -> harbour, 1);  // move the submarine to the harbour, and
    moveR(tug, submarineDock -> harbour, 2);        // move the tugs to the harbour

    putR(tug@harbour, 2);                           // release the tugs
  }
  etry [ getR(tug@harbour, 1); getR(secureTug@harbour, 1) ] then {

    moveR(secureTug, harbour -> submarineDock, 1);  // get the secure Tug to the submarineDock
    moveR(tug, harbour -> submarineDock, 1);        // get the tug to the submarineDock

    putR(jetty@submarineDock, 1);                   // now release the jetty

    moveR(submarine, submarineDock -> harbour, 1);  // move the submarine to the harbour, and
    moveR(secureTug, submarineDock -> harbour, 1);  // move the secure Tug to the harbour
    moveR(tug, submarineDock -> harbour, 1);        // move the tug to the harbour

    putR(secureTug@harbour, 1);                     // release the secureTug
    putR(tug@harbour, 1);                           // release the tug
  }

  moveR(submarine, harbour -> openSea, 1);          // submarine goes out to openSea
  putR(submarine@openSea, 1);                       // release the submarine

  launch(dockSub, 0)                                // relaunch this process
}
```

```
launch(genFerries, 0);
launch(genSubs, 0);

do 3 { launch(dockFerry, 0); launch(dockSub, 0); }

hold(80);   // length of simulation
close;        // close the simulation
```

A.1 Detailed Trace

Here is the detailed trace produced by the current support tool from which we extracted the trace fragment in the paper:

```
Example concur 08 example - Ferries and Subs:
[
  NewL("OpenSea");
  NewL("Harbour");
  NewL("FerryDock");
  NewL("SubmarineDock");
  Link("OpenSea","Harbour");
  Link("Harbour","OpenSea");
  Link("Harbour","FerryDock");
  Link("FerryDock","Harbour");
  Link("Harbour","SubmarineDock");
  Link("SubmarineDock","Harbour");
  NewR("jetty","FerryDock",3);
  NewR("jetty","SubmarineDock",2);
  NewR("tug","Harbour",5);
  NewR("secureTug","Harbour",3);
  NewR("ferry","OpenSea",4);
  NewR("submarine","OpenSea",6);
  DefnClass "genFerries"
  [
    GetR("ferry","OpenSea",1);
    Hold(2);
    MoveR("ferry","OpenSea","Harbour",1);
    PutR("ferry","Harbour",1);
    Entity("genFerries","genFerries",1)
  ];
  DefnClass "genSubs"
  [
    GetR("submarine","OpenSea",1);
    Hold(3);
    MoveR("submarine","OpenSea","Harbour",1);
    PutR("submarine","Harbour",1);
    Entity("genSubs","genSubs",1)
  ];
  DefnClass "dockFerry"
  [
    Try [ GetR("ferry","Harbour",1); GetR("tug","Harbour",2) ] Then:
      [

      ];
    MoveR("ferry","Harbour","FerryDock",1);
    MoveR("tug","Harbour","FerryDock",2);
    GetR("jetty","FerryDock",1);
    Hold(3);
    MoveR("tug","FerryDock","Harbour",2);
    Hold(4);
    Hold(5);
    GetR("tug","Harbour",2);
    MoveR("tug","Harbour","FerryDock",2);
    PutR("jetty","FerryDock",1);
    MoveR("ferry","FerryDock","Harbour",1);
    MoveR("tug","FerryDock","Harbour",2);
    PutR("tug","Harbour",1);
    MoveR("ferry","Harbour","OpenSea",1);
    PutR("ferry","OpenSea",1);
    Entity("dockFerry","dockFerry",1)
  ];
  DefnClass "dockSub"
  [
    Try [ GetR("submarine","Harbour",1); GetR("secureTug","Harbour",1); GetR("tug","Harbour",1) ] Then:
      [

      ];
    MoveR("submarine","Harbour","SubmarineDock",1);
    MoveR("secureTug","Harbour","SubmarineDock",1);
    MoveR("tug","Harbour","SubmarineDock",1);
    GetR("jetty","SubmarineDock",1);
    Hold(3);
    MoveR("tug","SubmarineDock","Harbour",1);
    PutR("tug","Harbour",1);
    MoveR("secureTug","SubmarineDock","Harbour",1);
    PutR("secureTug","Harbour",1);
    Hold(6);
    Try [ GetR("tug","Harbour",2) ] Then:
      [
        MoveR("tug","Harbour","SubmarineDock",2);
        PutR("jetty","SubmarineDock",1);
        MoveR("submarine","SubmarineDock","Harbour",1);
        MoveR("tug","SubmarineDock","Harbour",2);
        PutR("tug","Harbour",2)
      ];
    Else Try [ GetR("tug","Harbour",1); GetR("secureTug","Harbour",1) ] Then:
      [
        MoveR("secureTug","Harbour","SubmarineDock",1);
        MoveR("tug","Harbour","SubmarineDock",1);
        PutR("jetty","SubmarineDock",1);
        MoveR("submarine","SubmarineDock","Harbour",1);
        MoveR("secureTug","SubmarineDock","Harbour",1);
        MoveR("tug","SubmarineDock","Harbour",1);
        PutR("secureTug","Harbour",1);
        PutR("tug","Harbour",1)
```

```
      ];
    MoveR("submarine","Harbour","OpenSea",1);
    PutR("submarine","OpenSea",1);
    Entity("dockSub","dockSub",1)
  ];
  Entity("genFerries","genFerries",1);
  Entity("genSubs","genSubs",1);
  Entity("dockFerry","dockFerry",1);
  Entity("dockSub","dockSub",1);
  Entity("dockFerry","dockFerry",1);
  Entity("dockSub","dockSub",1);
  Entity("dockFerry","dockFerry",1);
  Entity("dockSub","dockSub",1);
  Hold(80);
  Close
]

Execution Trace:
0 ! *MAIN* NewL: created new location 'OpenSea'
0 ! *MAIN* NewL: created new location 'Harbour'
0 ! *MAIN* NewL: created new location 'FerryDock'
0 ! *MAIN* NewL: created new location 'SubmarineDock'
0 ! *MAIN* Link: created link : 'OpenSea' --> 'Harbour'
0 ! *MAIN* Link: created link : 'Harbour' --> 'OpenSea'
0 ! *MAIN* Link: created link : 'Harbour' --> 'FerryDock'
0 ! *MAIN* Link: created link : 'FerryDock' --> 'Harbour'
0 ! *MAIN* Link: created link : 'Harbour' --> 'SubmarineDock'
0 ! *MAIN* Link: created link : 'SubmarineDock' --> 'Harbour'
0 ! *MAIN* NewR: created resource 'jetty' with 3 at location 'FerryDock'
0 ! *MAIN* NewR: created resource 'jetty' with 2 at location 'SubmarineDock'
0 ! *MAIN* NewR: created resource 'tug' with 5 at location 'Harbour'
0 ! *MAIN* NewR: created resource 'secureTug' with 3 at location 'Harbour'
0 ! *MAIN* NewR: created resource 'ferry' with 4 at location 'OpenSea'
0 ! *MAIN* NewR: created resource 'submarine' with 6 at location 'OpenSea'
0 ! *MAIN* Defined class 'genFerries'
0 ! *MAIN* Defined class 'genSubs'
0 ! *MAIN* Defined class 'dockFerry'
0 ! *MAIN* Defined class 'dockSub'
0 ! *MAIN* Launching entity 'genFerries' (genFerries_1) at time 2
0 ! *MAIN* Launching entity 'genSubs' (genSubs_1) at time 3
0 ! *MAIN* Launching entity 'dockFerry' (dockFerry_1) at time 0
0 ! *MAIN* Launching entity 'dockSub' (dockSub_1) at time 0
0 ! *MAIN* Launching entity 'dockFerry' (dockFerry_2) at time 0
0 ! *MAIN* Launching entity 'dockSub' (dockSub_2) at time 0
0 ! *MAIN* Launching entity 'dockFerry' (dockFerry_3) at time 0
0 ! *MAIN* Launching entity 'dockSub' (dockSub_3) at time 0
0 ! *MAIN* Hold issued for 80
2 ! genFerries_1 GetR: claimed 1 units for resource 'ferry' at location 'OpenSea'
2 ! genFerries_1 Hold issued for 2
3 ! genSubs_1 GetR: claimed 1 units for resource 'submarine' at location 'OpenSea'
3 ! genSubs_1 Hold issued for 3
4 ! genFerries_1 MoveR: moved resource 'ferry' amount 1 from location 'OpenSea' to location 'Harbour'
4 ! genFerries_1 PutR: released 1 units for resource 'ferry' at location 'Harbour'
4 ! genFerries_1 Launching entity 'genFerries' (genFerries_2) at time 6
4 ! dockFerry_1 MoveR: moved resource 'ferry' amount 1 from location 'Harbour' to location 'FerryDock'
4 ! dockFerry_1 MoveR: moved resource 'tug' amount 2 from location 'Harbour' to location 'FerryDock'
4 ! dockFerry_1 GetR: claimed 1 units for resource 'jetty' at location 'FerryDock'
4 ! dockFerry_1 Hold issued for 3
6 ! genSubs_1 MoveR: moved resource 'submarine' amount 1 from location 'OpenSea' to location 'Harbour'
6 ! genSubs_1 PutR: released 1 units for resource 'submarine' at location 'Harbour'
6 ! genSubs_1 Launching entity 'genSubs' (genSubs_2) at time 9
6 ! genFerries_2 GetR: claimed 1 units for resource 'ferry' at location 'OpenSea'
6 ! genFerries_2 Hold issued for 2
6 ! dockSub_1 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'SubmarineDock'
6 ! dockSub_1 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
6 ! dockSub_1 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
6 ! dockSub_1 GetR: claimed 1 units for resource 'jetty' at location 'SubmarineDock'
6 ! dockSub_1 Hold issued for 3
7 ! dockFerry_1 MoveR: moved resource 'tug' amount 2 from location 'FerryDock' to location 'Harbour'
7 ! dockFerry_1 Hold issued for 4
8 ! genFerries_2 MoveR: moved resource 'ferry' amount 1 from location 'OpenSea' to location 'Harbour'
8 ! genFerries_2 PutR: released 1 units for resource 'ferry' at location 'Harbour'
8 ! genFerries_2 Launching entity 'genFerries' (genFerries_3) at time 10
8 ! dockFerry_2 MoveR: moved resource 'ferry' amount 1 from location 'Harbour' to location 'FerryDock'
8 ! dockFerry_2 MoveR: moved resource 'tug' amount 2 from location 'Harbour' to location 'FerryDock'
8 ! dockFerry_2 GetR: claimed 1 units for resource 'jetty' at location 'FerryDock'
8 ! dockFerry_2 Hold issued for 3
9 ! genSubs_2 GetR: claimed 1 units for resource 'submarine' at location 'OpenSea'
9 ! genSubs_2 Hold issued for 3
9 ! dockSub_1 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
9 ! dockSub_1 PutR: released 1 units for resource 'tug' at location 'Harbour'
9 ! dockSub_1 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
9 ! dockSub_1 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
9 ! dockSub_1 Hold issued for 6
10 ! genFerries_3 GetR: claimed 1 units for resource 'ferry' at location 'OpenSea'
10 ! genFerries_3 Hold issued for 2
11 ! dockFerry_1 Hold issued for 5
11 ! dockFerry_2 MoveR: moved resource 'tug' amount 2 from location 'FerryDock' to location 'Harbour'
11 ! dockFerry_2 Hold issued for 4
12 ! genSubs_2 MoveR: moved resource 'submarine' amount 1 from location 'OpenSea' to location 'Harbour'
12 ! genSubs_2 PutR: released 1 units for resource 'submarine' at location 'Harbour'
12 ! genSubs_2 Launching entity 'genSubs' (genSubs_3) at time 15
12 ! genFerries_3 MoveR: moved resource 'ferry' amount 1 from location 'OpenSea' to location 'Harbour'
12 ! genFerries_3 PutR: released 1 units for resource 'ferry' at location 'Harbour'
12 ! genFerries_3 Launching entity 'genFerries' (genFerries_4) at time 14
12 ! dockSub_2 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'SubmarineDock'
12 ! dockSub_2 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
12 ! dockSub_2 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
12 ! dockSub_2 GetR: claimed 1 units for resource 'jetty' at location 'SubmarineDock'
```

```
12 ! dockSub_2 Hold issued for 3
14 ! genFerries_4 GetR: claimed 1 units for resource 'ferry' at location 'OpenSea'
14 ! genFerries_4 Hold issued for 2
15 ! dockFerry_2 Hold issued for 5
15 ! genSubs_3 GetR: claimed 1 units for resource 'submarine' at location 'OpenSea'
15 ! genSubs_3 Hold issued for 3
15 ! dockSub_2 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
15 ! dockSub_2 PutR: released 1 units for resource 'tug' at location 'Harbour'
15 ! dockSub_2 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
15 ! dockSub_2 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
15 ! dockSub_2 Hold issued for 6
15 ! dockSub_1 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
15 ! dockSub_1 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
15 ! dockSub_1 PutR: released 1 units for resource 'jetty' at location 'SubmarineDock'
15 ! dockSub_1 MoveR: moved resource 'submarine' amount 1 from location 'SubmarineDock' to location 'Harbour'
15 ! dockSub_1 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
15 ! dockSub_1 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
15 ! dockSub_1 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
15 ! dockSub_1 PutR: released 1 units for resource 'tug' at location 'Harbour'
15 ! dockSub_1 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'OpenSea'
15 ! dockSub_1 PutR: released 1 units for resource 'submarine' at location 'OpenSea'
15 ! dockSub_1 Launching entity 'dockSub' (dockSub_4) at time 15
16 ! genFerries_4 MoveR: moved resource 'ferry' amount 1 from location 'OpenSea' to location 'Harbour'
16 ! genFerries_4 PutR: released 1 units for resource 'ferry' at location 'Harbour'
16 ! genFerries_4 Launching entity 'genFerries' (genFerries_5) at time 18
18 ! genSubs_3 MoveR: moved resource 'submarine' amount 1 from location 'OpenSea' to location 'Harbour'
18 ! genSubs_3 PutR: released 1 units for resource 'submarine' at location 'Harbour'
18 ! genSubs_3 Launching entity 'genSubs' (genSubs_4) at time 21
18 ! dockSub_3 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'SubmarineDock'
18 ! dockSub_3 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
18 ! dockSub_3 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
18 ! dockSub_3 GetR: claimed 1 units for resource 'jetty' at location 'SubmarineDock'
18 ! dockSub_3 Hold issued for 3
21 ! genSubs_4 GetR: claimed 1 units for resource 'submarine' at location 'OpenSea'
21 ! genSubs_4 Hold issued for 3
21 ! dockSub_3 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
21 ! dockSub_3 PutR: released 1 units for resource 'tug' at location 'Harbour'
21 ! dockSub_3 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
21 ! dockSub_3 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
21 ! dockSub_3 Hold issued for 6
21 ! dockSub_2 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
21 ! dockSub_2 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
21 ! dockSub_2 PutR: released 1 units for resource 'jetty' at location 'SubmarineDock'
21 ! dockSub_2 MoveR: moved resource 'submarine' amount 1 from location 'SubmarineDock' to location 'Harbour'
21 ! dockSub_2 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
21 ! dockSub_2 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
21 ! dockSub_2 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
21 ! dockSub_2 PutR: released 1 units for resource 'tug' at location 'Harbour'
21 ! dockSub_2 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'OpenSea'
21 ! dockSub_2 PutR: released 1 units for resource 'submarine' at location 'OpenSea'
21 ! dockSub_2 Launching entity 'dockSub' (dockSub_5) at time 21
24 ! genSubs_4 MoveR: moved resource 'submarine' amount 1 from location 'OpenSea' to location 'Harbour'
24 ! genSubs_4 PutR: released 1 units for resource 'submarine' at location 'Harbour'
24 ! genSubs_4 Launching entity 'genSubs' (genSubs_5) at time 27
24 ! dockSub_4 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'SubmarineDock'
24 ! dockSub_4 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
24 ! dockSub_4 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
24 ! dockSub_4 GetR: claimed 1 units for resource 'jetty' at location 'SubmarineDock'
24 ! dockSub_4 Hold issued for 3
27 ! genSubs_5 GetR: claimed 1 units for resource 'submarine' at location 'OpenSea'
27 ! genSubs_5 Hold issued for 3
27 ! dockSub_4 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
27 ! dockSub_4 PutR: released 1 units for resource 'tug' at location 'Harbour'
27 ! dockSub_4 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
27 ! dockSub_4 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
27 ! dockSub_4 Hold issued for 6
27 ! dockSub_3 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
27 ! dockSub_3 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
27 ! dockSub_3 PutR: released 1 units for resource 'jetty' at location 'SubmarineDock'
27 ! dockSub_3 MoveR: moved resource 'submarine' amount 1 from location 'SubmarineDock' to location 'Harbour'
27 ! dockSub_3 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
27 ! dockSub_3 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
27 ! dockSub_3 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
27 ! dockSub_3 PutR: released 1 units for resource 'tug' at location 'Harbour'
27 ! dockSub_3 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'OpenSea'
27 ! dockSub_3 PutR: released 1 units for resource 'submarine' at location 'OpenSea'
27 ! dockSub_3 Launching entity 'dockSub' (dockSub_6) at time 27
30 ! genSubs_5 MoveR: moved resource 'submarine' amount 1 from location 'OpenSea' to location 'Harbour'
30 ! genSubs_5 PutR: released 1 units for resource 'submarine' at location 'Harbour'
30 ! genSubs_5 Launching entity 'genSubs' (genSubs_6) at time 33
30 ! dockSub_5 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'SubmarineDock'
30 ! dockSub_5 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
30 ! dockSub_5 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
30 ! dockSub_5 GetR: claimed 1 units for resource 'jetty' at location 'SubmarineDock'
30 ! dockSub_5 Hold issued for 3
33 ! genSubs_6 GetR: claimed 1 units for resource 'submarine' at location 'OpenSea'
33 ! genSubs_6 Hold issued for 3
33 ! dockSub_5 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
33 ! dockSub_5 PutR: released 1 units for resource 'tug' at location 'Harbour'
33 ! dockSub_5 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
33 ! dockSub_5 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
33 ! dockSub_5 Hold issued for 6
33 ! dockSub_4 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
33 ! dockSub_4 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
33 ! dockSub_4 PutR: released 1 units for resource 'jetty' at location 'SubmarineDock'
33 ! dockSub_4 MoveR: moved resource 'submarine' amount 1 from location 'SubmarineDock' to location 'Harbour'
33 ! dockSub_4 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
33 ! dockSub_4 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
33 ! dockSub_4 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
```

```
33 ! dockSub_4 PutR: released 1 units for resource 'tug' at location 'Harbour'
33 ! dockSub_4 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'OpenSea'
33 ! dockSub_4 PutR: released 1 units for resource 'submarine' at location 'OpenSea'
33 ! dockSub_4 Launching entity 'dockSub' (dockSub_7) at time 33
36 ! genSubs_6 MoveR: moved resource 'submarine' amount 1 from location 'OpenSea' to location 'Harbour'
36 ! genSubs_6 PutR: released 1 units for resource 'submarine' at location 'Harbour'
36 ! genSubs_6 Launching entity 'genSubs' (genSubs_7) at time 39
36 ! dockSub_6 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'SubmarineDock'
36 ! dockSub_6 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
36 ! dockSub_6 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
36 ! dockSub_6 GetR: claimed 1 units for resource 'jetty' at location 'SubmarineDock'
36 ! dockSub_6 Hold issued for 3
39 ! genSubs_7 GetR: claimed 1 units for resource 'submarine' at location 'OpenSea'
39 ! genSubs_7 Hold issued for 3
39 ! dockSub_6 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
39 ! dockSub_6 PutR: released 1 units for resource 'tug' at location 'Harbour'
39 ! dockSub_6 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
39 ! dockSub_6 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
39 ! dockSub_6 Hold issued for 6
39 ! dockSub_5 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
39 ! dockSub_5 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
39 ! dockSub_5 PutR: released 1 units for resource 'jetty' at location 'SubmarineDock'
39 ! dockSub_5 MoveR: moved resource 'submarine' amount 1 from location 'SubmarineDock' to location 'Harbour'
39 ! dockSub_5 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
39 ! dockSub_5 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
39 ! dockSub_5 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
39 ! dockSub_5 PutR: released 1 units for resource 'tug' at location 'Harbour'
39 ! dockSub_5 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'OpenSea'
39 ! dockSub_5 PutR: released 1 units for resource 'submarine' at location 'OpenSea'
39 ! dockSub_5 Launching entity 'dockSub' (dockSub_8) at time 39
42 ! genSubs_7 MoveR: moved resource 'submarine' amount 1 from location 'OpenSea' to location 'Harbour'
42 ! genSubs_7 PutR: released 1 units for resource 'submarine' at location 'Harbour'
42 ! genSubs_7 Launching entity 'genSubs' (genSubs_8) at time 45
42 ! dockSub_7 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'SubmarineDock'
42 ! dockSub_7 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
42 ! dockSub_7 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
42 ! dockSub_7 GetR: claimed 1 units for resource 'jetty' at location 'SubmarineDock'
42 ! dockSub_7 Hold issued for 3
45 ! genSubs_8 GetR: claimed 1 units for resource 'submarine' at location 'OpenSea'
45 ! genSubs_8 Hold issued for 3
45 ! dockSub_7 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
45 ! dockSub_7 PutR: released 1 units for resource 'tug' at location 'Harbour'
45 ! dockSub_7 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
45 ! dockSub_7 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
45 ! dockSub_7 Hold issued for 6
45 ! dockSub_6 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
45 ! dockSub_6 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
45 ! dockSub_6 PutR: released 1 units for resource 'jetty' at location 'SubmarineDock'
45 ! dockSub_6 MoveR: moved resource 'submarine' amount 1 from location 'SubmarineDock' to location 'Harbour'
45 ! dockSub_6 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
45 ! dockSub_6 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
45 ! dockSub_6 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
45 ! dockSub_6 PutR: released 1 units for resource 'tug' at location 'Harbour'
45 ! dockSub_6 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'OpenSea'
45 ! dockSub_6 PutR: released 1 units for resource 'submarine' at location 'OpenSea'
45 ! dockSub_6 Launching entity 'dockSub' (dockSub_9) at time 45
48 ! genSubs_8 MoveR: moved resource 'submarine' amount 1 from location 'OpenSea' to location 'Harbour'
48 ! genSubs_8 PutR: released 1 units for resource 'submarine' at location 'Harbour'
48 ! genSubs_8 Launching entity 'genSubs' (genSubs_9) at time 51
48 ! dockSub_8 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'SubmarineDock'
48 ! dockSub_8 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
48 ! dockSub_8 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
48 ! dockSub_8 GetR: claimed 1 units for resource 'jetty' at location 'SubmarineDock'
48 ! dockSub_8 Hold issued for 3
51 ! genSubs_9 GetR: claimed 1 units for resource 'submarine' at location 'OpenSea'
51 ! genSubs_9 Hold issued for 3
51 ! dockSub_8 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
51 ! dockSub_8 PutR: released 1 units for resource 'tug' at location 'Harbour'
51 ! dockSub_8 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
51 ! dockSub_8 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
51 ! dockSub_8 Hold issued for 6
51 ! dockSub_7 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
51 ! dockSub_7 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
51 ! dockSub_7 PutR: released 1 units for resource 'jetty' at location 'SubmarineDock'
51 ! dockSub_7 MoveR: moved resource 'submarine' amount 1 from location 'SubmarineDock' to location 'Harbour'
51 ! dockSub_7 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
51 ! dockSub_7 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
51 ! dockSub_7 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
51 ! dockSub_7 PutR: released 1 units for resource 'tug' at location 'Harbour'
51 ! dockSub_7 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'OpenSea'
51 ! dockSub_7 PutR: released 1 units for resource 'submarine' at location 'OpenSea'
51 ! dockSub_7 Launching entity 'dockSub' (dockSub_10) at time 51
54 ! genSubs_9 MoveR: moved resource 'submarine' amount 1 from location 'OpenSea' to location 'Harbour'
54 ! genSubs_9 PutR: released 1 units for resource 'submarine' at location 'Harbour'
54 ! genSubs_9 Launching entity 'genSubs' (genSubs_10) at time 57
54 ! dockSub_9 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'SubmarineDock'
54 ! dockSub_9 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
54 ! dockSub_9 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
54 ! dockSub_9 GetR: claimed 1 units for resource 'jetty' at location 'SubmarineDock'
54 ! dockSub_9 Hold issued for 3
57 ! genSubs_10 GetR: claimed 1 units for resource 'submarine' at location 'OpenSea'
57 ! genSubs_10 Hold issued for 3
57 ! dockSub_9 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
57 ! dockSub_9 PutR: released 1 units for resource 'tug' at location 'Harbour'
57 ! dockSub_9 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
57 ! dockSub_9 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
57 ! dockSub_9 Hold issued for 6
57 ! dockSub_8 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
57 ! dockSub_8 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
```

```
  57 ! dockSub_8 PutR: released 1 units for resource 'jetty' at location 'SubmarineDock'
  57 ! dockSub_8 MoveR: moved resource 'submarine' amount 1 from location 'SubmarineDock' to location 'Harbour'
  57 ! dockSub_8 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  57 ! dockSub_8 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  57 ! dockSub_8 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
  57 ! dockSub_8 PutR: released 1 units for resource 'tug' at location 'Harbour'
  57 ! dockSub_8 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'OpenSea'
  57 ! dockSub_8 PutR: released 1 units for resource 'submarine' at location 'OpenSea'
  57 ! dockSub_8 Launching entity 'dockSub' (dockSub_11) at time 57
  60 ! genSubs_10 MoveR: moved resource 'submarine' amount 1 from location 'OpenSea' to location 'Harbour'
  60 ! genSubs_10 PutR: released 1 units for resource 'submarine' at location 'Harbour'
  60 ! genSubs_10 Launching entity 'genSubs' (genSubs_11) at time 63
  60 ! dockSub_10 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'SubmarineDock'
  60 ! dockSub_10 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  60 ! dockSub_10 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  60 ! dockSub_10 GetR: claimed 1 units for resource 'jetty' at location 'SubmarineDock'
  60 ! dockSub_10 Hold issued for 3
  63 ! genSubs_11 GetR: claimed 1 units for resource 'submarine' at location 'OpenSea'
  63 ! genSubs_11 Hold issued for 3
  63 ! dockSub_10 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  63 ! dockSub_10 PutR: released 1 units for resource 'tug' at location 'Harbour'
  63 ! dockSub_10 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  63 ! dockSub_10 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
  63 ! dockSub_10 Hold issued for 6
  63 ! dockSub_9 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  63 ! dockSub_9 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  63 ! dockSub_9 PutR: released 1 units for resource 'jetty' at location 'SubmarineDock'
  63 ! dockSub_9 MoveR: moved resource 'submarine' amount 1 from location 'SubmarineDock' to location 'Harbour'
  63 ! dockSub_9 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  63 ! dockSub_9 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  63 ! dockSub_9 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
  63 ! dockSub_9 PutR: released 1 units for resource 'tug' at location 'Harbour'
  63 ! dockSub_9 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'OpenSea'
  63 ! dockSub_9 PutR: released 1 units for resource 'submarine' at location 'OpenSea'
  63 ! dockSub_9 Launching entity 'dockSub' (dockSub_12) at time 63
  66 ! genSubs_11 MoveR: moved resource 'submarine' amount 1 from location 'OpenSea' to location 'Harbour'
  66 ! genSubs_11 PutR: released 1 units for resource 'submarine' at location 'Harbour'
  66 ! genSubs_11 Launching entity 'genSubs' (genSubs_12) at time 69
  66 ! dockSub_11 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'SubmarineDock'
  66 ! dockSub_11 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  66 ! dockSub_11 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  66 ! dockSub_11 GetR: claimed 1 units for resource 'jetty' at location 'SubmarineDock'
  66 ! dockSub_11 Hold issued for 3
  69 ! genSubs_12 GetR: claimed 1 units for resource 'submarine' at location 'OpenSea'
  69 ! genSubs_12 Hold issued for 3
  69 ! dockSub_11 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  69 ! dockSub_11 PutR: released 1 units for resource 'tug' at location 'Harbour'
  69 ! dockSub_11 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  69 ! dockSub_11 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
  69 ! dockSub_11 Hold issued for 6
  69 ! dockSub_10 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  69 ! dockSub_10 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  69 ! dockSub_10 PutR: released 1 units for resource 'jetty' at location 'SubmarineDock'
  69 ! dockSub_10 MoveR: moved resource 'submarine' amount 1 from location 'SubmarineDock' to location 'Harbour'
  69 ! dockSub_10 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  69 ! dockSub_10 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  69 ! dockSub_10 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
  69 ! dockSub_10 PutR: released 1 units for resource 'tug' at location 'Harbour'
  69 ! dockSub_10 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'OpenSea'
  69 ! dockSub_10 PutR: released 1 units for resource 'submarine' at location 'OpenSea'
  69 ! dockSub_10 Launching entity 'dockSub' (dockSub_13) at time 69
  72 ! genSubs_12 MoveR: moved resource 'submarine' amount 1 from location 'OpenSea' to location 'Harbour'
  72 ! genSubs_12 PutR: released 1 units for resource 'submarine' at location 'Harbour'
  72 ! genSubs_12 Launching entity 'genSubs' (genSubs_13) at time 75
  72 ! dockSub_12 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'SubmarineDock'
  72 ! dockSub_12 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  72 ! dockSub_12 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  72 ! dockSub_12 GetR: claimed 1 units for resource 'jetty' at location 'SubmarineDock'
  72 ! dockSub_12 Hold issued for 3
  75 ! genSubs_13 GetR: claimed 1 units for resource 'submarine' at location 'OpenSea'
  75 ! genSubs_13 Hold issued for 3
  75 ! dockSub_12 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  75 ! dockSub_12 PutR: released 1 units for resource 'tug' at location 'Harbour'
  75 ! dockSub_12 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  75 ! dockSub_12 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
  75 ! dockSub_12 Hold issued for 6
  75 ! dockSub_11 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  75 ! dockSub_11 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  75 ! dockSub_11 PutR: released 1 units for resource 'jetty' at location 'SubmarineDock'
  75 ! dockSub_11 MoveR: moved resource 'submarine' amount 1 from location 'SubmarineDock' to location 'Harbour'
  75 ! dockSub_11 MoveR: moved resource 'secureTug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  75 ! dockSub_11 MoveR: moved resource 'tug' amount 1 from location 'SubmarineDock' to location 'Harbour'
  75 ! dockSub_11 PutR: released 1 units for resource 'secureTug' at location 'Harbour'
  75 ! dockSub_11 PutR: released 1 units for resource 'tug' at location 'Harbour'
  75 ! dockSub_11 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'OpenSea'
  75 ! dockSub_11 PutR: released 1 units for resource 'submarine' at location 'OpenSea'
  75 ! dockSub_11 Launching entity 'dockSub' (dockSub_14) at time 75
  78 ! genSubs_13 MoveR: moved resource 'submarine' amount 1 from location 'OpenSea' to location 'Harbour'
  78 ! genSubs_13 PutR: released 1 units for resource 'submarine' at location 'Harbour'
  78 ! genSubs_13 Launching entity 'genSubs' (genSubs_14) at time 81
  78 ! dockSub_13 MoveR: moved resource 'submarine' amount 1 from location 'Harbour' to location 'SubmarineDock'
  78 ! dockSub_13 MoveR: moved resource 'secureTug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  78 ! dockSub_13 MoveR: moved resource 'tug' amount 1 from location 'Harbour' to location 'SubmarineDock'
  78 ! dockSub_13 GetR: claimed 1 units for resource 'jetty' at location 'SubmarineDock'
  78 ! dockSub_13 Hold issued for 3
  80 ! *MAIN* - Simulation Closed.
- : eventList * processMgmt * resourcePool * declarationMap =
([("*MAIN*", PD ([], [], 80));
  ("dockSub_12",
```

```
 PD
  ([Try
     [([GetR ("tug", "Harbour", 2)],
       [MoveR ("tug", "Harbour", "SubmarineDock", 2);
        PutR ("jetty", "SubmarineDock", 1);
        MoveR ("submarine", "SubmarineDock", "Harbour", 1);
        MoveR ("tug", "SubmarineDock", "Harbour", 2);
        PutR ("tug", "Harbour", 2)]);
      ([GetR ("tug", "Harbour", 1); GetR ("secureTug", "Harbour", 1)],
       [MoveR ("secureTug", "Harbour", "SubmarineDock", 1);
        MoveR ("tug", "Harbour", "SubmarineDock", 1);
        PutR ("jetty", "SubmarineDock", 1);
        MoveR ("submarine", "SubmarineDock", "Harbour", 1);
        MoveR ("secureTug", "SubmarineDock", "Harbour", 1);
        MoveR ("tug", "SubmarineDock", "Harbour", 1);
        PutR ("secureTug", "Harbour", 1); PutR ("tug", "Harbour", 1)])];
    MoveR ("submarine", "Harbour", "OpenSea", 1);
    PutR ("submarine", "OpenSea", 1); Entity ("dockSub", "dockSub", 0)],
   [(("SubmarineDock", "submarine"), RA 1);
    (("SubmarineDock", "jetty"), RA 1)],
   81));
("genSubs_14",
 PD
  ([GetR ("submarine", "OpenSea", 1); Hold 3;
    MoveR ("submarine", "OpenSea", "Harbour", 1);
    PutR ("submarine", "Harbour", 1); Entity ("genSubs", "genSubs", 3)],
   [], 81));
("dockSub_13",
 PD
  ([MoveR ("tug", "SubmarineDock", "Harbour", 1);
    PutR ("tug", "Harbour", 1);
    MoveR ("secureTug", "SubmarineDock", "Harbour", 1);
    PutR ("secureTug", "Harbour", 1); Hold 6;
    Try
     [([GetR ("tug", "Harbour", 2)],
       [MoveR ("tug", "Harbour", "SubmarineDock", 2);
        PutR ("jetty", "SubmarineDock", 1);
        MoveR ("submarine", "SubmarineDock", "Harbour", 1);
        MoveR ("tug", "SubmarineDock", "Harbour", 2);
        PutR ("tug", "Harbour", 2)]);
      ([GetR ("tug", "Harbour", 1); GetR ("secureTug", "Harbour", 1)],
       [MoveR ("secureTug", "Harbour", "SubmarineDock", 1);
        MoveR ("tug", "Harbour", "SubmarineDock", 1);
        PutR ("jetty", "SubmarineDock", 1); MoveR ...]);
       ...];
     ...],
    ...));
 ...],
...)
```