A program is defined by task declarations, placeholders for external implementation, plus procedures made up of instruction sequences. A program’s execution comprises the dynamic selection of vehicles from the network “cloud” through select instructions, and vehicle bindings to task execution through step instructions. Variables and imperative-like constructs (e.g., do-while) structure a program’s control and data flows. In the example program (right), the UUVs are first selected from the network. The following step then fires the data sampling tasks concurrently, one per UUV. When the data sampling step terminates for all of the UUVs, the UAV is selected and engages in rendezvous with each of the UUVs. For each UUV-UAV rendezvous, the rendezvous procedure fires the cooperative RV task once or more (execution loops if the task yields output moreData). At the end of the program, the vehicles are released back to the network cloud.

EXAMPLE SCENARIO

Three UUVs (unmanned underwater vehicles) are used for environmental data sampling (e.g., bathymetry measurements) across a given ocean region, together with an UAV (unmanned aerial vehicle) that acts as a “data mule” to collect the data from the UUVs. The sampling region is distributed evenly between the three UUVs, corresponding to three distinct tasks that take form as “row pattern” maneuvers.

The UUVs emerge after data sampling and the UAV collects their data through three instances of a cooperative rendezvous task, each of which involving the UAV and each of the UUVs. A UAV-UUV rendezvous makes the UAV move close to the UUV operation area for improved connectivity first, after which the actual data transfer proceeds. The rendezvous may loop if a single round of data transfer is insufficient.

EXPERIMENTS

We conducted simulation and field tests for the NVL prototype. The field tests took place at the Leixões harbour near Porto and involved LAUV Seacon vehicles developed at LSTS [1].

We used the Neptus system for the specification of tasks as IMC maneuver plans [2]. During program execution, the progress of tasks could also be monitored using a Neptus console.

IMPLEMENTATION

The prototype implementation of NVL consists of an Eclipse IDE plug-in for program edition and validation, an interpreter for program execution, and supervisor modules that run onboard each vehicle. The implementation builds on top of the open-source LSTS toolchain [2] (http://github.com/LSTS) comprising: IMC, a message-based interoperability protocol; DUNE: a platform for onboard software; and the Neptus command-and-control infrastructure.

REFERENCES
