The 2008 SCM-PC
User’s Guide

The SCM-PC Team:
Alberto Sardinha,
Michael Benisch,
Ram Ravichandran,
Norman Sadeh

Carnegie Mellon University
Contents

1. Introduction ................................................................. 3
2. Installing and Running the SCM Procurement Challenge Server .............................. 4
3. Installing and Running the SCM Procurement Challenge AgentWare ........................ 7
4. Implementing an Agent for the SCM Procurement Challenge ..................................... 13
1. Introduction

The TAC SCM Procurement Challenge aims to reflect the importance of long-term procurement contracts, such as quantity flexible contracts, in many actual supply chains. It complements the current “baseline” TAC-SCM scenario by extending the space of procurement options available to manufacturer agents and allowing them to enter long-term contracts with supplier agents. In contrast to the 2007 SCM-PC supplier model, we have decided to allow each supplier to offer both one-off and long term contracts, rather than restricting each supplier to offer only one type of contract. This is representative of how most suppliers operate in the real world. Specifically, manufacturer agents will rely on a combination of:

- Long-term “quantity flexible” contracts. These contracts specify minimum component quantities a manufacturer agent commits to purchasing weekly (at a fixed price) from a given supplier agent and include options to increase these quantities by up to some percentage (at the same fixed price).
- One-off contracts. These are the same supply contracts as the ones negotiated in the baseline TAC-SCM scenario.

The TAC-SCM Procurement Challenge (or "SCM-PC") game simulates $D$ days of operation (where $D = 100$ days). It features $n$ manufacturer agents (where $n = 3$) competing for supply contracts from 8 different supplier agents (the same suppliers as the baseline TAC-SCM scenario). In SCM-PC-08, every supplier offers both long-term and one-off contracts. Long-term contracts are negotiated at the start of the game, and last for the game's full duration. Each week, manufacturer agents may decide to order more than the minimum quantities they committed to up to a pre-specified max quantity. Each day, they may also decide to procure additional components outside of their long-term procurement contracts (specifying quantity and delivery date).

The SCM-PC server will simulate the supplier agents and provide banking, production and warehousing services to the manufacturer agents. It will also randomly generate the demand and require each manufacturer agent to satisfy an equal part: this allows entrants to ignore the customer bidding dimension of their supply chain, making SCM-PC simpler than the baseline game. In SCM-PC, manufacturer agents are only expected to focus on procurement decisions. At the end of the game, the manufacturer agent with the most money in the bank is declared the winner.

In summary, the SCM-PC Procurement Challenge differs from the baseline TAC-SCM game in two significant ways: (i) manufacturer agents don't have to worry about customer bidding, and (ii) manufacturer agents are now required to manage risk across a combination of long-term and one-off contracts. This risk management is consistent with practices found in many actual supply chains.
2. Installing and Running the SCM Procurement Challenge Server

This section presents a step-by-step guide on how to install and run the SCM Procurement Challenge server.

Step 1 - Download the SCM Procurement Challenge Server

The server (a Zip archive) can be downloaded from the following website:

http://www.escm.cs.cmu.edu/scm_pc/
Step 2 - Extract the files

Select a folder (or directory) and extract the files in the Zip archive:

```
CMU-233232:temp alberto$ unzip scmpc_server_beta1.2.zip
Archive: scmpc_server_beta1.2.zip
  creating: scmpc_server_beta1.2/
  creating: scmpc_server_beta1.2/config/
  inflating: scmpc_server_beta1.2/config/server.conf
  inflating: scmpc_server_beta1.2/config/tac08scm_sim.conf
  creating: __MACOSX/
  creating: __MACOSX/scmpc_server_beta1.2/
  creating: __MACOSX/scmpc_server_beta1.2/config/
  inflating: __MACOSX/scmpc_server_beta1.2/config/._tac08scm_sim.conf
  inflating: scmpc_server_beta1.2/config/tac08scm_viewer.conf
  creating: scmpc_server_beta1.2/images/
  inflating: scmpc_server_beta1.2/images/assemblyline.jpg
  inflating: scmpc_server_beta1.2/images/banner2.jpg
  inflating: scmpc_server_beta1.2/images/color_info2.jpg
  inflating: scmpc_server_beta1.2/images/customer.jpg
  inflating: scmpc_server_beta1.2/images/customer2.jpg
  inflating: scmpc_server_beta1.2/images/customer_plate.jpg
  inflating: scmpc_server_beta1.2/images/Factory.jpg
  inflating: scmpc_server_beta1.2/images/Factory_plate.jpg
  inflating: scmpc_server_beta1.2/images/manufacturer.jpg
  inflating: scmpc_server_beta1.2/images/manufacturer_plate3.jpg
  inflating: scmpc_server_beta1.2/images/supplier2.jpg
  inflating: scmpc_server_beta1.2/images/supplierText.jpg
  inflating: scmpc_server_beta1.2/images/Thumbs.db
  creating: scmpc_server_beta1.2/lib/
  inflating: scmpc_server_beta1.2/lib/javac.servlet.jar
  inflating: scmpc_server_beta1.2/lib/JETTY_LICENSE.html
  inflating: scmpc_server_beta1.2/lib/org.mortbay.jetty.jar
  creating: scmpc_server_beta1.2/public_html/
  creating: scmpc_server_beta1.2/public_html/code/
  creating: scmpc_server_beta1.2/public_html/code/config/
  inflating: scmpc_server_beta1.2/public_html/code/config/tac08scm_viewer.conf
  inflating: scmpc_server_beta1.2/public_html/code/simviewer.jar
  inflating: scmpc_server_beta1.2/public_html/robots.txt
  inflating: scmpc_server_beta1.2/README.txt
  inflating: __MACOSX/scmpc_server_beta1.2/._README.txt
  inflating: scmpc_server_beta1.2/scmserver.jar
  inflating: scmpc_server_beta1.2/SRVM_manifest.txt
CMU-233232:temp alberto$
```
Step 3 – Running the SCM Procurement Challenge Server

Open a command prompt (or terminal) and navigate to the folder where you extracted the files from the Zip archive:

```
CMU-233232:scmpc_server_beta1.2 alberto$ ls -l
total 1810
-rwxr-xr-x 1 alberto alberto 976 Mar 7 16:13 README.txt
-rwxr-xr-x 1 alberto alberto 152 Jul 13 2004 SRVManifest.txt
drwxr-xr-x 5 alberto alberto 170 Mar 7 15:31 config
drwxr-xr-x 15 alberto alberto 510 Aug 15 2007 images
drwxr-xr-x 5 alberto alberto 170 Aug 15 2007 lib
drwxr-xr-x 4 alberto alberto 136 Aug 15 2007 public_html
-rwxr-xr-x 1 alberto alberto 918299 Mar 13 16:13 scmserver.jar
CMU-233232:scmpc_server_beta1.2 alberto$
```

You will need Java SDK 1.5.0 (you can find it at [http://java.sun.com/](http://java.sun.com/)) to be able to run this server. The server reads the configuration file 'server.conf' at startup (in the 'config' folder). This file is used to configure the simulation settings (such as log levels and simulation files). You should at least change the password for the admin user in this file.

The 'config/tac06scm_sim.conf' file is also used for changing the game parameters (such as game length, builtin agent implementations, etc.). The file already contains all the default values for the 2008 SCM Procurement Challenge.

To run the server just type "java -jar scmserver.jar":

```
CMU-233232:scmpc_server_beta1.2 alberto$ java -jar scmserver.jar
12:30:38.207 EVENT Starting Jetty/4.2.9
12:30:38.283 EVENT Started org.mortbay.http.NCSAMultipartLogb03c1a
12:30:38.293 EVENT Started HttpContext[/]
12:30:38.290 EVENT Started SocketListener on 0.0.0.0:8080
12:30:38.290 EVENT Started org.mortbay.http.HttpServer@6de49c
```

3. Installing and Running the SCM Procurement Challenge AgentWare and PCExampleAgent

This section presents a step-by-step guide on how to install and run the SCM Procurement Challenge AgentWare. The purpose of the AgentWare is to provide a basic set of tools for developing SCM-PC agents. The current release of the AgentWare contains:

- Preliminary APIs for SCM-PC agents;
- Basic setup for communicating with the SCM-PC server;
- Implementation of an example agent.

Step 1 - Download the SCM Procurement Challenge AgentWare

The AgentWare (a Zip archive) can be downloaded from the following website:

http://www.escm.cs.cmu.edu/scm_pc/
Step 2 - Extract the files

Select a folder (or directory) and extract the files in the Zip archive:

```
CMU-233232:temp alberto$ unzip scmpc_agentware_beta1.2.zip
Archive: scmpc_agentware_beta1.2.zip
  creating: scmpc_agentware_beta1.2/
  inflating: scmpc_agentware_beta1.2/.DS_Store
  creating: __MACOSX/
  inflating: __MACOSX/scmpc_agentware_beta1.2/
  inflating: __MACOSX/scmpc_agentware_beta1.2/..DS_Store
  inflating: scmpc_agentware_beta1.2/Make.conf
  inflating: scmpc_agentware_beta1.2/com/
  creating: scmpc_agentware_beta1.2/com/botbox/
  inflating: scmpc_agentware_beta1.2/com/botbox/Manifest.txt
  inflating: scmpc_agentware_beta1.2/com/botbox/ArrayQueue.java
  inflating: scmpc_agentware_beta1.2/com/botbox/ArrayUtils.java
  inflating: scmpc_agentware_beta1.2/com/botbox/compile.bat
  inflating: scmpc_agentware_beta1.2/com/botbox/compile.sh
  creating: scmpc_agentware_beta1.2/edu/
  creating: scmpc_agentware_beta1.2/edu/cmu/
  creating: scmpc_agentware_beta1.2/edu/cmu/cs/
  creating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/
  creating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/CM
  inflating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/cm
  inflating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/cm/ManufacturerInfo.java
  inflating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/cm/Order.java
  inflating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/cm/OrderStore.java
  inflating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/cm/RFOStore.java
  inflating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/cm/SCMPAgent.java
  creating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/props/
  inflating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/props/Build.java
  inflating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/props/ComponentOrder.java
  inflating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/props/Contract.java
  inflating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/props/CustomerOrder.java
  inflating: scmpc_agentware_beta1.2/edu/cmu/cs/scmpc/props/StartNegotiationParam.java
  creating: scmpc_agentware_beta1.2/images/
  inflating: scmpc_agentware_beta1.2/images/manufacturer.jpg
  inflating: scmpc_agentware_beta1.2/images/Thumbs.db
  inflating: scmpc_agentware_beta1.2/PCEXampleAgent.java
  inflating: scmpc_agentware_beta1.2/README.txt
```
Step 3 – Compiling the SCM Procurement Challenge AgentWare and PCExampleAgent.java

Open a command prompt (or terminal) and navigate to the folder where you extracted the files from the AgentWare’s archive. You will need Java SDK 1.5.0 (you can find it at http://java.sun.com) to be able to compile and run this AgentWare. Type "compile.bat" (or compile.sh) to compile the AgentWare:

![Compilation Result]

Step 3 – Register a new agent

If the SCM-PC server is not running then start it right now (see section 2 - step 3 to start your server). Register your new agent by accessing the following URL:

http://<yourcomputer>:8080/
Click on the “Register new user” link, and register a new user name and password:

Step 4 – Running the PCExampleAgent

The AgentWare will read the configuration file 'aw.conf' once you start your agent. This file is used to change the AgentWare’s configuration (such as the user name, the network address of the server and the main class of your agent). Change the userName, agentPassword and serverHost properties to reflect your current configuration:
Then type "java -jar scmaw.jar" to run an example agent:

Step 5 – Using the Game Viewer

The Game Viewer allows a user to follow a game in real time. Click on the “Coming games” link and then on the “Launch Game Viewer” link to start the Game Viewer:
4. Implementing an Agent for the SCM Procurement Challenge

This section presents a step-by-step guide on how to implement a simple agent for the SCM Procurement Challenge. If you have any questions or comments regarding this document please contact cmieux@cs.cmu.edu.

Step 1 - Download the AgentWare (section 3)

The AgentWare can be downloaded from the following website (an example of an agent is also included in the AgentWare’s archive):

http://www.escm.cs.cmu.edu/scm_pc/

Step 2 – Implementing an Agent

Your agent must extend the class SCMPCAgent. SCMPCAgent is an abstract class that extends Agent and provides the following features for the agent developers:
- APIs for accessing game information.
- APIs for long-term supply contract negotiation
- Automatic bookkeeping of RFQs, orders, and inventory.
- Provides an easier API for handling messages.

```java
public class PCExampleAgent extends SCMPCAgent {
    ...
}
```

Step 3 – Negotiating long-term supply contracts

The method handleLongTermContractNegotiation is called when the negotiation parameters (reserve prices, the quantity flexible parameters and the price parameters) of the long-term contract suppliers have been received (on day -1). The manufacturer agent should use this method to send bids.

```java
public class PCExampleAgent extends SCMPCAgent {
    ...
    protected void handleLongTermContractNegotiation(StartNegotiationParam startNegotiationParam) {
        // add bid
        addLongTermSupplyBid(supplierName, componentID, price,
```
Step 4 – Sending day 0 orders to long-term contract suppliers

The method `handleLongTermContracts` is called when the long-term supply contracts have been received (day 0). The manufacturer agent should use this method to send the first orders.

```java
public class PCExampleAgent extends SCMPCAgent {
    ...
    protected void handleLongTermContracts(List<Contract> contracts) {
        // Send orders for day 0
        for (Contract contract : contracts) {
            // get supplier name, component ID and quantity
            String supplierName = contract.getSupplierAgentAddress();
            int componentID = contract.getComponentID();
            int quantity = contract.getMaxQuantity();
            // add order
            addLongTermSupplyOrder(supplierName, componentID, quantity);
        }
        // send orders
        sendLongTermSupplyOrders();
    }
}
```

Step 5 – Receiving customer orders

The method `handleCustomerOrders` is called when all the customer orders have been received. In the Procurement Challenge, the Customer Agent sends the same bundle of orders to all Manufacturer Agents.

```java
public class PCExampleAgent extends SCMPCAgent {
    ...
}
```
protected void handleCustomerOrders(List<CustomerOrder> customerOrders)
{
    // iterate over all new customer orders
    for (CustomerOrder customerOrder : customerOrders)
    {
        ...
    }
}
...

Step 6 – Sending weekly orders to long-term contract suppliers

The method weeklyLongTermSupplyOrderReminder is called on the last day a manufacturer agent can send the weekly orders to the long-term contract suppliers.

public class PCExampleAgent extends SCMPCAgent
{
    ...
    protected void weeklyLongTermSupplyOrderReminder(List<Contract> contracts)
    {
        // Send orders to long-term contract suppliers
        for (Contract contract : contracts)
        {
            // get supplier name, component ID and quantity
            String supplierName = contract.getSupplierAgentAddress();
            int componentID = contract.getComponentID();
            int quantity = contract.getMaxQuantity();
            // add order
            addLongTermSupplyOrder(supplierName, componentID, quantity);
        }
        // send orders
        sendLongTermSupplyOrders();
    }
    ...
}
Step 7 – Implementing agent’s main loop

The method `handleSimulationStatus` is called when a simulation status has been received and all the messages from the server on that day have been received. For example, the manufacturer agent can use this method to check the inventory and the bank status or send RFQs to the short-term contract suppliers.

```java
public class PCEexampleAgent extends SCMPCAgent {
 ...
 protected void handleSimulationStatus(SimulationStatus status) {
      // Prints inventory status
      printInventory();

      // Print bank status
      printBankStatus();

      // send RFQs to short-term contract suppliers
      sendComponentRFQs();
 }
 ...
}
```

Step 8 – Sending orders to short-term contract suppliers

The method `handleShortTermContractSupplierOffers` is called when a bundle of offers have been received from a short-term contract supplier. In TAC SCM suppliers only send an offer bundle per day in reply to RFQs (and only if they had something to offer). This method can be used to send orders (in response to the offers).

```java
public class PCEexampleAgent extends SCMPCAgent {
 ...
 protected void handleShortTermContractSupplierOffers(
      String supplierAddress,
      OfferBundle offers)
 {
      // order everthing!
      for(int i = 0; i < offers.size(); i++)
      {
         addShortTermSupplierOrder(supplierAddress,
                  offers, i);
      }
```
// send orders
sendShortTermSupplierOrders();
}
...
}

Step 9 – Start and end of simulation

The method `simulationStarted` is called after the agent received all startup information and it is time to start participating in the simulation, and the method `simulationEnded` is called when a game/simulation has ended and the agent should free its resources.

This method can be used to send orders (in response to the offers).

class PCExampleAgent extends SCMPCAgent
{
  ...
  protected void simulationStarted()
  {
    ...
  }

  protected void simulationEnded()
  {
    ...
  }

  ...
}