Proceedings of the 2018 IISE Annual Conference
K. Barker, D. Berry, C. Rainwater, eds.

The X-Supply Game

Sinan Salman
College of Technological Innovation
Zayed University, Abu Dhabi, UAE

Suzan Alaswad
College of Business
Zayed University, Abu Dhabi, UAE

Abstract

This paper introduces the X-Supply Game (XSG), a multi-player, team-based, online pedagogical game that simulates on a small-scale system dynamics found in real supply chains. It challenges players to take on the roles of virtual stations managers in a small supply chain. XSG is played in turns, in which players analyze weekly standings of their stations and decide on orders to suppliers and shipments to customers, and simultaneously deal with multiple competing objectives involving inventory, backorders, fulfillment rate, and transportation resource utilization. The game highlights the inherent difficulty and the importance of tighter integration to achieve higher performance supply chain wide. The XSG is influenced by previous supply chain simulation games, such as the Beer and Wood Supply games, however, it extends on them to include the triple bottom line performance evaluation concept in addition to game configurability and accessibility. For configurability, much of the game setup can be configured to create various game scenarios, including different supply chain network topologies. As for accessibility, the game is designed to run on most internet capable devices, and its source code is released under an open source license allowing its adoption and extension by instructors around the world.

Keywords
Supply chain management, pedagogical team games, supply chain simulation

1. Introduction

System complexity in supply chain management (SCM) is a challenging concept to address in the classroom. Although examples of basic SCM concepts are abundant in our everyday life and are relatable to students, true appreciation of the complexity in supply chain systems and the interdependence found within it remains a challenge. Hence, educators pursue different experiential learning techniques to bridge this gap. One of the most successfully used experiential learning tools is the Beer game, a role-play team-based simulation game invented in the 1960s by Jay Forrester at MIT while investigating system dynamics in SCM (see [1]). Since its initial introduction, the game was adapted into multiple formats including a table game, different online games (e.g., [2]), and into other industries (e.g., the Wood Supply Game [3]). For a full description of the Beer game, its history, and its use the reader is referred to [4].

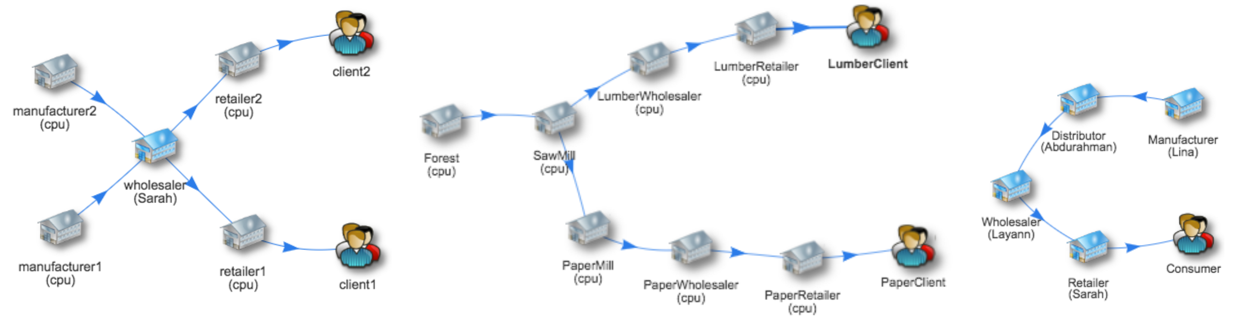
This paper introduces a supply chain game simulator, the X-Supply Game (XSG), which builds on previous games and extends on them in several aspects. First, much of the game setup can be configured to create various game scenarios and supply chain designs. This enables the XSG to simulate most existing games, such as the Beer game and the Wood supply game, via simple configurations, giving the XSG its name. Second, the simulator allows the player to take on the shipping decision making it a part of their station management responsibilities. In contrast, most other games either automate this decision or dictate its outcome through rules that require meeting all existing customer demands using available material. Third, the XSG introduces the Triple Bottom Line (TBL), a performance evaluation framework, to supply chain games where players explore decision tradeoffs impacting supply chain's performance as it relates to operational costs, customer satisfaction, and environmental footprint. Finally, the game's source code is released under an open source license allowing its adoption and extension by educators around the world. These features enable educators to design game configurations to meet their topic and classroom needs. As such, the XSG attempts to address the need for a game simulator tool that is flexible, extensible, and open to help bring experiential learning to SCM classrooms. This article focuses on the design, implementation, capabilities, and configurability of the XSG. The pedagogical use of the XSG in SCM education will be the focus of a subsequent journal article.

The remainder of this paper is organized as follows. Section 2 presents the game simulator design, simulation logic, and game configurability. Section 3 discusses the game user interface and its various reporting capabilities. Section 4 briefly overviews the implementation aspects of the simulator as a starting point for educators interested in extending it further. Finally, Section 5 summarizes and concludes the article.

2. Game Design

The XSG is designed as a synchronized, multi-player, team-based, and online supply chain simulation game. In the game, players connect to a central web server and take on the roles of virtual station managers in a small supply chain. Every week players analyze their stations' stats and decide on orders to suppliers and shipments to customers, while managing multiple competing objectives. When deployed in classrooms, the game helps players realize through experiential learning the complexities associated with managing even small supply chains. It also enables players to explore and test strategies to improve supply chain performance.

The XSG utilizes a flexible simulation engine built using an object-oriented design. In the simulation engine, games are dynamically built from two node types: stations and consumer demand points. Nodes are implemented as independent objects, and once instantiated and configured they are connected to form a supply chain. This design enables the simulator to model numerous supply chain network topologies (layers, multiple suppliers/customers per node, multiple consumer demand points, etc.). Figure 1 illustrates three examples of supply chain games made possible by this flexible simulator design. It should be noted that the XSG is capable of modeling virtually any supply chain network topology given that it does not include closed loops, as such networks result in circular reference in the simulator logic.



(a) Five-stations network (b) Wood supply game network (c) Beer game network

Figure 1: Network representation diagrams

Tables 1 and 2 list configuration parameters for both node types and illustrate how they are used to setup nodes according to scenario requirements. Since parameters listed in the tables are node-instance specific, different instances of the same node type may have different values, and therefore configurations. Using this design, much of the game setup can be configured to create an expansive range of game scenarios, including different supply chain network topologies, node throughput limits, supply chain delays, and operational costs, to name a few. In addition, two of the station node configuration parameters (*auto-shipping* and *auto-ordering*) deal with station decision automation. Thus, in the same simulation game, while shipping and ordering decisions can be managed manually in some stations, they can also be completely or partially automated in other stations.

Allowing players to take on ordering and shipping decision responsibilities in the game simulates real supply chain operations more closely and expose the players to some of its decision complexity and associated consequences. The importance of enabling both decisions becomes evident when one considers the option of delaying material shipping to optimize truck utilization. Since the XSG measures the environmental impact of player decisions via the number of extraneous trucks used to ship units, minimizing this number will also minimize the environmental footprint. Therefore, the XSG implements a TBL approach by challenging players to analyze the results of their decisions on network's operational costs (holding, backorder, and transportation costs), customer satisfaction (fulfillment rate), and environmental footprint (number of extraneous trucks). Such tradeoffs help players appreciate the SCM challenge: managing competing objectives while working with limited information, resources, and set timeframes.

Table 1: Parameters for consumer demand point nodes

Parameter	Description	Example
Name	Unique demand point name	Customer
Demand	An array of weekly demand values (units)	[98, 135, 176, 124]

Table 2: Parameters for station nodes

Parameter	Description	Example
Name	Unique station name	Retailer
Auto-shipping	Use auto shipping logic	False
Auto-ordering	Use auto ordering logic	True
Holding cost	Cost of holding a unit of product for a week (\$/unit/week)	\$1
Backorder cost	Cost of not fulfilling a unit ordered for a week (\$/unit/week)	\$2
Transport cost	Cost of shipping one truckload (\$/truck)	\$220 per truck
Transport size	Truck capacity (units/truck)	200 units per truck
Shipping delay	Shipment transit time between a supplier and a customer (weeks)	2 weeks
Ordering delay	Processing time before a supplier receives a customer order (weeks)	1 weeks
Queue initial value	Value used to initialize order/shipment queues for week 1 (units)	50 units
Initial inventory	Inventory count at the start of simulation (units)	200 units
Safety stock	Target safety stock; used in auto ordering logic (units)	50 units
Production minimum	An array of weekly throughput minimums (units)	[0, 0, 50, 50]
Production maximum	An array of weekly throughput maximums (units)	[200, 300, 300, 200]

To help the reader follow the game simulator logic, key concepts used in its design are explained here prior to the logic presentation:

- The simulation includes two types of delays at each station: transportation/receiving and order processing. These delays are modeled in the simulation engine using FIFO queues. Each week every queue releases an entry and admits another. Therefore, the queue size is determined by the corresponding delay configuration parameter, with one week delay equal to one queue position. In the below logic, transportation (i.e. receiving) queues are denoted by RQ, order processing queues are denoted by OQ, and queue inserts and removals are indicated using the $Add()$ and $Pop()$ methods, respectively.
- If station i is an end station node, i.e. a station with no suppliers, it will produce units demanded by its customer(s) internally, and the receiving queue RQ will simulate production time delay instead of the transportation delay. In this case, the order processing queue OQ simulates the internal order processing delay.
- If the order decision at a given station is manual, the function $DecideOrders()$ simply reads player's input. If the decision is automated, the function calculates $TotalOrders = \sum_c BO_c + SS - I - \sum_s OO_s$, and allocates the result to suppliers such that: those with lower outstanding orders receive more unit orders. Note that BO_c is the number of units on backorder from customer c, SS is the safety stock level (units), I is the number of units available in inventory, and OO_s is the number of units on outstanding orders from supplier s.
- If the shipping decision at a given station is manual, the function $DecideShipments()$ simply reads player's input. If the decision is automated, the function will attempt to satisfy all current and backorders if inventory permits. When there is not enough inventory, customers with higher unit backorder counts will receive more unit shipments.
- N denote the sorted list of network nodes, such that later nodes can only supply earlier nodes and not vice versa, assuming no closed loops exist. The sorted list enables the simulation logic to process orders in the network sequentially from consumer demand points to end nodes.
- The simulation model requires initialization to correctly process week 1. Initial values for inventory levels, queues, and outstanding orders are needed. These are set based on values found in *Queue initial value* and *Initial inventory* parameters (see table 2).

Now we are in position to introduce the simulator pseudocode:

Game simulator main logic

Let:

N	be the sorted set of network nodes
N_{end}	be the set of station nodes with no suppliers; AKA end station nodes
I_i	be the number of units available in inventory at node i
BO_c	be the number of units on backorder from customer c
OO_s	be the number of units on outstanding orders from supplier s
PO_{cs}	be the number of units in a purchase order from customer c to supplier s
R_s	be the number of units received from supplier s
S_c	be the number of units shipped to customers c
RQ_{xy}	be node x receiving queue from supplier y (transport/production delay)
OQ_{is}	be node i ordering queue from supplier s (processing delay)

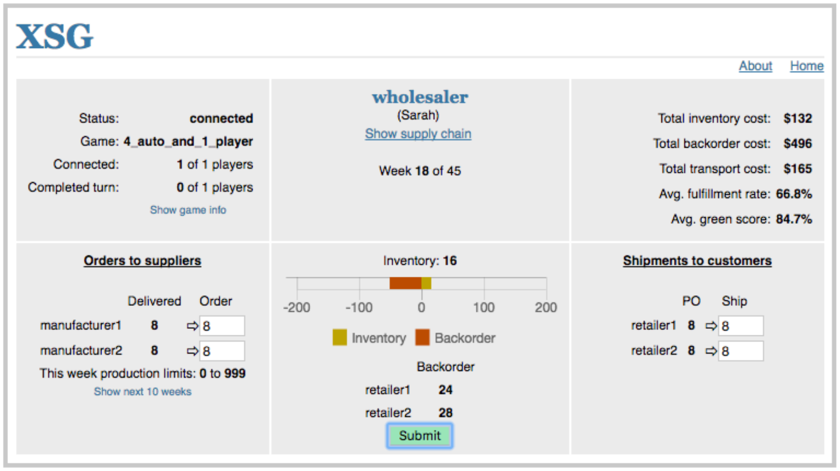
Input: I, OO, RQ, and OQ initial values

```
1.  For $week$ = 1 to $week_{stop}$
2.      For i in N
3.          $\text{Let c} = \text{set of}\{\text{direct customer nodes of i}\}$
4.          $\text{Let s} = \text{set of}\begin{Bmatrix}\text{direct supplier nodes of i,} & \text{if i} \notin \text{N}_{\text{end}} \\ \text{i,} & \text{if i} \in \text{N}_{\text{end}}\end{Bmatrix}$
5.          $\text{R}_\text{s} = \text{RQ}_{\text{is}}.\text{Pop}(), \forall\text{s}$
6.          $\text{OO}_\text{s} = \text{OO}_\text{s} - \text{R}_\text{s}, \forall\text{s}$
7.          $\text{I}_\text{i} = \text{I}_\text{i} + \sum_\text{s} \text{R}_\text{s}$
8.          $\text{PO}_{\text{is}} = \text{OQ}_{\text{is}}.\text{Pop}(), \forall\text{s}$
9.      Display weekly stats on station screen
10.     Wait until all manual stations complete turn
11.     For i in N
12.         $\text{Let c} = \text{set of}\{\text{direct customer nodes of i}\}$
13.         $\text{Let s} = \text{set of}\begin{Bmatrix}\text{direct supplier nodes of i,} & \text{if i} \notin \text{N}_{\text{end}} \\ \text{i,} & \text{if i} \in \text{N}_{\text{end}}\end{Bmatrix}$
14.         $\text{BO}_\text{c} = \text{BO}_\text{c} + \text{PO}_{\text{ci}}, \forall\text{c}$
15.         $\text{S}_\text{c} = \text{DecideShipments}(\text{I}, \text{BO}), \forall\text{c}$
16.         $\text{I}_\text{i} = \text{I}_\text{i} - \sum_\text{c} \text{S}_\text{c}$
17.         $\text{BO}_\text{c} = \text{BO}_\text{c} - \text{S}_\text{c}, \forall\text{c}$
18.         $\text{RQ}_{\text{ci}}.\text{Add}(\text{S}_\text{c}), \forall\text{c}$
19.         $\text{PO}_{\text{is}} = \text{DecideOrders}(\text{I}, \text{BO}, \text{OO}), \forall\text{s}$
20.         $\text{OQ}_{\text{is}}.\text{Add}(\text{PO}_{\text{is}}), \forall\text{s}$
21.         $\text{OO}_\text{s} = \text{OO}_\text{s} + \text{PO}_{\text{is}}, \forall\text{s}$
```

The following describes the above logic. Lines 3 through 8 are the weekly initialization block; where unit shipments are received from transportation delay queues (line 5), outstanding orders and inventory levels are adjusted with received units (lines 6 and 7), and purchase orders are received from processing delay queues (line 8). Lines 3 and 4 define the set of supplier and customer nodes to the current node i. After all nodes complete weekly initialization, weekly stats are displayed to players, and the simulation waits for players to enter their ordering and/or shipping decisions (lines 9 and 10). When all players complete their turns, weekly processing starts (lines 12 through 21). The following takes place in weekly processing: backorders are adjusted with new purchase orders (line 14), shipments are decided (line 15), inventories are decreased by the number of shipped units (line 16), backorders are again adjusted with the number of shipped units (line 17), shipments are entered into the transportation delay queue (line 18), orders to suppliers are decided and entered into the order processing delay queue (lines 19 and 20), and finally outstanding orders are adjusted with ordered units (line 21). Lines 12 and 13 perform the same function as lines 3 and 4.

3. Game User Interface and Outcomes Reporting

The XSG provides a clean user interface for players, and a variety of screens for educators to manage and monitor on-going games, and report on game results for post-game debriefing discussions using a variety of interactive graphs. Figure 2 shows a screenshot of the player screen, which is divided into upper and lower portions, with each including three sections. The upper portion is reserved for informational data, including: game server status, game progress, and Key Performance Indicators (KPI). The lower portion is reserved for player's action, and includes an ordering section, an inventory/backorder section, and a shipping section. The number of ordering and shipping entry boxes depends on the automation of decisions in each station, as well as the number of customs and suppliers for each station.



XSG

About Home

Status:	connected
Game:	4_auto_and_1_player
Connected:	1 of 1 players
Completed turn:	0 of 1 players

Show game info

wholesaler
(Sarah)
Show supply chain

Week 18 of 45

Total inventory cost:	$132
Total backorder cost:	$496
Total transport cost:	$165
Avg. fulfillment rate:	66.8%
Avg. green score:	84.7%

Orders to suppliers

	Delivered	Order
manufacturer1	8	⇨ 8
manufacturer2	8	⇨ 8

This week production limits: 0 to 999

Show next 10 weeks

Inventory: 16

	Backorder
retailer1	24
retailer2	28

Submit

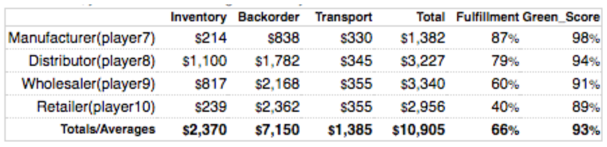
Shipments to customers

	PO	Ship
retailer1	8	⇨ 8
retailer2	8	⇨ 8

Figure 2: Player screen

During the simulation and at its completion the game manager (i.e. educator) has access to a variety of reporting tools, which include: a game summary table (Figure 3), a radar graph comparing performance objectives between stations (Figure 4), and several graphs showing orders (Figure 5.a), inventory levels and backorders (Figure 5.b), nodes' available units ($I - BO$), unit deliveries, truck shipments, and extra shipments (Figure 5.c).

Using these reports, the game manager can run a constructive debriefing session where questions on performance challenges and ways to address them can be discussed in an open investigative format. The educator may start the session with a few open-ended questions on performance challenges, and then use the reports to guide the discussion.



	Inventory	Backorder	Transport	Total	Fulfillment	Green_Score
Manufacturer(player7)	$214	$838	$330	$1,382	87%	98%
Distributor(player8)	$1,100	$1,782	$345	$3,227	79%	94%
Wholesaler(player9)	$817	$2,168	$355	$3,340	60%	91%
Retailer(player10)	$239	$2,362	$355	$2,956	40%	89%
Totals/Averages	**$2,370**	**$7,150**	**$1,385**	**$10,905**	**66%**	**93%**

Figure 3: Game summary table

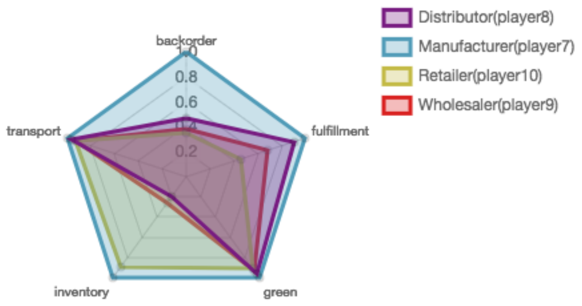


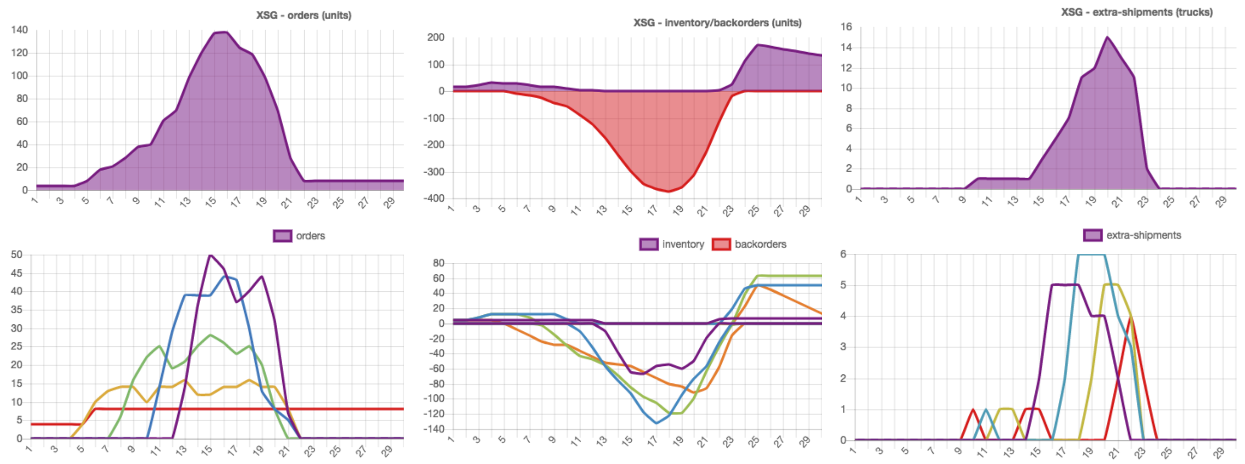
Figure 4: Stations performance comparison graph

4. Simulator Implementation

The XSG is designed for accessibility and extensibility; the game is designed to run on most internet capable devices (e.g., laptops, tablets, and smart phones) without the need to install additional software. To that end, all simulation logic is implemented on the server side, and only minimal user interface logic remains on the client (player) side. The simulation engine is implemented using object-oriented Python, with web-server functions implemented using

Python's Flask web-framework library. On the client side, only HTML, CSS, and JavaScript are used to ensure cross platform compatibility.

The XSG source code is released under the GPLv3 open source license [5] allowing its adoption and extension by educators around the world. Its project site can be found at [6] which includes a link to a public server hosting the game to allow its direct adoption into classrooms without the burden of installation.



(a) Orders (b) Inventory and backorders (c) Extra shipments

Figure 5: KPI plots (top: network total, bottom: individual stations)

5. Conclusion

This paper introduces the XSG, an educational game that aims at teaching supply chain management using an experiential approach. The game highlights the inherent difficulty and the importance of tighter integration and better sharing of information to achieve higher performance in supply chains. The paper discusses the game's design, major concepts, and capabilities. The introduction of TBL objectives in the game invites a sustainability prospective into this educational activity, and highlights the increased difficulty of managing supply chains when multiple competing objectives are involved, as would be expected in a real-world scenario. The game design also allows for a great deal of flexibility in constructing games in different network designs and parameter configurations to meet varying educational needs in this area.

The XSG is still under active development to include additional features to facilitate its adoption by the widest range of users. Future directions for the XSG may include multi-product supply chains and stochastic events (e.g., stochastic demand, supply chain delays, and production capacities).

Acknowledgements

This work was supported by the Zayed University Research Cluster Fund (R16084).

References

1. Forrester, J. W., 1957, "Industrial Dynamics. A Major Breakthrough for Decision Makers," Harvard Business Review, 36(4), 37–66.
2. Sarkar, S., and Kumar, S., 2016, "Demonstrating the Effect of Supply Chain Disruptions through an Online Beer Distribution Game*," Decision Sciences Journal of Innovative Education, 14(1), 25–35.
3. D'Amours, S., Marier, P., Rönnqvist, M., Azouzi, R., and Fjeld, D., 2017, "The Online Wood Supply Game," INFORMS Transactions on Education, 18(1), 71–87.
4. Riemer, K., 2008, "The Beergame in Business-to-Business Ecommerce Courses-A Teaching Report," BLED 2008 Proceedings, 1.
5. "GPLv3," GNU GENERAL PUBLIC LICENSE [Online]. Available: https://www.gnu.org/licenses/gpl-3.0.en.html. [Accessed: 07-Jan-2018].
6. Salman, S., XSG [Online]. Available: https://sinansalman.github.io/xsg/. [Accessed: 04-Mar-2018].