Salt Seller
Realization of a Business Game

Felix Kiechle, Corinna Uhr, Mario Schlereth, Yao Zhang, Leon O. Burkard
Andreas C. Sonnenbichler and Andreas Geyer-Schulz
Institute of Information Systems and Marketing, Karlsruhe Institute of Technology, Zirkel 2, Karlsruhe, Germany

Keywords: Salt Seller, Business Games, Gaming Simulation, Simulators, Business Simulation, Browser, Multi-player.

Abstract: This paper describes the realization and implementation of a business game named Salt Seller. Salt Seller was first described by John Sterman in 2014 and implemented for the MIT Sloan School of Management. However, Sterman only provided a high-level description of Salt Seller and no low-level details about the realization or the market model were revealed. Our first aim is to fill this gap. As our second contribution introduces a generalized *n*-person framework for round-based games which serves as an abstract game architecture and encapsulates the round synchronization mechanisms. Utilizing this framework we provide a detailed view on a possible model of the market mechanisms for Salt Seller. We describe all the market functions which are needed to calculate the game outputs in each period. Our work aims to improve decision making in management and to support teaching. To support this third goal we implemented the game using modern web technologies allowing us to use the game in students' lessons at the KIT. All input and output parameters of every round are stored in a database for later analysis of the players' behaviour. Generally, our implementation demonstrates how general round-based multi-player games can be realized in a browser, our architecture supports various other 2-player round-based games.

1 INTRODUCTION

Simulations are an essential tool to teach pilots, power plant operators, managers and experts in complex decision situations. Simulations are used to train regular work processes as well as emergencies: Landing a plane during a storm can be best trained with the help of realistic simulators.

Management simulations are used to train managers, experts, and students to get a better understanding of the effects of decisions in complex systems, e.g. markets and supply chains. One of the earliest examples is the Beer Distribution Game (Forrester, 1961; Jarmain, 1963; Sterman, 1989). It demonstrates the dynamics of a supply chain and illustrates the bullwhip effect. Simulators and games like the Beer Distribution Game or the Salt Seller Game are designed to teach management principles, strategy, dynamics and sustainability.

The most complex decisions managers have to make are price decisions in competitive markets. The Salt Seller is one of the most basic simulators to understand the general dynamics of stable markets (Sterman, 2014, p.90ff.). It reduces the dimensions of competition in industry to pricing giving "participants the opportunity to set prices for a commodity in an important industry" (Sterman, 2014, p.92) and to experience the effects of their decisions on market shares and profits. The salt seller 2-person game reduces complexity by the following assumptions: Constant marginal production costs, symmetric cost structure, an undifferentiated product and a stable market with low rates of entries or exits, slow demand growth, slow technological changes and few opportunities for cost reductions. In spite of these strong assumptions, "understanding the basic dynamics of such markets is fundamental" (Sterman, 2014, p.92).

In the Salt Seller game the players represent two salt producers who have the same factors of production and who face the same marginal costs. They produce salt of the same quality and each of them has enough capacity to cover the whole salt demand of the world market. The only factor the players can influence is the price per ton and round (period) they sell their salt at. Depending on this price in relation to the price of the opponent their market shares, sales

Kiechle F., Uhr C., Schlereth M., Zhang Y., O. Burkard L., C. Sonnenbichler A. and Geyer-Schulz A..
Salt Seller - Realization of a Business Game.
DOI: 10.5220/0005503501270135
In *Proceedings of the 12th International Conference on e-Business* (ICE-B-2015), pages 127-135
ISBN: 978-989-758-113-7

and profits per round will be calculated. After a fixed number of rounds the game ends and the player with the highest total profit wins the game.

This paper is structured as follows: In section 2 the term business game is defined in relation to the history of business games, other business games are cited and related work is discussed. Section 3 gives a brief review of the taxonomy of business games and classifies Salt Seller. Section 4 introduces a general framework for n-player round-based games. As application of the general framework of section 4 we describe in section 5 the Salt Sellers' market model which consists of parametrized market functions in order to calculate the game's outputs in each period. The implementation which is presented in section 6 of the market model functions and the multi-player architecture has been done with the help of the Python Web application framework Django (Django, 2015) followed by an evaluation and a conclusion in section 7 and 8.

2 BUSINESS GAMES

A gaming simulation which can be traced back to a Chinese war game Wei Qi (game of encirclement) as far as 3000 BC consists of interactions among groups of players (decision makers) placed in a predefined setting and constrained by a set of rules and procedures. In a gaming simulation a player fulfills a prescribed role with a defined task in a particular realistic situation (Hsu, 1989).

Business games (BGs) are games within a business environment that can lead to one or both of the following results: the training of players in business skills (hard and/or soft) or the evaluation of players' performances (quantitatively and/or qualitatively) (Greco et al., 2013).

Triggered by the integration of developments in war games, operations research, computer technology and education theory, BGs arrived on the scene in the late 1950s for example with development of the American Management Association (AMA) decision simulation game (Ricciardi and Marting, 1957). In the following years games such as the Beer Distribution Game (Sterman, 1989; Jarmain, 1963; Forrester, 1961) or the Markstrat Simulation Game (Larréché and Gatignon, 1977) were released and remained popular until today (Stratx Simulations, 2015). An exhaustive survey about the historical development of the field of business games is presented by Wolfe (Wolfe, 1993).

Today the field of business games is influenced by current technology developments such as natural language and advanced graphics processing, new business models like learning on demand and online feedback mechanisms based on artificial intelligence (Summers, 2004). BGs also serve as experimental environment for training artificial agents and for studying deviations of human players from rationality. Furthermore, BGs are used as incentives to make crowd based online work (crowdsourcing) more attractive and effective (Rokicki et al., 2014). They are implemented by global enterprises like Google, Microsoft, American Express, Caterpillar, etc. to train their employees and managers (Uskov and Sekar, 2014).

Since their inception the main purpose of BGs has been to teach. Over the years BGs became very popular in management education. Nowadays, almost every MBA program requires students to play one or more management simulations, and BGs are even more commonly applied at the undergraduate level (Faria, 1998). In addition, role-playing in the business context can improve soft-skills such as team work, leadership, and practicing concepts and skills used in strategic management, marketing, finance, and project management.

3 GAME CLASSIFICATION OF SALT SELLER

Eilon's early taxonomy (Eilon, 1963) classifies BGs according to design characteristics (computer/non-computer, total enterprise/functional, ...) and expected use (management training, selling purpose, research).

Other taxonomies focus on a specific aspect of a game, such as the web technologies used in order to provide distant education (Bernard, 2014) or the skills that are mediated (product development, project management, logistic skills, risk management) (Riedel and Hauge, 2011).

A complete taxonomy of BGs is presented by Greco et al. (Greco et al., 2013) which combines and extends previous work (Maier and Größler, 2000; Aarseth et al., 2003; Elverdam and Aarseth, 2007) in this field. Greco et al. classify BGs along five dimensions: application environment; elements of the user interface; target groups, goals, and feedback; user relation; and model. Each dimension is described by an enumeration of characteristics and sub-characteristics. One game can be categorized by a subset of characteristics. Below, Salt Seller is classified according to this taxonomy.

Application Environment. Salt Seller is a standalone simulation which means it is currently not

integrated into any learning environment such as an undergraduate teaching course. However, its modular implementation and high configurability facilitate an integration in the future. Salt Seller is played over a computer network (LAN, Internet) on the HTTP stack.

Elements of User Interface (UI). The user interface of Salt Seller is browser based and includes texts and charts. Gaming results and the complete decision making process are stored in a database for further investigation. The game behaves like a black box since the user is not supposed to have detailed information about the market mechanisms (= the simulation model and the choice of parameters). The game is round-based in contrast to e.g. a flight simulator where interventions within the simulation are almost continuous. Each round (period), the players take quantitative decisions (price for salt). Every player must enter his or her decision for the next period independent from each other within a certain time. The time frame to take a decision is synchronized between all parties which means the players act simultaneously. If one party does not respond within the time frame the game will be terminated so all parties have the same time to reflect and take a decision. In contrast to alternating rounds, e.g. chess, where one player decides after the other, decisions in the Salt Seller game are made simultaneously. Furthermore, since a timer counts down the remaining time to submit a price, haste is present; real time can alter the game state and remove one player from the game if he takes too much time for his decision.

Target Groups, Goals and Feedback. The goal of Salt Seller is to achieve a higher total profit than the opponent and to develop an understanding for the behaviour of a duopoly market. The goal is not absolute (fixed score on a scale) but relative to the opponent. It is easy to explain and thus the game is not restricted to a special target group as for example economy students. Salt Seller is supposed to be a teaching game and can also be used in research in order to analyse the strategies of the players. Target audience are undergraduate students or bright pupils with minor economic prior knowledge. So far it provides only basic feedback mechanisms like input validation or status messages. Debriefing is not provided either.

User Relation. In this implementation Salt Seller is a two player game and teams are not supported. The users are evaluated individually. Interaction between the users is indirect since they submit their price decisions independently from each other. There are no means of direct interactions, e.g. an integrated communication channel. Salt Seller has not the character of a role-playing game, where users take roles and interact socially with each other. Alliances do not exist in competitive two-person games. There is neither a players' nor a developers' community (like a Facebook page, mailing list etc.). It is open to the game moderator respectively the intention of the game session in question whether communication between the players is prohibited or not.

Model. The domain of Salt Seller is realistic in contrast to a fantasy game or a game which plays in the past or in the future. All parts of the game behave deterministically even though some components as the total salt demand could be modelled stochastically/randomly in the future (see section 8). The game is not influenced by external data such as exchange or inflation rates. It is highly configurable since all market functions of the model are parametrized and can be adjusted (see section 5).

4 GENERAL FRAMEWORK FOR *n*-player ROUND-BASED GAMES

In this section we introduce a general framework for *n*-player round-based games by utilizing inputs, outputs, fixed parameters, result functions, an objective function, and a stop condition. We will apply the Salt Seller game to this framework in section 5.

We define a *n*-player round-based game through the 6-tuple $(In_t, P, \mathcal{F}, Out_t, obj_t, sc)$ with $t \in T$ represents one round of all rounds T. We call a round also turn or period.

- In_t is a vector of inputs in turn t. E.g., in Salt Seller the prices the two players choose for their offer.
- P is a set of fixed parameters (constants), e.g. a pre-defined number of rounds.
- $\mathcal{F}$ is a set of result functions transforming the state from turn t to $t+1$. We provide examples for the Salt Seller game in section 5.
- Out_t is a vector of outputs of turn t calculated by inputs In_t, parameters P, and the result functions $\mathcal{F}$.
- obj_t is an objective function weighting the outputs related to the target of the game. In Salt Seller the objective function is the cumulative profit over all turns.

- *sc* is a stop condition. If the condition is met, the game ends. In Salt Seller the stop condition is a pre-defined number of turns.

Please note that in many games the players do not receive complete information about all game describing variables. Additionally, P, $\mathcal{F}$, and *sc* can be generalized and defined dependent on t as well, e.g. in Skat $\mathcal{F}$ is defined differently when playing "Null". Furthermore, all tuple parts could also been made dependent on the number of players n.

In each round, n players or a random number generator or a stochastic distribution submit $l = |In_t|$ input values which results in the input matrix In_t. We assume l constant in t. Thus, In_t is of the dimension $l \times n$. The domain of In_t has to be chosen depending on the game, generally, $\mathbb{R}$ can be a good choice. The output Out_t is a matrix of the dimension $k \times n$, $k = |Out_t|$ is the number of outputs for n players, again k is assumed constant in t. Like In_t the domain of Out_t is game-dependent.

The state transformation functions $\mathcal{F}$ are used to calculate the outputs of a round t. Each $f \in \mathcal{F}$ is defined as $f: In_t \times P \rightarrow Out_t$.

The result of the entire game for each player i is defined with the (linear) objective function $obj_t: (Out_t \times W) \rightarrow \mathbb{R}$. W is a weighting vector of dimension k.

It is:

$$obj_t = \sum_{s=1}^{t} \sum_{j=1}^{k} w_j * Out_{ji}^s, \text{ for } 1 \leq i \leq n, w_j \in W \quad (1)$$

5 SALT SELLER GAME MODEL

This section describes our game and market model of Salt Seller. It explains how the game's outputs of each round are calculated from its inputs. The model consists of three parametrized functions which are described further in the following three subsections. Subsection 5.5 shows how these functions work together in order to calculate the outputs of a game period.

5.1 Formal Game Model

Following the general game framework from section 4 we define Salt Seller as:

- $In_t = \{price_t^1, price_t^2\}$
  being two turn-dependent prices for players 1 and 2.
- $P = \{maxrounds, MC, SI, ZR, ZDP, ZDPr, MaxD\}$
  MC being the marginal costs, *SI* sensitivity index, *ZR* zero rate, *ZDP* zero delay percentage, *ZDPr* zero demand price, *MaxD* maximal demand. All parameters are covered in the next subsections.
- $\mathcal{F}$ is a set of 4 market functions: Market share (equation 5), Sales (equation 6), Profit (equation 7), and total salt demand (equation 4). We refer to the next subsections.
- $Out_t = \{$market share, sales, profit, total salt demand$\}$;
- $obj_t = \sum_{s=1}^{t} \sum_{j=1}^{k} (w_j * out_{ji}^s)$, for $i \in \{1,2\}$;
  with the weighting factor $W = (0,0,1,0)$ since only the output *profit* is maximized in Salt Seller's objective function. The objective function calculates the cumulative profit over all so-far player periods for a player i.
- $sc: t > maxrounds$
  The stop condition checks whether the current round t is greater than the pre-defined number of rounds *maxrounds*.

5.2 Market-Price-Sensitivity

This class of functions describes the market's sensitivity towards price differences between the two competitors. It thus provides a mapping between the price-ratio $\frac{p^1}{p^2}$ or $\frac{p^2}{p^1}$ and the market share of the first or the second player. Please note, we omit the index t for the sake of readability, however, all prices and *ms* (see below) are set per turn. In our model, the *market share* is defined as the continuous function $ms: [1,\infty) \rightarrow [0,1]$.

$$ms(x) = \begin{cases} \frac{1}{2} - zc_{ms} * (x-1)^{1/(2*SI-1)}, & 1 \leq x \leq ZR \\ 0, & x > ZR. \end{cases} \quad (2)$$

In this function, x is the price ratio of the higher price divided by the lower price ($\implies x \geq 1$) and $ms(x)$ is the market share of the competitor with the higher price. The zero coefficient zc and the Sensitivity Index *SI* are explained below. The market share of the competitor with the lower price is calculated as $1 - ms(x)$. We denote the market share of player i by ms^i.

For the salt seller game Sterman (Sterman, 2014, p.100) describes market-price-sensitivity as *"sensitivity of product attractiveness to price"*. This corresponds to the standard economic concept of the price elasticity of demand. For the general problems of measuring the price elasticity of demand we refer to (Wilson, 1997). One approach is by experimental markets, see e.g. for scientific information (Neumann, 2007). Function 2 is a simple approximation of the price elasticity of demand.

The rationale behind the function *ms* is the following: At equal prices ($x = 1$) the market shares are equal. We assume that a price ratio exists at which all customers buy from one player only. We call this the Zero Rate (ZR). The simplest price elasticity of demand function is linear. In order to provide a family of price elasticity of demand functions we chose power functions, parameterized by the second parameter *SI*. *SI* defines the speed of convergence. For $SI = 1$ the function reduces to a linear function.

Sensitivity Index (SI). This parameter defines the average sensitivity of salt customers towards price differences between the two salt producers. It is chosen from $\mathbb{N}$. The higher the value of *SI*, the higher the customers' awareness for price differences. A high value of *SI* thus signifies, that even small price differences will lead to a huge gain of market share for the price leader in the current period. At $SI = 1$, *ms* is a linear function.

Zero Rate (ZR). This parameter defines the price ratio from up to which the price leader will win the complete market in the current period. For instance, if $ZR = 3$ and $p^1 = 50$, $p^2 \geq 150$, the first competitor will have a market share of 100% in the current period. In figure 1 this is the crossing of the market share function *ms* with the x-axis.

Zero Coefficient (zc_{ms}). Find zc_{ms} so that $ms(ZR) = 0$ for a fixed *SI*.

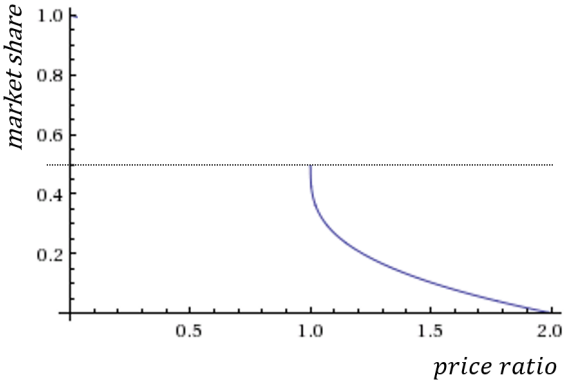

Figure 1: Market-price-sensitivity function for SI=2 and ZR=2.

Figure 1 shows a plot of *ms* for $SI = 2$ and $ZR = 2$. In this example, the derived zero coefficient zc_{ms} is $1/2$.

ms is easy to calibrate: It suffices to determine two points of the function *ms*. This can be done either by choosing proper values for *ZR* and *SI* or by experimental markets, as e.g. described in (Neumann, 2007).

5.3 Market Delay Percentage

To simulate a more realistic market behaviour the customers in *Salt Seller* are divided into two groups: The first group, the "*early adaptors*", reacts immediately to the latest price offers of the two salt producers. The second group, the "*conservatives*", reacts with a delay of one period which means their buying decision depends on the prices of the prior period (while actually paying the new prices). This provides a simplified model of a customer who once decided to buy from one producer (based on the price situation back then) and then just keeps on buying from the same producer, no matter how the prices develop.

The distribution of these two groups again depends on the price ratio. This follows the intuition that a very big difference in the prices for salt will alert even the most inattentive customer. The larger the difference between both prices, the lower the percentage of conservatives, the lower the mdp. In our model, the market delay is defined as the continuous function $mdp\colon [1,\infty) \to [0,1]$, with

$$mdp(x) = \begin{cases} (1-ZDP) - \\ \quad -zc_{mdp}(x-1)^2, & 1 \leq x \leq ZR \\ 0, & x > ZR. \end{cases} \tag{3}$$

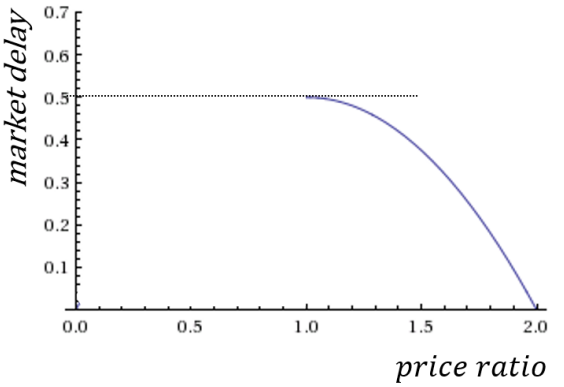

Figure 2: Market-delay function for ZDP=0.5 and ZR=2.

In this function, x is the price ratio defined as in the previous section and mdp is the percentage of the customers in the second group, thus $mdp(x)$ provides the percentage of the conservatives. The percentage of the early adaptors is calculated as $1 - mdp(x)$.

Using the decreasing part of a negative parabola function suited the simplified model of market delay well. It decreases only slowly where both opponents sell at similar prices. The more one player raises his price above that of his competitor, the faster the customers change groups and react to the big price

difference.

Zero Delay Percentage (ZDP). This parameter defines the minimum percentage of customers that belong to group one.

ZR is defined as in the previous subsection. zc_{mdp} is calculated by $mdp(ZR) = 0$. It determines the price ratio from up to which every customer will be part of the first group and nobody will buy from the more expensive producer anymore out of convenience. Figure 2 shows a plot of mdp for $ZDP = 0.5$ and $ZR = 2$.

5.4 Market Demand

The market demand of the current period is defined as

$$demand(p^{min}) = MaxD - zc_{demand} * p^{min} \quad (4)$$

where p^{min} is the price of the price leader. This is a linear function, zc_{demand} is calculated for a given choice of the parameters Zero Demand Price $ZDPr$ and Maximal Demand $MaxD$.

According to the article "The price of salt: Salt sellers"[1] there are very different types of salt which leads to prices between 40$ to 50$ per ton for de-icing salt and 70,000$ per ton for a French high quality salt. Furthermore the global consumption of salt is rising. In 2018 the forecast for global demand is 324 million tons per year[2]. To keep the simulation game simple a fictive multifunctional salt and linear market demand is used for the function above. Derived from the information about the salt market example values of 320 million tons per year for $MaxD$ and 960$ per ton for $ZDPr$ are supposed to be realistic. However, we are aware that salt is a essential good for every human being, thus assuming a Zero Demand Price is a bit unrealistic.

5.5 Output Calculation

Since market delay is taken into account the calculation of the market share of the current period does not solely depend on the market-price-sensitivity function $ms(x)$. Instead of that we have to calculate the sum of the outcome of $ms(x)$ with the price ratio of the previous period weighted with the percentage of the delayed customers from group two and the outcome of $ms(x)$ with the current price ratio weighted with the percentage of customers without delay from group one. Altogether we get for the market share MS of the price leader and $p^h = higher\_price$ and $p^l = lower\_price$ in period t:

$$MS_t = \underbrace{ms(\frac{p^h_{t-1}}{p^l_{t-1}}) * mdp(\frac{p^h_t}{p^l_t})}_{\text{“conservatives”}} + \underbrace{ms(\frac{p^h_t}{p^l_t}) * (1 - mdp(\frac{p^h_t}{p^l_t}))}_{\text{“early adaptors”}} \quad (5)$$

where $ms(x)$ and $mdp(x)$ are the market-price-sensitivity and market delay function from the previous subsections. The market share for the other salt producer in period t then calculates as $1 - MS_t$.

$$sales^i_t = MS^i_t * demand(p^{min}_t) * p^i_t \text{ for } i \in \{1,2\} \quad (6)$$

The sales are calculated by multiplying the market share with the total demand of salt which is determined using equation 4 and multiplying that value with the salt price of the player.

$$profit^i_t = sales^i_t - MS^i_t * demand(p^{min}_t) * MC \text{ for } i \in \{1,2\} \quad (7)$$

The profits can be determined by subtracting the production costs from that value. The production costs are calculated by again multiplying the market share with the salt demand and multiplying that value with the marginal costs which are another parameter of Salt Sellers's market model.

The player with the highest cumulative profit over all periods wins the game.

6 IMPLEMENTATION

Running in a browser and dealing with the multiplayer issues of a round-based game were the two main requirements for the implementation of *Salt Seller*. The scalable Python Web framework *Django* meets these requirements well.

Use of Python. Python is a lightweight scripting language with a clean and elegant syntax. Its big library and Object Oriented language model suites well for a wide variety of applications and so it does for web applications.

Model-view-Controller. Django follows the model-view-controller software paradigm which separates data model, data handling logic and user interfaces from each other. This brings the advantage that one

[1] http://www.economist.com/node/15276675, retrieved on 20 Feb. 2015

[2] http://www.reportlinker.com/p0788601-summary/World-Salt-Market.html, retrieved on 20 Feb. 2015

part can be developed and changed without affecting the other. The framework also takes care for the object-relational mapping (ORM) and introduces a separation between the data model and the database layer. Nearly all well-known relational database engines are supported and can be used without changing the data model.

6.1 Data Model

Figure 3 shows the data model of Salt Seller. Since it is a round-based game it consists of the classes *Game* and *Round*.

The class Game holds all the attributes which describe an instance of a game - like the values of the parameters chosen for this specific game or the users who are playing.

One game consists of as many rounds as specified by the parameter *maxrounds*. The class Round consists of the round-specific input and output attributes like the prices for salt, market shares, sales, profits, and total salt demand.

In its mapping from Python objects to relational databases each class of Django's model becomes a table and each class variable becomes a column in the database. Many-to-one relationships are translated into foreign key relationships. Each round holds a foreign key to one single game and since Salt Seller is integrated into KIT's infrastructure each game holds the two foreign keys of the two players playing. Valid users must be members of KIT.

Please note that the classes in Django's model are more than a sheer abstraction layer of the database. Round and Game define methods as in any Object Oriented language. A round for example has the method *calculate_round* which implements the result functions $\mathcal{F}$ from section 5. It is then part of the data handling logic (controller) to call this method of a round object at the right time - namely after both players have submitted their price decisions.

6.2 User Interface

Figure 4 shows the user interface of Salt Seller. On the left side the players can submit the salt price and read information about the last round as well as the marginal costs which stay the same during the entire game. On the right side the Google Chart API[3] is used for visualizing the salt prices and the game outputs.

[3]https://developers.google.com/chart/, retrieved on 20 Feb. 2015

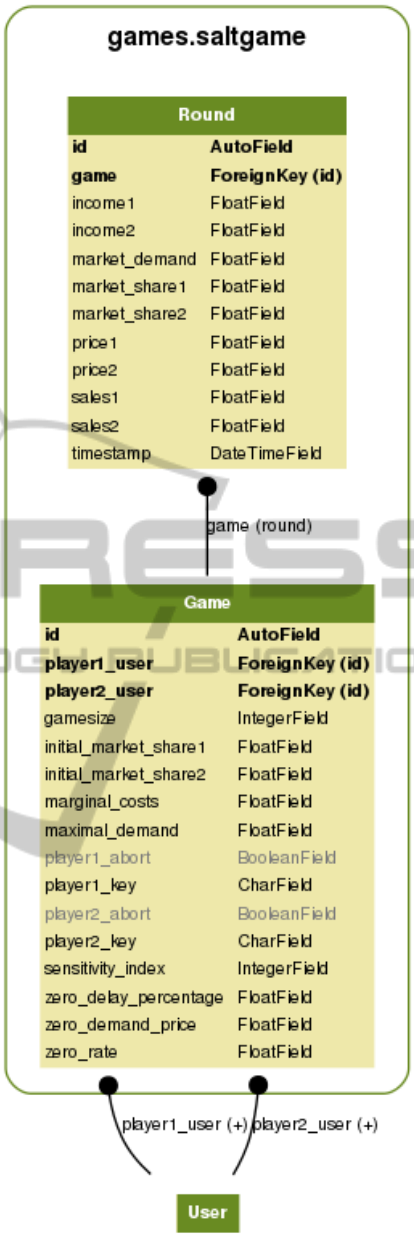

Figure 3: ER model of Salt Seller.

Location of the Game. The implementation described above can be found at the KIT web site.

7 PRELIMINARY EVALUATION

Whenever Salt Seller is played, the complete game history is stored to the database. This means that every decision of a player, every output it created, every parameter setting of the economic model as well as time and user information can be retrieved for detailed

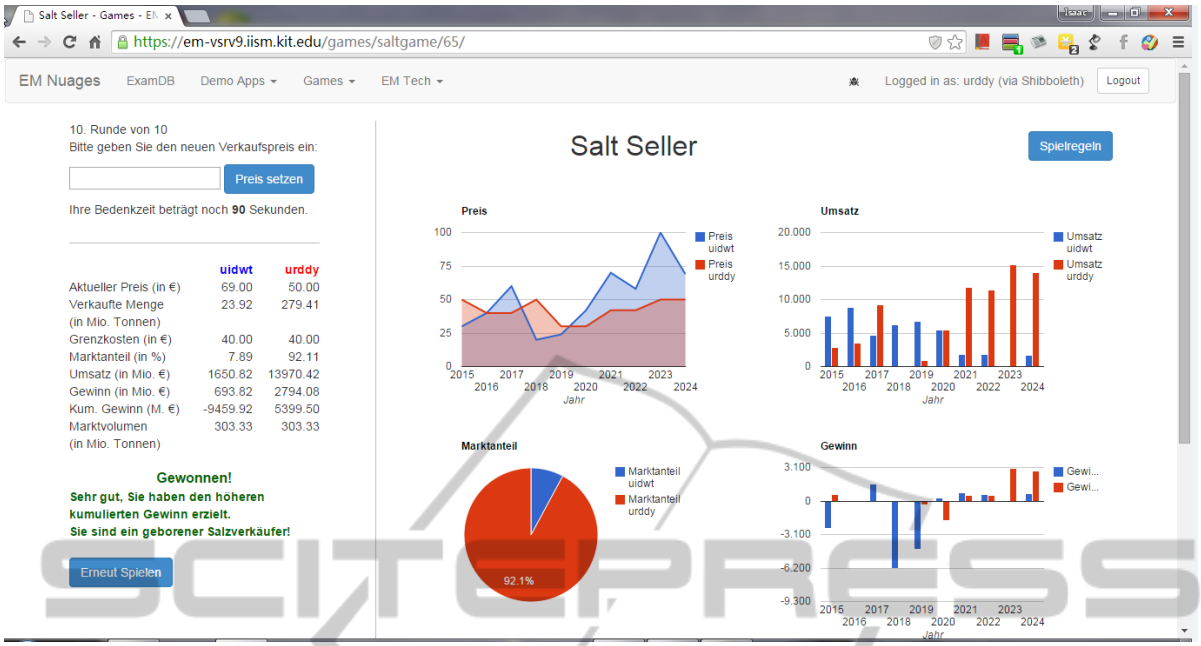

Figure 4: User interface (in German).

analysis later. So far, this capability was used for implementation issues and for validation of the market model. The next step is to use Salt Seller in class or in laboratory experiments. While the students profit from the lesson within a gaming environment, further research can be done with the collected data. It will be of use in order to analyse the strategies of the players as well as to observe strategy changes in response to the behaviour of the opponent. Since Salt Seller is already integrated into KIT's infrastructure, the data can be easily linked with further information about the player such as age, field of study and so on. If carried out in a laboratory, questionnaires can be used to gather further demographic, psychological, sociological, or strategy related information. Altogether, the collected data will contribute to the understanding of the behaviour of actors on a stable market. It hopefully allows us to identify players with different strategies (strategy families).

8 CONCLUSIONS

This paper gives a complete and detailed description and implementation of the Salt Seller business game of Sterman (Sterman, 2014). It shows how the stable market of Salt Seller can be modelled realistically considering effects like market delay or different price-market-sensitivities. Sterman's paper lacks a detailed explanation of market mechanisms, functions and parameters. A main contribution of our paper is to fill those gaps by providing these details. Our approach is fully parametrized which makes it highly configurable. We include a short discussion about parameters, reasonable parameter values and their influence on game progression. Aside, we classify the game following the scheme of Greco (Greco et al., 2013).

The second major contribution of this work is the proposition of a framework for a general n-person round-based game as a 6-tuple of turn-based inputs, fixed parameters, state transition functions, outputs, objective function, and a stop condition.

As third contribution we implemented the game as a browser game. Therefore a software architecture was designed supporting various 2-player, round-based, concurrently played, business games within a browser. The architecture enables a later analysis of all played games including details of input variables, parameters, and output variables of each round as well as the overall game result. The chosen architecture is light-weight and adaptable and can act as a blueprint for further 2-player round-based business simulations. Consequently the implementation of the game is available at the KIT web site for usage in University lectures, demonstration purposes and management trainings.

Further work could be done to extend the market model: Effects like the business cycle could be modelled by introducing stochastic/random behaviour. Apart from that we plan to integrate the Salt Seller game implementation into lectures for our

students. This will allow us to analyze played game strategies. However, before this the game needs to be embedded in a learning environment including manuals and instructions for players and facilitators. Furthermore, we think of including non-player characters, computer agents implementing machine learning algorithms for strategy learning. The architecture can be extended to support an *n*- instead of a 2-player base.

REFERENCES

Aarseth, E., Smedstad, S., and Sunnana, L. (2003). A multidimensional typology of games. *Level Up Conference Proceedings*, pages 48–53.

Bernard, R. R. (2014). Characterizing business games used in distance education. *Developments in Business Simulation and Experiential Learning*, 33:124–130.

Django (2015). Django web framework (version 1.5). https://djangoproject.com. [Computer Software; accessed 03-March-2015].

Eilon, S. (1963). Management games. *Journal of the Operational Research Society*, 14(2):137–149.

Elverdam, C. and Aarseth, E. (2007). Game classification and game design construction through critical analysis. *Games and Culture*, 2(1):3–22.

Faria, A. J. (1998). Business simulation games: Current usage levels - an update. *Simulation & Gaming*, 29(3):295–308.

Forrester, J. (1961). *Industrial Dynamics*. Pegasus Communications, MA, United States.

Greco, M., Baldissin, N., and Nonino, F. (2013). An exploratory taxonomy of business games. *Simulation & Gaming*, 44(5):645–682.

Hsu, E. (1989). Role-event gaming simulation in management education: A conceptual framework and review. *Simulation & Games*, 20(4):409–438.

Jarmain, W. E. (1963). *Problems in industrial dynamics*. M.I.T. Press, Cambridge.

Larréché, J.-C. and Gatignon, H. (1977). *Markstrat – A Marketing Strategy Game: Teaching Notes*. Scientific Press, Palo Alto.

Maier, F. H. and Größler, A. (2000). What are we talking about? - A taxonomy of computer simulations to support learning. *System Dynamics Review*, 16(2):135–148.

Neumann, A. W. (2007). Price elasticity of digital scientific information – a field experiment. *IADIS International Journal on WWW/Internet*, 5(2).

Ricciardi, F. M. and Marting, E. (1957). *Top management decision simulation: The AMA approach*. American Management Association.

Riedel, J. and Hauge, J. B. (2011). State of the art of serious games for business and industry. In *Proceedings of the 17th International Conference on Concurrent Enterprising (ICE), 2011*, pages 1–8. IEEE.

Rokicki, M., Chelaru, S., Zerr, S., and Siersdorfer, S. (2014). Competitive Game Designs for Improving the Cost Effectiveness of Crowdsourcing. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 1469–1478, New York, NY, USA. ACM.

Sterman, J. (2014). Interactive web-based simulations for strategy and sustainability: The MIT Sloan learningedge management flight simulators, part i. *System Dynamics Review*, 30(1-2):89–121.

Sterman, J. D. (1989). Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment. *Management Science*, 35(3):321–339.

Stratx Simulations (2015). Markstrat. http://web.stratxsimulations.com/simulation/strategic-marketing-simulation. [Online; accessed 03-March-2015].

Summers, G. J. (2004). Todays business simulation industry. *Simulation & Gaming*, 35(2):208–241.

Uskov, A. and Sekar, B. (2014). Serious games, gamification and game engines to support framework activities in engineering: Case studies, analysis, classifications and outcomes. In *2014 IEEE International Conference on Electro/Information Technology (EIT)*, pages 618–623.

Wilson, R. (1997). *Nonlinear Pricing*. Oxford University Press, Oxford.

Wolfe, J. (1993). A history of business teaching games in english-speaking and post-socialist countries: The origination and diffusion of a management education and development technology. *Simulation & Gaming*, 24(4):446–463.