

CAME – A TOOL SET FOR CONFIGURING ELECTRONIC MARKETS

Neumann, Dirk, University of Karlsruhe (TH), Information Management and Systems,
Englerstr. 14, 76131 Karlsruhe, Germany, dirk.neumann@iw.uni-karlsruhe.de

Mäkiö, Juho, University of Karlsruhe (TH), Information Management and Systems, Englerstr.
14, 76131 Karlsruhe, Germany, juho.maekioe@iw.uni-karlsruhe.de

Weinhardt, Christof, University of Karlsruhe (TH), Information Management and Systems,
Englerstr. 14, 76131 Karlsruhe, Germany, christof.weinhardt@iw.uni-karlsruhe.de

Abstract

Designing electronic markets is a difficult task. Due to the complexities inherent to the design, the approach of market engineering provides a structured engineering process that divides the design process into phases and recommends methods to solve the tasks within the single design phases. This paper introduces the tool workbench CAME for designing electronic markets. The workbench supports all design phases of the market engineering process, starting from the conceptual design that configures the appropriate auction rules to the detail design and implementation, which automatically generates the platform running the specified auction rules.

Keywords: Electronic Markets, Market Engineering, Auction Design.

1 INTRODUCTION¹

In literature, the origin of markets has long been discussed between economists. The supporters of the Austrian School argued that markets should evolve over time without interference of any central planner. Accordingly, *laissez-faire* of markets is assumed to be superior, as no central entity can have all relevant information about the economic environment that would be necessary to establish an adequate market institution attaining an overall objective (e.g. allocative efficiency) (Richter und Furubotn 1997).

Contradicting the Austrian school, the constructivistic school including the Economic Design favors designed market institutions. The constructivist epistemology stems from Descartes (also Bacon and Hobbes), who basically argued, “*that all worthwhile social institutions were and should be created by conscious deductive processes of human reason*” (Smith 2003, 467). Since the society itself has a particular function (e.g., achieving secure living), it is the aim of an institution to coordinate individual members in a way that that this function is achieved. Due to the discrepancy between individual and societal objectives, *laissez faire* among the individuals would result in inferior from a society’s point of view (McElroy 1998).

For traditional non-electronic markets, these two contradicting paradigms are still highly debated. A final resolution of this epistemological dispute is not to be expected. When turning the attention from markets to electronic markets, this implication cannot be kept upright. Electronic markets inherently require a conscious design of their institutional rules. The design of electronic market institutions is *indispensable*, as the (institutional) rules must be implemented in some sort of information system by some sort of market designer.

As the Austrian School suggests, designing electronic markets – understood as institution – is extremely demanding. This follows from the fact that the designer of the electronic markets wants to achieve a certain market outcome (e.g. resource allocation or individual revenues). This market outcome, however, is dependent on the behaviour of all market participants. By the trading rules implemented in the electronic market, the designer can affect not precisely control the behaviour. What makes it difficult is that the designer has incomplete or none information about the market participants’ internal decision making process, which determines the behaviour. The argument of *laissez faire* does, nevertheless, not solve the problem, since the institutions of electronic markets require design pertaining to the information system. Apparently, there is an obvious imbalance: On the one hand, the institutions of electronic markets *require* conscious design; on the other hand, conscious design turns out to be a practical extremely demanding endeavour. How can the electronic market be reasonably defined considering all interdependencies?

These questions gave rise to the development of a structured market engineering approach as a solution. Basically, market engineering is devoted to support designing electronic markets. At heart, the market engineering methodology is spawned around two pillars. The first pillar appears to be a design process model that decomposes design into less complex design tasks for which methods exist to solve them. As such, the second pillar of market engineering is the toolbox of particular methods (Weinhardt, Holtmann et al. 2003; Neumann 2004). As any other prescriptive design process model, the market engineering process is a recommendation, how to proceed, not a law. This recommendation character accounts for the needs of the designers. Creativity is typically not as straightforward, as the design process model would suggest. Behavioural studies have shown that designers often zigzag on different design levels leaving the prescriptive design path (Suh 1990). Any deviation from the design

¹ The work presented here is supported by the Transcoop program of the Alexander von Humboldt Foundation (AvH) and the German Ministry of Education and Science (BMBF).

process model, however, creates uncertainty concerning the overall design process, resulting in a trade off between creativity and certainty.

In the following, this dilemma between prescriptive character of the market engineering process and the designer's acceptance is addressed. In essence, the computer supported workbench CAME (stands for computer aided market engineering) is presented that implements the design methods at any given level of abstraction. Technically, this so-called computer aided market engineering workbench allows the automatic design of the electronic market. The value of this workbench can indisputably be increased by incorporating the design expert into the design process.

The paper is organized as follows. In Section 2, the market engineering methodology is reviewed with emphasis on the synthetic design steps leaving the analytical steps aside. In Section 3, the seamless computer-aided market engineering workbench is presented, highlighting the theoretical foundation and the implementation. Section 4 concludes with a summary and an outlook on future research.

2 MARKET ENGINEERING

As aforementioned, designing electronic markets that attains a specified objective (e.g. allocates the resources efficiently or maximizes the sellers' revenue) is a demanding task involving several activities. The market engineering methodology provides a discursive approach that divides the process into several, less complex phases. The right panel of Figure 1 sketches the higher-level stages of the market engineering process.

At the outset of the market engineering process stands the objectives of the electronic market and the strategy that governs the market engineering approach. In the first stage – the *clarification of the task* – the requirements of the envisioned new electronic market are deduced. This step is dubbed environmental analysis, as it comprises the specification of the economic environment, i.e. who are the potential customers, what are their preferences, endowments and constraints? As a result of the environment analysis, the designer is given the information about the requirements of the potential customers and of the market operator. The *design & implementation* stage is more or less a container for several design phases. Following the engineering design process (Pahl und Beitz 1984), the *design & implementation* stage is decomposed into four major phases being the *conceptual design*, *embodiment design*, *detail design & implementation* (see left panel of Figure 1). In essence the phases are concerned with the following:

- **Conceptual Design:** In the conceptual design phase the design problem is abstracted in a way that the design object is broken into its functions. As the design object is an electronic market, the abstraction has the function to allocate resources, provide the customers with information, enforce the allocation, and sue infringements, and so on. Those functions are further divided into sub-functions reducing the overall complexity of the design problem. Subsequently it is searched for solution principles that fulfil the sub-functions. Solution principles are basically economic effects in combination with the institutional rules that cause them. By doing so, the designer can generate alternative abstract descriptions of the institutional rules – the so-called *concept*.
- **Embodiment Design:** The concepts produced along the conceptual design phase are of abstract nature. For example, concepts define the trading rules in terms of the offer types available to the agents or the computation of offers into allocations and prices, but they do not exactly specify the flow of offers in detail. As such, different trading protocols can be used to implement the same conceptual representation of the trading rules. Embodiment design is thus primarily concerned with developing a *blueprint* that refines the concept into (semi-) formal descriptions of the institutional rules. The blueprint is thereby intended to allow an implementation, but is itself free of implementation details.
- **Detail Design & Implementation:** The detail design phase starts out with the blueprint, which describes the central aspects of the system, but is still at a level that is not implementable. Detail design further refines the blueprint into a fully-fledged system model that is subsequently implemented into a *preliminary electronic market*.

The peculiarity of market engineering process is embedded in the stage “design & implementation”. Having implemented the appropriate electronic market transaction service, it is tested upon its economic properties and its operational functionality. Those services that surpass the testing are subsequently introduced into the market. At any stage of the market engineering process there is a decision, whether to proceed with the next step or better to repeat the prior one. The use of prototypes is again possible at any stage of the process, such that they are left out in the picture.

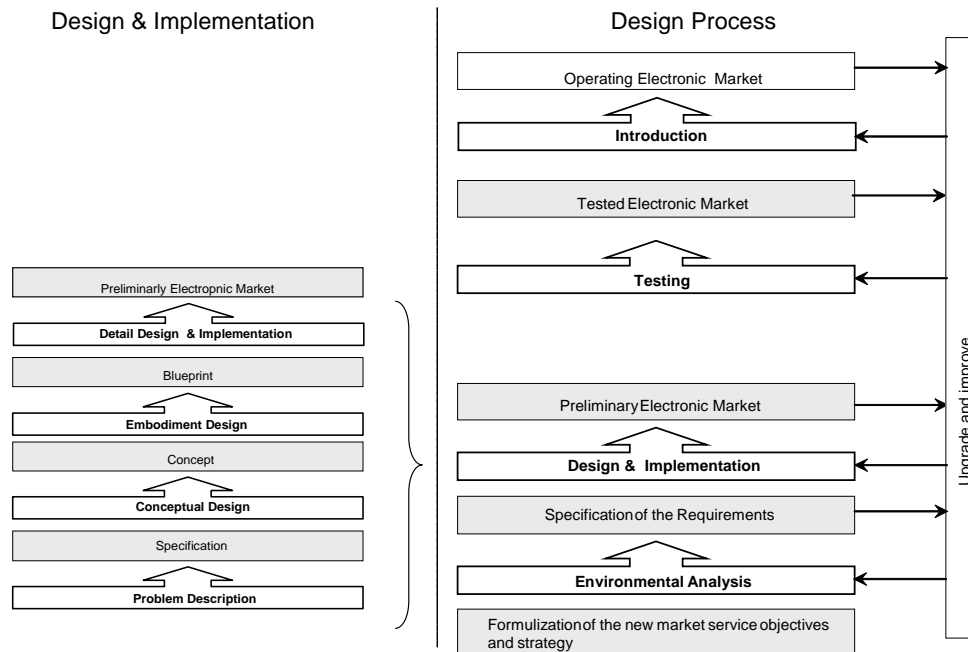


Figure 1: Stages of the Market Engineering Process

3 COMPUTER AIDED MARKET ENGINEERING WORKBENCH

Market engineering comprises several different tasks, which are inherently interdisciplinary. The strategic task of defining the segment in which the electronic market is intended to operate is on the one hand a management and marketing endeavour (Fennell und Allenby 2003); the design of the institution meaning the trading rules, which describe the flow of the negotiation process, is on the other hand pertains to economics (Roth 2002), whereas the implementation of the trading rules into a running information system is a software engineering problem. The market engineering process alleviates the interface problem by defining the documents, which are result of the respective phases and the procedure how to develop those documents. The integrated computer-aided market engineering workbench strives now for automating these procedures beginning with the design of the trading rules and ending with the implementation.²

The computer aided market engineering CAME workbench comprises several components: At heart of the workbench is the generic auction server core. This component instantiates any conceivable trading rule configuration, which is part of the design space. The core is fed by the trading rule configurator. With these two components at hand, the technical problem of the market engineering approach is addressed. What is missing appears to be a decision support system, which prescribes how the trading

² In other words, the following elaboration assumes that the management and marketing problem of defining the segment and eliciting the requirements the electronic markets must satisfy are already completed.

rules *should* be in order to attain the desired goal. The decision support system stores and makes economic design knowledge accessible by the user of the workbench.

To assume that the decision support system would produce one, unanimous trading rule set, is, however, myopic. Taking into consideration that the designer has not complete knowledge about the economic environment, which also contains the determinants of individual decision making behaviour (e.g. risk attitude), it is clear that the trading rules the decision support system constructs are both, uncertain and often contradictory. This problem is imminent to market engineering and *cannot* be removed. What can be done is to test the proposed trading rules in laboratory experiments and simulations. The CAME workbench is coupled with an experiment and simulation shell, which generically assist the set up of simulations and experiments. As such, the workbench can assist in this (preliminary) testing endeavour as well. Once the decision support system suggests a trading rule set, it can be instantaneously instantiated on the auction server and tested by referring to the experiment and simulation shell.

In the following, the decision support system, the configurator and the generic auction server are closer observed referring to their position in the market engineering process.

3.1 Conceptual Design

As aforementioned, the conceptual design phase decomposes the entire solution into functions, which can further on be tackled on an abstract level. Abstraction serves for concentrating on the creation of original solution rather than imitating already existing (me-too) solutions.

The CAME workbench focuses at the moment on the core functionality of the electronic market only, the transaction service (including auctions with price discovery). This function can be fulfilled by a verbal description of the trading rules as abstract solution.

One particular class of abstract solutions for this function refers to auctions. Auctions are straightforward, as they all share a common structure (Wurman, Wellman et al. 1998b). Principally, an auction follows through several phases from bidding to resolution. Each respective phase is characterized by a number of parameters, which describe the activities of the phases in a rather generic manner. For instance, a parameter could be the existence of an allocation rule, which specifies which agent receives what goods depending on the submitted offers. For each parameter a set of concrete rules is given. A concrete rule can for instance demand that the agent with the highest offer is awarded with the good. As such, an auction is defined by the assignment of concrete rule to the parameters (i.e. parametric design). Following this parametric approach, the design space of all auctions that can be constructed using this framework is defined. Having described the solution format of the conceptual design phase, it is necessary to define the reasoning behavior of the decision support system that justifies design recommendations.

3.1.1 Knowledge Representation

Before the reasoning behavior can be described, it is helpful to review the ‘big picture’ of electronic markets. Essentially, trading rules affect the outcome of the electronic market dependent on the economic environment. In other words, the trading rules exert an effect on the outcome. It is well accepted that these *effects* are “*remarkably robust*” (McAfee und McMillan 1996), such that it is possible to base the conceptual design upon those effects justifying the knowledge based approach. Obviously, market engineering is less stringent than its prototype mechanical engineering, since the later founds their design upon physical effects, which are caused by natural laws (Little 1993). Market engineering, on the contrary, is dealing with social effects reflecting social regularities.

In essence, an effect captures the impact a set of trading rules in a specific environment has on the outcome. Formally, it is possible to represent the environment, the trading rules and the outcome by parametric description. This parametric description allows the convenient storage of effects. The

antecedent of the rule comprises a description of the economic environment that is required and the set of trading rules. The consequent of the rule is the market outcome.³ A short example may clarify the rule definition.

Example

Suppose a painting is sold over the eBay platform. In this example, the environment is characterized by the following description: one item is for sale by one seller to many potential buyers. Furthermore, the elements of the environment pertaining to the individual decision making behavior can in this example be given by reference to independent private valuations.⁴ Furthermore, the valuations among the buyers are statistically independent; buyers are risk neutral. The trading rules are described by an open, ascending bidding format with a hard close.⁵ The buyer who has submitted the highest offer receives the item for the price he has posted.

What effects are, most likely, occurring in such a situation? In other words, what effects are applicable on this environment – trading rule combination? The first effect, which contains the environment-trading rule as antecedent, suggests that the participating buyers will incrementally bid up to their valuation and then drop out. Since it is assumed that many buyers will take part the consequent of this rule is that the revenue will be relatively high compared to a benchmark (marked by the independent private value model) (see for example Wolfstetter 1995). There is, however, another countering effect applicable. This countering effect suggests that many buyers will attempt to snipe, i.e. to bid in the very last moment of the auction. Sniping can result in relatively lower revenues, as some buyers may not have the chances to improve their offer. Apparently, there are two inconsistent effects applicable for the same antecedent.

As this example suggests the effects applicable for a certain environment-trading rule combination are often inconsistent with each other. To remove those inconsistencies the reasoning behavior also stores rules that are applicable for certain effects that are defined on the same antecedent. The so-called overruling function determines the combined effect that most likely occurs when two or more effects interact. More specifically, the antecedent of the overruling functions comprises the contradictory effects and the consequent is the outcome of the combined effect. Recall that in our previous example, the two effects were contradicting each other. From empirical observation in eBay it is known that the sniping effect prevails. This empirical observation can be captured in an overruling function. Comprising, those two rule types, effects and overruling functions, can be used to predict a market outcome the trading rules would achieve.

3.1.2 Design Approach

At the core of the design approach stands the parametric structure of the trading rules, which reduce the search space considerably. Several design methods exist that provide heuristic search strategies for parametric design problems. In the following a simplified *propose-and-revise* method is adopted.

³ Thus, a rule R could look like as follows: $R(\{\text{verbal description of the environment}\}, \{\text{verbal description of the trading rules}\}) \rightarrow \text{Outcome}$.

⁴ This term circumscribes that all buyers have a valuation for the painting in monetary terms. This valuation, however, is only known to the respective buyer reflecting private values.

⁵ As such the verbal description of the trading rules would be of the format (Biddinglanguage={open}, BiddingDirection={ascending}, StoppingRule={hardclose}), where the terms Biddinglanguage, BiddingDirection and StoppingRule refer to the parameters of any auction and the strings assigned to those rules are the concrete, {known} rule specifications. This example is overly simplified – the complete parametric description comprises sixteen different rules. A full treatment of the rules can be found at (Neumann 2004).

At the outset of the design problem the designer has two pieces of information, being an incomplete description of the economic environment and the desired outcome (e.g. high revenue for the seller). Now the problem is to define a set of trading rules, which implements the desired outcome within the context of the environment. Apparently, the environment is greatly underspecified. This is inevitable since there are unobservable elements that are simply unknown to the designer and cannot be obtained (e.g. determinants of individual decision making behavior). Clearly, this lack of information hinders the design tremendously, but this is inherent to any market engineering endeavour.

The idea used here to parametrically design a verbal description of the trading rules – the concept – is the following: Firstly, all effects that pertain to the observable socio-economic environment are extracted from the knowledge base. These effects denote all possible reactions that are known with respect to that observable environment. Then, only those effects are selected that yield the desired outcome. Since any of those effects is associated with parts (elements) of the trading rules the auction designer becomes an idea what particular trading rules to use. Ideally, these effects together render an entire description – this will, however, not be the case. Some elements will not be specified at all, as the effects do not consider them. Other elements – on the contrary – will be specified in different ways by different effects. Now the auction designer has to select as many elements of the trading rules as possible. Since those elements are not fully specifying the trading rule set, the market engineer has to complete it in a way that all parameters are assigned to a corresponding parameter. Subsequently, the verification of this generated verbal description of the trading rule set starts.

With the complete verbal description of the trading rule set at hand and the description of the environment, the market engineer can extract all effects that may occur. Subsequently, it can be checked whether these effects contradict the desired effect entailing the outcome. Typically, contradictions occur, as the environment is underdetermined (recall that the unobservable information of the environment is missing). Underdetermined environments entail that more effects principally occur. As especially the unobservable part is missing, the effects are more than likely contradicting.

These contradictions can be classified into two categories. In the first category, the contradictions occur within the same environment. By means of laboratory experiments those contradictions can be resolved. In the second category, the contradictions may stem from variations in the unobservable socio-economic environment. Laboratory experiments can only resolve the robustness of these effects in different environments – they, however, do not help in the particular engineering problem, as laboratory experiments require the induction of the environment. Notwithstanding, the auction designer simply does not know the unobservable environment.

Note that the decision support system has not yet been implemented. The implementation, however, is expected to create less problems, as the parametric design method as well as the domain knowledge representation comply with the task and domain layer of the CommonKADS specifications, the European de-facto standard methodology for designing knowledge-based systems (Schreiber, Akkermans et al. 2001).

3.2 Embodiment Design

In the embodiment design phase, the resulting parametric description of the trading rules is refined into a computer readable language, called market modelling language or MML, which is in the subsequent implementation phase used for instantiating a running electronic market (see Figure 2).

In essence, the conceptual design phase creates an abstract description of the trading rules. It is intended that market design experts are provided with the necessary description, which they understand. Revisions in the early phase of market engineering are very likely to occur, as there may be several trading rule descriptions that would satisfy the requirements. Note that the verbal parametric description generated along the conceptual design phase is incomplete in a sense that information about the *exact* flow of information is not incorporated. This missing information is intentional, as the concept – the parametric description – focuses on a very abstract solution of the

problem. Since domain theory, namely economics, cannot distinguish small differences in the auction process⁶, they are correspondingly left out for building the concept. For implementation purposes, these missing details (e.g. tie breaking rules, currency of the offered prices, minimum tick size) become relevant and must be added along the embodiment design phase. Further, the exact time-out of a hard closed auction must be specified.

The configurator merely codes the verbal description of the trading rules into a XML-schema based language that resembles the same structure as the previous verbal description.

Computer-aided configuration of the trading rules

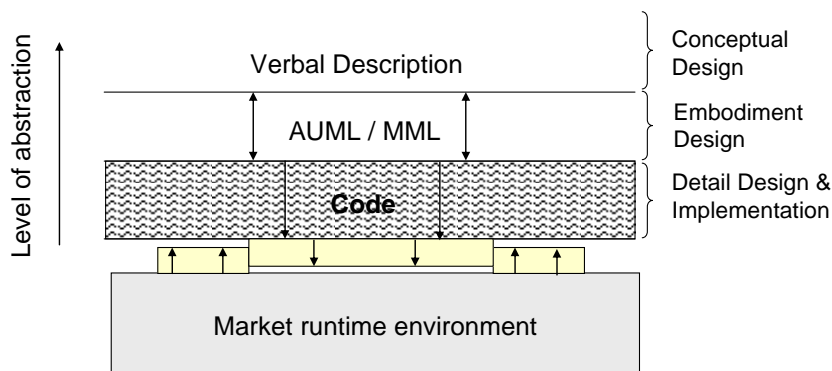


Figure 2: Computer-aided configuration of the trading rules

As Figure 2 illustrates, the MML itself is neither runnable nor does it produce code. The idea adopted here refers to the parametric structure of auctions. Recall that all auctions share a common structure. This common structure gave rise to the development of a generic auction process that describes on an abstract level this common process.

Figure 3 exhibits this generic process by using agent interaction protocol (Bauer, Müller et al. 2001). Almost any known auction⁷ follows along these lines: First of all, having initiated the auction, the participants may receive additional information (e.g. number of items to be sold, etc.). Then, the bidding phase commences, where the participants can – depending on the rules – submit one or many offers or alternatively express that they have not fully understood the call for auction. Having received offers, the electronic market system checks them, and, tentatively converts offers into scores or prices, checks the provisional winner, resolves ties and subsequently verifies the termination condition (see the activity diagram in Figure 3). Then, it is checked whether the termination condition is satisfied. If so, the participants are informed about the auction outcome. This process is generic but not concrete as the parameters of the auction are not specified. In Figure 3 the bold terms represent the parameters that must be defined by the MML. Basically, all information feedback the participants receive, the form of the offers and the algorithms of winner determination and so forth are not given by the generic process. Also the cardinalities, who receive what messages is subject to the concrete auction. Apparently, by means of the MML those missing components are filled transforming the hard-coded generic process into a concrete, running auction.

⁶ In economics the auction types (e.g. English Auction) rather specifies a class of different auction types than a single auction type (Wolfstetter 1995). For instance, a tie-breaking rule is strictly speaking not defined. In theory, this is unproblematic as long as the valuations of the participants are drawn from a continuous distribution, where ties (same valuations of two or more participants) occur with the probability of 0.

⁷ In fact, there are few auctions that are not complying with these generic process (e.g. Anglo-Dutch auction (Klemperer 2002)). Apparently, these auctions cannot *directly* be configured by using the parametric approach.

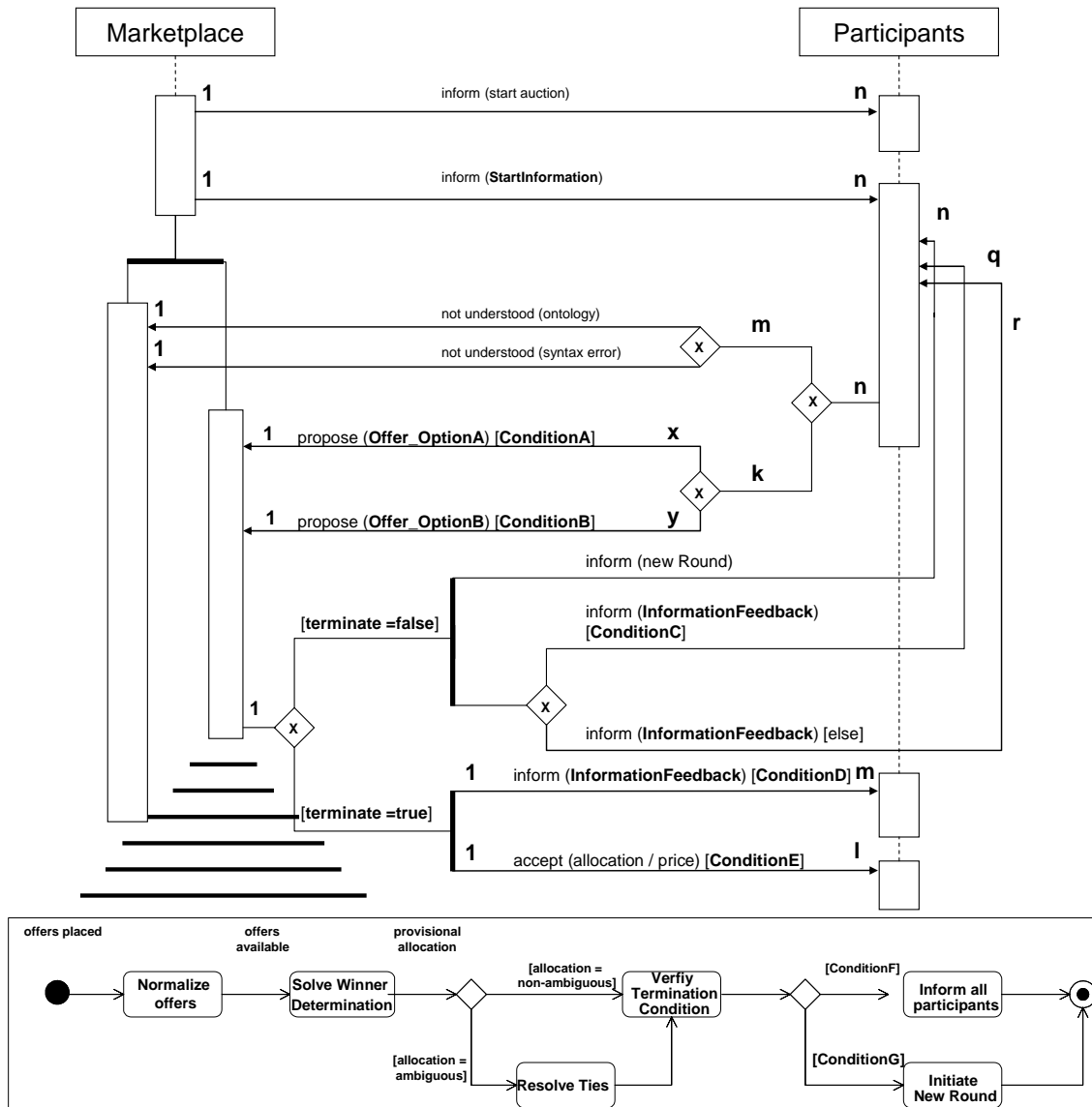


Figure 3: A generic auction process

3.3 Detail Design and Implementation

The initialization of an auction described by the MML instance is straightforward. The elements of the MML instance are plugged into the generic process, which they both describe and configure for one specific auction. Additionally, they describe further elements needed for the auction, like information feedback and fees.

The entire of MML currently contains a total of 279 elements, which are mainly responsible for specifying the generic auction process (e.g., offer description, and communication between marketplace and participants – see Figure 3). In the following the information feedback is used as an example for specifying an electronic auction. When modelling the information feedback, various issues have to be considered, like timing (e.g. clocked, continuously, event based), transparency (e.g. to whom the current status is made available), and content (e.g. displayed information).

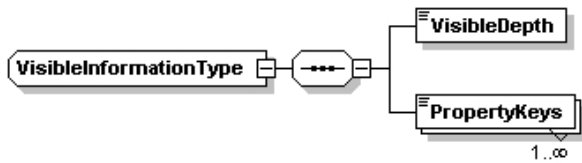


Figure 4: Information feedback from the order book

For the content modelling the relevant attributes are *VisibleDepth* and *PropertyKeys* (c.f. Figure 4). These attributes are used to define the information that is presented to the market participants (this refers to the argument of the inform primitive in Figure 3). The first attribute, *VisibleDepth*, defines the number of order visible to the participants and the latter attribute, *PropertyKeys*, determines which information from the orders is visible. Note that the information feedback can be given for each side (buy and sell) separately.

The initialization of new auctions is automatically executed. For the initialization the MML-description of the auction is analyzed and according to the description, the trading rules are generated. Technically, the market model initialization interface supports not solely MML files, but also further data structures, like Hashtables. Thus, a new auction can be initialized by using the MML description of it or, like in following example done, by using other data structures containing the required auction parameters. The component *AuctionGenerator* is responsible for the generation of new auction instances. A newly generated auction is deployed automatically into the auction server. The component *AuctionManager* is responsible for the deployment and management of new auctions. The generated auction is available, whereby the availability of the auction depends on the starting and stopping rules of the particular auction. Following code fragment demonstrates how these steps are implemented in the auction server (see Meet2Trade 2004).

```

Auction auction = AuctionGenerator.generateAuctionFromDescription(auctionDes);
AuctionManager.deployAuction(auction);

```

As aforementioned, the parameterization of the current trading rule set is performed by the *AuctionGenerator*. The concrete steps are briefly demonstrated referring to the following code excerpt. Note that in the code fragment the *InstanceFactory* is implemented according to the abstract factory design pattern that is used to construct complex objects (c.f. Gamma, Helm et al. 1995). Firstly, an auction template is generated by the method `generateAuctionDescription`:

```

public void generateAuctionFromDescription(Hashtable auctionDes){
    AbstractAuction abstrAuction =
        InstanceFactory.createAuction("Auction", (String)auctionDes.get("auctionName"));

```

Subsequently, the following steps are necessary to configure the auction template. Firstly, the matchmaking component (“matcher”) is generated by the *InstanceFactory*. Secondly, the attributes determining the scoring rule for the matchmaking (e.g. price, volume, delivery time) are specified. According to these attributes all feasible matching offers of the auction are identified.

```

    abstrAuction.setMatcher(InstanceFactory.createMatcher(auctionDes.get("matcher")));
    abstrAuction.setMatchingProperties(auctionDes.get("matchingProperty"));

```

In the next step the allocation algorithm for the current auction is determined. To do so, the allocation component (“allocator”) identifies the winning participants and computes the corresponding contract (e.g. price, volume, delivery time).

```

    String all = getAllocatorName(auctionDes.get("allocator"));
    abstrAuction.setAllocator(InstanceFactory.createAllocator(all, abstrAuction, 0));

```

The information feedback is determined by defining the information shown to the trading participants (c.f. Figure 4). It can be individually configured for each participant or for each group of participants. Participant groups can be built according to the roles they take in the market process. The current example shows the easiest case defining information feedback for two roles, buyers and sellers as well as the number of order visible to the participants. Note that this example merely models the content.

```
abstrAuction.setInformationFeedbackBuyer(auctionDes.get("buySide"));
abstrAuction.setInformationFeedbackSeller(auctionDes.get("sellSide"));
abstrAuction.setInformationDepth(new Integer((String)
    auctionDes.get("auctionInfoDepth")).intValue());
```

In the next step the starting and stopping rules are defined. In this example, time-based rules are used in order to start or to stop the auction at a particular point of time. At the technical level the reflection mechanism of the Java programming language is used, which allows invoking methods of one object solely by their name and parameters. This implies that by giving the name of the method (“*startAuction*” or “*stopAuction*”) these methods can be invoked automatically at the given point of time (“*startTime*”).

```
abstrAuction.addNotification("startAuction", (auctionDes.get("startTime")));
abstrAuction.addNotification("stopAuction", (auctionDes.get("stopTime")));
```

Having specified all detail information of the generic auction process, the auction can be instantiated and deployed, concluding the market engineering process.

4 CONCLUDING REMARKS

Designing electronic markets is not only a software engineering problem it is also a *social engineering* problem. The idea of the electronic market software is to attain a desired market outcome. Since the market outcome is created by the interaction of the market participants, the designer of electronic markets needs to influence the behaviour of market participants. This is a very difficult problem, as the economic sociologists have argued (Granovetter 1985; DiMaggio und Louch 1998). Many well thought electronic market platforms have failed, because the market participants reacted in an undesired way, exposing flaws in the design or did not show up on the market at all (Day, Fein et al. 2003). This is an inherent problem to any social engineering activity, as anticipating the participants’ behaviour is extremely difficult. The market engineering approach provides a methodology that combines technical, economic as well as social aspects. This paper extends the market engineering methodology by a workbench of tools that automatically support the configuration of electronic markets.

The presented approach of computer-aided market engineering is unique in a way that the entire market engineering process is supported. Typically, generic market servers such as GNP (Benyoucef, Keller et al. 2000) or the AuctionBot (Wurman, Wellman et al. 1998a) provide a toolbox for configuring auctions or negotiations. They, however, do not support any insight what auction type *should* be used. Even small details (e.g. the information feedback, or the stopping rule) totally change the behaviour of the market participants and thus of the outcome. As such, those purely technical approaches lack of an adequate decision support. The presented approach provides both decision support and an adequate implementation founded on the common basis of a parametric approach.

The parametric approach incurs, however, restrictions in the modelling variety; auctions that do not comply with the structure required by the parametric approach cannot directly be modelled. The same holds for more complex negotiation protocols that are not consistent with the parametric approach. Having in mind that especially B2B commerce demands richer negotiation protocols (Lomuscio, Wooldridge et al. 2003), it is challenging to extend the presented approach by relaxing the strict structure of auctions.

References

- Bauer, B., J. P. Müller and J. Odell (2001). "Agent UML: A Formalism for Specifying Multiagent Software Systems." International Journal of Software Engineering and Knowledge Engineering **11**(3): 1-24.
- Benyoucef, M., R. K. Keller, S. Lamouroux, J. Robert and V. Trussart (2000). Towards a Generic E-Negotiation Platform. Sixth International Conference on Re-Technologies for Information Systems, Zurich, Switzerland 95-109.
- Day, G. S., A. J. Fein and G. Ruppertsberger (2003). "Shakeouts in Digital Markets: Lessons from B2B Exchanges." California Management Review **45**(2): 131-150.
- DiMaggio, P. and H. Louch (1998). "Socially embedded consumer transactions: For what kinds of purchases do people most often use networks?" American Sociological Review **63**(5): 619-637.
- Fennell, G. and G. M. Allenby (2003). "Specifying your Market's Boundaries." Marketing Research **15**(2): 32-37.
- Gamma, E., R. Helm, R. Johnson and J. Vlissides (1995). Design patterns: elements of reusable object-oriented software. Reading, MA, Addison Wesley.
- Granovetter, M. (1985). "Economic Action and Social Structure: The Problem of Embeddedness." American Journal of Sociology **91**(3): 481-510.
- Klemperer, P. (2002). "What really matters in Auction Design." Journal of Economic Perspectives **16**(1): 169-190.
- Little, D. (1993). "On the Scope and Limits of Generalizations in the Social Sciences." Synthese **97**(2): 183-208.
- Lomuscio, A. R., M. Wooldridge and N. R. Jennings (2003). "A Classification Scheme for Negotiation in Electronic Commerce." Group Decision and Negotiation **12**(1): 31-56.
- McAfee, P. and J. McMillan (1996). "Analyzing the airwave Auction." Journal of Economic Perspectives **10**(1): 159-175.
- McElroy, W. (1998). "Human Ignorance and Social Engineering." The Freeman **48**(5): 278-282.
- Meet2Trade (2004). Meet2Trade - Technical Report. The Generic Server. Karlsruhe, University of Karlsruhe.
- Neumann, D. (2004). Market Engineering - A Structured Design Process for Electronic Markets. Fakultät für Wirtschaftswissenschaften. Karlsruhe, Universität Karlsruhe (TH).
- Pahl, G. and W. Beitz (1984). Engineering Design. Bath, UK, The Pitman Press.
- Richter, R. and E. Furubotn (1997). Institutions and Economic Theory: The Contribution of New Institutional Economics (Economics, Cognition, and Society). Ann Arbor, University of Michigan Press.
- Roth, A. E. (2002). "The Economist as Engineer: Game Theory, Experimental Economics and Computation as Tools for Design Economics." Econometrica **70**(4): 1341-1378.
- Schreiber, G., H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde and B. J. Wielinga (2001). Knowledge Engineering and Management - The CommonKADS Methodology. Cambridge, MA, MIT Press.
- Smith, V. (2003). "Constructivist and Ecological Rationality in Economics." American Economic Review **93**(3): 465-508.
- Suh, N. (1990). Principles of Design. New York, Oxford University Press.
- Weinhardt, C., C. Holtmann and D. Neumann (2003). "Market Engineering." Wirtschaftsinformatik **45**(6): 635-640.
- Wolfstetter, E. (1995). "Auctions: An Introduction." Journal of Economic Surveys **10**(4): 367-420.
- Wurman, P., M. P. Wellman and W. E. Walsh (1998a). The Michigan Internet AuctionBot: A configurable auction server for human and software agents. Second International Conference on Autonomous Agents, Minneapolis 301-308.
- Wurman, P., M. P. Wellman and W. E. Walsh (1998b). "A Parametrization of the Auction Design Space." Games and Economic Behavior **35**(1-2): 271-303.