A HIERARCHICAL SIMULATION BASED SOFTWARE ARCHITECTURE FOR BACK-TESTING AND AUTOMATED TRADING

Arne Koors and Bernd Page
Department of Informatics
University of Hamburg
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
E-mail: {koors, page}@informatik.uni-hamburg.de

KEYWORDS

Simulation, Software Architecture, Back-Testing, Automated Trading Systems, Financial Markets

ABSTRACT

Financial markets are highly complex adaptive systems. This paper deals with the application of simulators in software architectures for back-testing and automating financial market trading strategies. It characterizes traits and problems of algorithmic trading and describes the established use of simulators in back-testing and automated trading. A new approach in the form of a hierarchical software architecture is introduced, containing simulators as integral parts in all layers, using them both during back-testing and automated trading. In addition to the software architecture the opening objects of investigation are outlined. Finally, the potential of generalizing the application domains of our approach beyond financial market trading strategies is pointed out.

INTRODUCTION

Financial markets are highly complex adaptive systems (Maboussin 2002, Darley and Outkin 2007, Haldane 2009), where a multitude of institutional and individual investors exchange financial goods like stocks, bonds, currencies or commodities. The financial markets provide liquidity as well as buy and sell quotes, enabling market participants to find trading partners, exchange asset valuation and to finally agree upon a trade price. The effective trade prices are recorded as price histories, remaining accessible in a machine-readable form, as a basis for subsequent research.

Simulation in Financial Markets

Dynamic simulation has been applied to the field of financial markets mainly in two ways:

1. Description, reproduction, explanation, organization and forecasting of market behavior as a whole.

   Beginning with macro-economic market models in continuous system dynamics style (Sharp and Price 1984), recent research has emphasized the role of individual market participants. Thus modeling of financial markets has turned towards discrete event models (Jacobs et al. 2004) and shifted to multi agent based approaches (Arthur et al. 1997, Lux and Marchesi 2000, Levy et al. 2000, Hommes 2006, LeBaron 2006), especially during the last decade.

2. Research, test and optimization of investors' trading strategies.

   Independently, academics, financial institutions and individual investors have analyzed and simulated numerous trading strategies in order to answer questions on the profitability and risk of systematic trading.

This paper addresses the second aspect of application of simulation methods in financial markets.

Algorithmic Trading

Around 1900, Charles Dow published the assumption that financial markets quickly discount for all news, and thereby prices reflect the available information correctly. His further hypothesis that prices do move in trends with certain characteristics lead to the development of technical analysis (Hamilton 1922, Rhea 1932, Schaefer 1960).

Technical analysis solely relies on market price histories and examines price series for trends, patterns, anomalies, etc. in order to support – or suggest – investment decisions. As this is a pure quantitative task, computers have increasingly been used in the field of algorithmic investment analysis and decision making over the last thirty years. Today, automated trading accounts for 61 percent of the U.S. stock market activity and 70 percent of individual trades (Kearns et al. 2010).

Back-testing

For the purpose of this paper, a trading *strategy* is a formally specified, systematic sequence of actions at financial markets. An *automated trading strategy* is implemented algorithmically by a computer, without discretionary influence or other – intuitive – human interventions.

Before put into action in real markets, trading strategies are generally back-tested against historical price series. This makes it possible to determine statistical perform-

Proceedings 25th European Conference on Modelling and Simulation ©ECMS Tadeusz Burczynski, Joanna Kolodziej Aleksander Byrski, Marco Carvalho (Editors)
ISBN: 978-0-9564944-2-9 / ISBN: 978-0-9564944-3-6 (CD)

ance measures of a strategy, e.g. the historical risk/return profile, without losing real money.

Back-testers are an application-specific type of discrete event simulators designed to develop, offline-test, debug, evaluate and optimize financial market trading strategies under conditions nearly identical to real markets. Finally, they allow to connect a tested strategy to an online broker or to an exchange in order to have it automatically traded.

Modern commercially offered back-testers are typically structured as shown in Fig. 1 (Kocur 1999, SmartQuant 2006, Rightedge 2010):

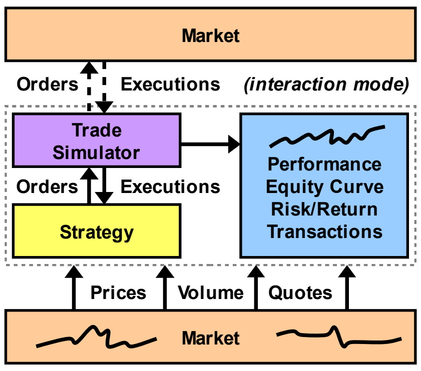

Figure 1: Common Architecture for Back-testing and Automated Trading

A strategy formulated in a proprietary or general programming language is provided with market data. This includes trade prices, trading volume or ask and bid quotes, in elementary form as ticks or equidistantly aggregated to bars. Market data is either received and transferred directly from the market or recorded beforehand and then replayed afterwards. In the latter case we can differentiate between the less common real-time synchronous playback and the *as fast as possible*-procedure known from discrete event simulation, where the simulation clock advances to the time stamp of the next historical market data. In both cases the (missing) connection to the markets is transparent from the strategy's point of view.
If the strategy algorithm calls for action at the market, typically a buy or sell order is created. This order is transferred to an internal trade simulator, which decides on the basis of the market data whether, when and at which price orders of the strategy are executed in the simulation. The trade simulator notifies the strategy about order status changes and order executions. In addition it keeps a virtual brokerage account for the strategy, which is accessible for the strategy at any time.
The order executions are recorded in a *transaction history*, as a foundation for an extensive reporting. The *equity curve* describes the development of total equity over time. Along with the transaction history, it forms the basis for other performance indicators, allowing an estimation of the ratio of risk vs. return.

The operational mode delineated above is called *simulation mode*, because the results of strategy behavior are simulated with regard to the trading account; the strategy does not interact with the real market.
It should be stressed that we do not simulate the behavior of the market itself or the reaction of the market on the strategy decisions made. On the one hand, there are no reliable formal models for financial market behavior available; on the other hand, the impact of individual market participants is usually so small that one can abstract from their influence.

If it is decided that a strategy should trade at the market automatedly, the trade simulator is switched to *interaction mode*. In this mode it sends orders of the strategy to a broker or directly to exchanges. Conversely, the simulator immediately returns received order states and executions to the strategy. The simulator runs in bypass mode in terms of order processing, however, it keeps a record of the transaction history as well as a copy of the trading account data, enabling unchanged continuation of reporting.
From the strategy's point of view it is transparent whether it is trading in simulation or in interaction mode.

In the standard architecture a strategy always trades *against* a simulator as counterparty. Usually the simulator does not contribute anything to the strategy.

Changing Environments

The following citation of a practitioner illuminates one of the difficulties of automated algorithmic trading:

One cannot stick to the rules, because the rules do change every six months. (Ridpath 1997)

According to this, any trading strategy – as a rule-based behavior – is subject to the risk of obsolescence and consequently of limited applicability.
The approach of permanent self-adaptation to market development encounters the problem that reasonable behavior rules may not be identifiable for any environmental setting. Beyond that the non-applicability of current rules can only be recognized in hindsight, after losses already have been incurred. We can further argue that ongoing self-adaptation by continually developing new suitable behavior strategies may simply take too long, leaving no satisfactory period of use.

Multiple Strategies For Multiple Market Aspects

It can be observed that certain characteristic financial market phases or price movements are apparently reappearing, so that typical terms have been established for naming them. Relating to price direction, expressions such as *crash*, *sideways market* or *uptrend* are used. For coarse-granular price formations terms like *head-shoulder* or *triangle* exist, whereas fine-granular candlestick patterns bear names like *hammer* or *morning star*. Periodicity of phenomena has e.g. been phrased *presidential cycle*, moreover concepts like *January effect* or *window dressing* refer to yearly recurring developments. Besides Huang (2009) denominates a number of historical financial market anomalies each of which could be used in a profitable manner for several years.

Considering the only rudimentally listed multitude of potentially recurring phenomena and phases at financial markets, it cannot be excluded in general that trading strategies may be profitable at times, provided they are adjusted to their respective temporary trading environment.

The imprudent attempt to cope all market phases with the same single strategy is problematic: Given the multifaceted history of the target markets, the strategy had to incorporate (too) many degrees of freedom to consistently prove successful during back-testing. Practical experience shows that this approach ever leads to over-fitted systems failing after a short time, as soon as the market starts behaving different from the past.

Sticking to a single strategy, a reduction of the degrees of freedom promises more general robustness. Restricting the existing specializations in a largely uniform manner may indeed lead to a robust, but otherwise consistently mediocre and therefore unattractive strategy.

Alternatively, reducing the degrees of freedom selectively may allow to preserve certain specializations, while behavior patterns for other market phases may be lost completely. As a consequence the profit and loss phases of such a trimmed strategy will alternate in an unpredictable manner, hindering practical applicability. As an example we mention a remaining trend-following component of a strategy, causing false signals and subsequent losses in trendless markets.

The problems described brought institutional market participants such as investment banks or hedge funds to proceed to merge several specialized strategies into one single architecture. Accordingly, back-testers and automated trading systems have to be designed for an interplay of multiple strategies. This will be discussed in more detail in the next section.

STATE OF THE ART

Given multiple specialized strategies for different market aspects, the question arises on which specialists' advice to invest the capital. For this purpose a superordinated *selection strategy* could form an opinion on the general market situation and in further consequence choose a basic strategy that appears suitable (Chande 1997), see Fig. 2.

This intuitive two-layered approach is problematic in the sense that the selection strategy has to be correct in its opinion of a currently suitable basic strategy. While an intermediate failure of a chosen basic strategy may be realized by a drawdown of the equity curve, there are no context-specific confirmations which basic strategy to choose next, apart from the context-free rules of the selection strategy.

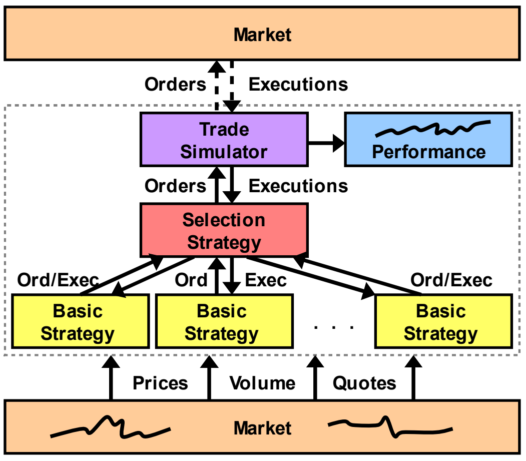

Figure 2: Multiple Basic Strategies with Superordinated Selection Strategy

Another weak point is that only the concatenated total sequence of the basic strategies can be analyzed, however, not the single basic strategies themselves: There is no equity curve attributable to the non-active strategies, therefore there are no individual evaluation possibilities. Finally, it is also unclear which part of the outcome results from the basic strategies and which from the selection strategy. Thus, reporting can only relate to the total complex in an indifferent manner.

In *portfolio trading* multiple specialized basic strategies trade in parallel on a shared trading account as shown in Fig. 3 (e.g. WealthLab 2007, Janeczko 2010).

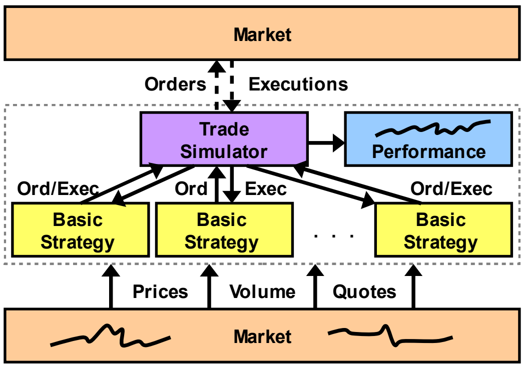

Figure 3: Multiple Basic Strategies with one Shared Trading Account

The shared account constitutes a synchronization point and results in undesired problems: If the total free cash is already exhausted by other strategies, no further trade signals can be realized. Thus, not every basic strategy can be applied in a guaranteed and independent manner. The reporting concerns the transactions of all basic strategies together. It evaluates the dynamic strategy mix on the highest level; however, the genuine contribution and quality of single strategies cannot be identified in a reliable manner due to the strategy blockage described above.

Institutional investors such as investment banks and hedge funds often trade a double-digit number of basic strategies on around hundred markets simultaneously, easily leading to thousand or more effective strategy/market combinations. Every basic strategy owns one separate trading account per market and is subject to a central superordinated *risk management* (Fig. 4).

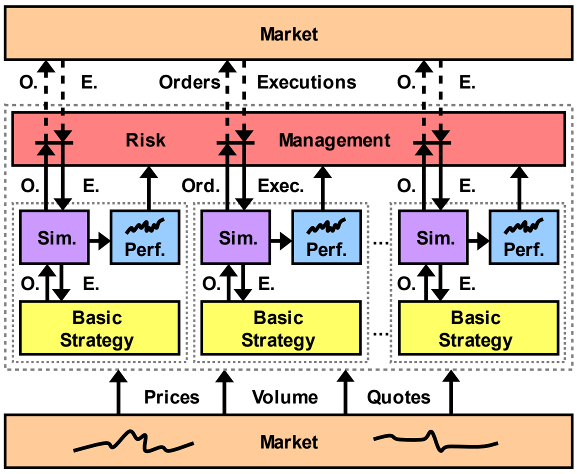

Figure 4: Multiple Basic Strategies with Separated Trading Accounts and Superordinated Risk Management

The risk management monitors each basic strategy in regard to performance by means of the individual reporting. It pauses strategies that are temporarily unsuccessful or reduces their position size, respectively.

At this, the risk management also considers relations between the basic strategies with reference to correlation of the equity curves, i.e. whether two strategies in combination increase total risk, are independent from each other or reduce risk when run in parallel.

As in the aforementioned selection strategy approach this is a two-layered architecture, but here each basic strategy can be rated and weighted individually without mutual interference. In addition, a global reporting can be generated at the aggregate level.

The risk management affects the total result in a significant manner. Since it takes a unique role in the architecture, it is a critical weakness. Two or more parallel risk management strategies are not feasible; their combination (*and*, *or*, …) may be too restrictive or too tolerant, resulting in mutual blockage or annulment of risk management decisions, respectively.
Thus, a multiple, superimposed risk management would be in danger of complete failure, whereas a single risk management is a risk factor itself.

A NEW APPROACH

In the following, a multi-layered software architecture for back-testing and automated trading is introduced, where each strategy is assigned a separate simulator (see Fig. 5). Instead of a monolithic risk management we provide a hierarchically organized strategy evaluation with subsequent imitation, combination or synthesis of analyzed strategy behavior.

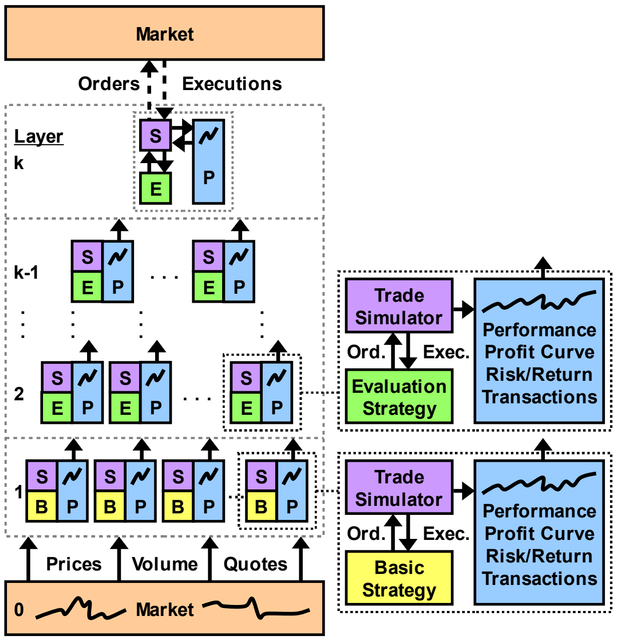

Figure 5: Introduced Software Architecture

Basic Strategies

Similar to the last-mentioned approach of the previous section, every trading strategy is assigned an own simulator, leading to individual and independent measurability. Our approach is novel in the sense that also in interaction mode of the overall system, each basic strategy only trades against its own local simulator in simulation mode.

Beyond that, each basic strategy is provided with the same initial starting capital again after each completed transaction. Instead of the equity curve (containing the starting capital and accumulating recent profits or losses as a percentage) evaluation is based on the absolute *profit curve*, adding only the net results of the transactions during trading.

In this way each basic strategy can add up an unlimited loss without going bankrupt. It is important to maintain the strategies in order to avoid thinning out the strategy pool during longer loss periods. Thus, rarely applicable strategies for only sporadic stock market phases remain within the strategy spectrum.

The profit curve of a strategy rises when it captures an aspect of the market successfully (e.g. a trend, a price pattern or an inefficiency) and implements it into profitable trading sequences. However, if it fails, the profit curve will decrease. Nevertheless, a strategy with high absolute loss should be considered in times of temporary loss reduction, because then it is apparently in a profitable phase. Thus not the absolute value of the profit curve is of interest, but merely its slope (i.e. the first derivation or its momentum, respectively).

The basic strategies are not restricted by a superior instance, such as selection strategies or risk management, but are trading at their own discretion against their own simulator anytime. Hence strategy quality is reflected in profit curves unaffectedly and under realistic conditions, without risking real losses.

As usual, the simulator records the profit curve, additional performance indicators and the transaction history for each strategy. The trade orders and executions contained in the transaction history can be regarded as the external behavior of a strategy – thus it is possible to relate a strategy's behavior to its profit curve. From this vantage point, the transaction history allows other instances to access and further use strategy behavior.

Evaluation Strategies

Instead of a single conventional risk management as in the above-mentioned approach, the presented software architecture provides a complete superordinated layer with several evaluation strategies.

The aim of the evaluation strategies is to recognize and prevent errors of strategies of the subordinated layer and to adopt and possibly further develop successful behavior on the contrary. Considering the basic failure possibilities of the subordinated strategies, the evaluation strategies are serving as risk filter to the benefit of increased robustness and error tolerance.

For this purpose the evaluation strategies of a layer rate the strategies of the subordinated layer in parallel and independently. Subsequently they carry out own trading decisions based upon the subordinated strategies and pass them on to their own local trade simulator.

According to their task and in contrast to basic strategies, evaluation strategies do not receive market data. Instead they analyze the profit curves and performance indicators of the strategies of the subordinated layer and assess their quality by means of individual, different valuation standards. Apart from total yield and statistical measures for the risk/return profile, the frequency, amount and recovery time of drawdowns and upswings can be computed and weighted on an individual basis.

The evaluation strategies also have access to the transaction histories of the strategies of the subordinated layer. Apart from quality, they are thereby able to investigate the behavior responsible for the observed results and to further use this behavior as orientation.

As a result, evaluation strategies derive their own behavior from the strategies of the subordinated layer in order to trade against their own local simulator. In contrast to the conventional software architectures above, the subordinated layer is not restricted at the same time.

Since the behavior of evaluation strategies is only oriented towards transaction histories of the strategies of the subordinated layer, we can speak of *imitation* in the easiest case of adopting behavior in a 1:1 manner, of *combination* if existing behavior is recombined and of *synthesis* for the case of generating new, so far unobserved behavior from the behavior of subordinated strategies.

Below some examples for behavior strategies on the evaluation level are outlined:

1. Imitation: Identify the currently most successful subordinated strategy and reproduce its transactions.
2. Negation: Trade contrarily to the transactions of a currently strongly loss-making subordinated strategy.
3. Basis for success: Do the currently most successful subordinated strategies have a common transaction intersection? If so, this intersection is possibly responsible for the success. Reproduce only the congruent transactions.
4. Diversification: Reproduce the transactions of the conjunction set of the currently most successful subordinated strategies.
5. Error prevention: Do the currently least successful subordinated strategies have a common transaction intersection? If so, this intersection may be responsible for the failure. Avoid these transactions.
6. Insignificance-Filter: Do transactions exist that are carried out by the currently most successful subordinated strategies as well as by the least successful subordinated strategies? Avoid such transactions, as they cannot be causative for the success or failure.
7. Pair Trading: Is there a subordinated strategy (not necessarily absolutely successful) which is almost always relatively more successful than another subordinated strategy? Reproduce the transactions of the first strategy and trade contrarily to the transactions of the second strategy.
8. Mutual confirmation and divergence: Is there a group of subordinated strategies whose group members traded identically in success periods but inconsistently in loss periods? Reproduce the transactions as soon as all group members trade identically and stop imitation of the group at the first instance of deviation of a group member from the common trading scheme.

More evaluation strategies have been elaborated that are not explicitly outlined in this paper. Generally, relations between the strategies of the subordinated layer are analyzed and own behavior is deduced from these relations, using strategy behavior of the subordinated layer.

Evaluation Strategy Layers

The simulators assigned to the evaluation strategies are technically identical with the simulators of the basic strategies. Therefore trading decisions of each evaluation strategy are documented by a profit curve, performance indicators and a transaction history as well and hence are rateable on their part.

Since it can hardly be taken for granted that evaluation strategies are impeccable, it is obvious to rate them likewise. Accordingly the proposed software architecture provides multiple evaluation layers built on each other, operating on the profit curves, performance indicators and transaction histories of their respective subordinated layers.
The strategies of each evaluation layer can be identical to those of the subordinated layer. Alternatively it is worth considering to implement adjusted evaluation and behavior strategies for each layer.

The number of strategies per evaluation layer need not necessarily be constant. It is also conceivable that the number of strategies decreases on each layer, resulting in a pyramidal evaluation hierarchy.

The k^{th} upper evaluation layer contains only one single strategy representing the total result of the strategy hierarchy. This strategy always trades against its own local simulator, providing profit curve, performance indicators and transaction history at any time.

If the software architecture is switched from simulation mode to interaction mode, the most superior simulator records solely the trading orders of the most superior evaluation strategy in its transaction history. It forwards only these orders to a broker or directly towards the exchanges. External order executions are recorded in the transaction history again and returned right away to the most superior strategy. The profit curve and performance indicators are updated analogously to simulation mode. In global interaction mode, apart from the most superior simulator all other simulators remain in local simulation mode.

Since the most superior evaluation strategy has direct or indirect access to the trading decisions of all lower layers, it can trade simultaneously and with diversified strategies on distinct markets. It is not at all limited to just one sub-strategy in one market.

The most superior evaluation strategy has – in contrast to all others – access on its own profit curve, performance indicators and transaction history, due to the absence of a higher instance to analyze them. Thus it can also carry out self-control in interaction mode and discontinue its own (i.e. the overall) online trading temporarily in case of undesirably strong drawdowns of the profit curve. In such phases the strategy could switch back to simulation mode on a temporary basis, avoiding real losses. Awaiting the end of its own unprofitable phase by observation of the profit curve updated in simulation mode, it could return to the market in interaction mode afterwards.

Of course the most superior evaluation strategy can also implement selection, diversification and control functions in the sense of classical risk management. In this respect conventional architectures as described in the preceding section *state of the art* merely represent limited special cases of the general hierarchical architecture introduced here.

OBJECTS OF INVESTIGATION

A number of issues arise from the software architecture described above. First of all we have to investigate whether our presented approach is able to gain additional value in the case of identical basic strategies, compared to conventional architectures. This would be the case if the evaluation strategies created additional benefit, e.g. in terms of an improved risk/return profile or a more reliable compliance of specified minimal or maximal performance indicators.

Beyond that it should be explored to what extent a general relation between the three success factors *quality of basic strategies*, *quality of evaluation strategies* and *number of evaluation layers* can be established. For example, is it possible to compensate for poorer evaluation strategies by an increased number of evaluation layers?

Furthermore it has to be clarified how many evaluation layers are adequate, subject to the market data frequency and the intended trade frequency. Each additional evaluation layer means an extra layer of indirection against market data and potentially complicates precise reactions. Since (bad) success of subordinated strategies can only be recognized after a certain time for sure, each additional evaluation layer introduces an extra latency. This can mean that changes in behavior of the basic strategies are detected by the most superior evaluation strategy with high delay, in such a way that chances are used behind time and risks are identified and limited too late.

Besides it has to be examined whether all evaluation strategies are equally suitable for each evaluation layer or whether some strategy types are more adequate for the lower or higher layers, respectively. In addition we have to check whether basic strategies can also be reused in the evaluation layers appropriately.

Cross-references to other research areas have to be further elaborated:

- The hierarchy of processing layers bears a structural resemblance to *Neural Networks* (NN), in particular with multilayer perceptrons (MLP, Rumelhart and McClelland 1986). While neurons are structured relatively simple and in a homogeneous manner, each strategy node of our architecture has an individual and independent algorithm and an assigned simulator at its disposal. Despite differences in detail, methods and insights from the field of Neural Networks and MLP may be applicable.
- The use of several, complementary basic and evaluation strategies accommodates the request to access an adequate behavior spectrum under as many differing environmental conditions as possible. Therefore an adaptation of strategies is only desirable if improved diversity and optimized coverage of the problem space can be achieved. For this purpose *Learning Classifier Systems*, especially eXtended Classifier Systems (XCS, Wilson 1995) could be integrated into the software architecture.
- For the beginning, *Learning* has deliberately been left out of consideration. With a fixed number of strategies, learning always means forgetting previous behavior in favor of newly acquired approaches. If market phases specialized on by certain strategies are absent for a longer time, a replacement of „old“ strategies and an overestimation of the more recent past (an “after the event”-learning) impends. Thus there is a risk that the behavior pool unwantedly focuses on the historical short term horizon and the capability for adequate reactions is lost, provided per se profitable market phases randomly recur only after a longer break. It has to be examined to what extend these concerns can be met by a suitable selection and parametrization of learning methods.

It is of particular interest whether a fixed minimal set of evaluation strategies and layers can be determined, utilizing the basic strategies independently of their specific form in an optimal manner in terms of a specified objective function.

STATUS OF WORK

The theoretical foundation of the introduced software architecture has been elaborated and generalized beyond the concrete strategy domain of financial markets.

On basis of the public domain simulation software *DESMO-J*, a special simulator for financial market trading strategies has been created (*FiSSMo*) allowing backtesting as well as automated trading of basic strategies (Golombek 2010). (DESMO-J is a simulation framework for discrete event simulation, which has been developed in the research group of Prof. B. Page at the Department of Informatics at the University of Hamburg, Germany (Page and Kreutzer 2005), see www.desmo-j.de.)

A superordinated software framework providing the described hierarchy of simulators both in general and particularly with regard to financial market trading strategies is currently under development (*MESSiE*).

At the same time a walk forward optimizer for DESMO-J is implemented, allowing dynamic re-optimization of model strategies in parallel to simulation experiments (Felgendreher 2011). For this the simulation framework DESMO-J was extended by storage, reset and resume of any simulation states during experimental run time (Janz 2010) as well as support of handling synchronization to wall clock time (Klückmann 2009).
The walk forward optimizer on one hand is intended to contrast the multi-layered approach with a „flat“ alternative by relinquishing the evaluation layers and introducing adaption by periodic or success dependent re-optimization merely of the basic strategies.
On the other hand it will be integrated into the superordinated software framework MESSiE, in order to assess the usefulness of dynamic adaptation in the strategy layers.

The basic strategy simulator FiSSMo works comparable to other modern back-testers as SmartQuant (2006) or RightEdge (2010). It produces the same results as the aforementioned commercial software, but has more expressive power and is more flexible in its architecture. Thus it can implement a wider range of basic strategies.
The superordinated software framework MESSiE is in an early stage of development. We have planned a number of experiments and will conduct, validate and evaluate them as soon as the software is available in a dependable manner. Results will be published in a subsequent paper.

SUMMARY AND OUTLOOK

In our paper we report on the development of a novel hierarchical software architecture where simulators establish an integral component. Basic strategies adjusted on special aspects of the application domain (here: financial market trading) are each assigned a separate simulator, determining success curves, performance indicators and action histories for their simulated behavior. These intermediate results are constantly updated and analyzed by superior evaluation strategies on several hierarchical layers. Each evaluation strategy imitates, combines or synthesizes the behavior patterns of the strategies of the subordinated layer against its own simulator. Finally, in interaction mode the most superior evaluation strategy is connected with the real environment by its simulator.

A number of interesting issues concerning the characteristics and the interplay of basic strategies, evaluation strategies and number of evaluation layers arise. In addition comparisons with conventional approaches without evaluation layers or without integral simulators have to be drawn. The relations to Neural Networks have to be considered, and the effects of introducing adaptation by Learning Classifier Systems, Walk-Forward-Optimization or Learning into the architecture have to be investigated.

In general the presented software architecture is domain-independent, even if proving is carried out in the field of financial market trading strategies initially. It could be of interest to investigate its applicability in other strategy domains such as navigation in road traffic or behavior in social networks. Due to comparably high numbers of independently and consciously acting individuals, similar phenomena could also exist in these fields. Therefore the application of the described software architecture could make some sense here. A comparative study of successful evaluation strategies in different domains could possibly lead to a generalized, domain-independent set of evaluation strategies and evaluation layers, capable of generically optimizing given application-specific basic strategies.

REFERENCES

Arthur, W.B.; J.H. Holland; B. LeBaron; R. Palmer and P. Tayler. 1997. "Asset pricing under endogenous expectations in an artificial stock market". In *The economy as an evolving complex system.* Addison-Wesley, Reading, Mass, 15–44.

Chande, T.S. 1997. *Beyond technical analysis. How to develop and implement a winning trading system.* Wiley, New York, 116–123.

Darley, V. and A.V. Outkin. 2007. *A NASDAQ market simulation. Insights on a major market from the science of complex adaptive systems.* World Scientific, Hackensack, NJ.

Felgendreher, K. 2011. *Konzeption und Realisierung eines Walk-Forward-Optimierers für das Simulationsframework Desmo-J mit exemplarischer Anwendung.* Diploma Thesis. Department of Informatics, University of Hamburg, Hamburg, Germany.

Golombek, O. 2010. *Entwurf und Implementation eines simulationsbasierten Frameworks zur Analyse von Finanzmarkt-Handelsstrategien.* Diploma Thesis. Department of Informatics, University of Hamburg, Hamburg, Germany.

Haldane, A.G. 2009. *Rethinking the financial network.* Speech delivered at the Financial Student Association, Amsterdam, April 2009.

Hamilton, W.P. 1922. *The stock market barometer.* Harper, New York, London.

Hommes, C.H. 2006. "Heterogeneous agent models in economics and finance". In *Handbook of Computational Economics. Volume 2. Agent-Based Computational Economics.* L. Tesfatsion and K.L. Judd (Eds.). Elsevier/North-Holland, Amsterdam, 1109–1186.

Huang, Z.J. 2009. *Real-Time Profitability of Published Anomalies: An Out-of-Sample Test.* Working Paper. Department of Finance, University of Wisconsin, Milwaukee. http://ssrn.com/abstract=1364813.

Jacobs, B.I.; K.N. Levy and H.M. Markowitz. 2004. "Financial market simulation. In the 21st century". *Journal of Portfolio Management,* 30th Anniversary Issue, 142–151.

Janeczko, T. 2010. *AmiBroker 5.30 User's Guide.*

Janz, T. 2010. *Prototypische Implementierung einer Verwaltung von Momentaufnahmen im Simulationsframework Desmo-J.* Bachelor Thesis. Department of Informatics, University of Hamburg, Hamburg, Germany.

Kearns, J.; W. Kisling and N. Mehta 2010. "Goldman Tops JP-Morgan as Best Broker as Speed Shakes Up Trading". http://www.businessweek.com/news/2010-01-29/goldman-tops-jpmorgan-as-best-broker-as-speed-shakes-up-trading.html.

Klückmann, F. 2009. *Realzeitsynchrone Simulation – Begriffe, Anwendungen und exemplarische Umsetzung anhand des Simulationsframework DESMO-J.* Diploma Thesis. Department of Informatics, University of Hamburg, Hamburg, Germany.

Kocur, M. 1999. *System-Konzeptionen. Systeme von Handelsstrategien an Futures-Märkten. Konzeption und Performance.* TM-Börsenverlag, Rosenheim.

LeBaron, B. 2006. "Agent-based computational finance". In *Handbook of Computational Economics. Volume 2. Agent-Based Computational Economics.* L. Tesfatsion and K.L. Judd (Eds.). Elsevier/North-Holland, Amsterdam, 1187–1233.

Levy, M.; H. Levy and S. Solomon. 2000. *The microscopic simulation of financial markets. From investor behavior to market phenomena.* Academic Press, San Diego.

Lux, T. and M. Marchesi. 2000. "Volatility clustering in financial markets. A microsimulation of interacting agents". *International Journal of Theoretical and Applied Finance* 3, No.4, 675–702.

Mauboussin, M.J. 2002. "Revisiting Market Efficiency: The Stock Market As A Complex Adaptive System". *Journal of Applied Corporate Finance* 14, No.4, 47–55.

Page, B. and W. Kreutzer. 2005. *The Java simulation handbook. Simulating discrete event systems with UML and Java.* Shaker, Aachen.

Rhea, R. 1932. *The dow theory. An explanation of its development and an aid in speculation.* Baron's The national financial weekly, New York.

Ridpath, M. 1997. *Trading reality.* Mandarin, London.

RightEdge 2010. *RightEdge Help.*

Rumelhart, D.E. and J.L. McClelland. 1986. *Parallel distributed processing 1. Explorations in the microstructure of cognition.* MIT Press, Cambridge, Ma.

Schaefer, E.G. 1960. *How I helped more than 10,000 investors to profit in stocks.* Prentice-Hall, Englewood Cliffs, N.J.

Sharp, J.A. and D.H.R. Price. 1984. "System dynamics and operational research: An appraisal". *European Journal of Operational Research* 16, No.1, 1–12.

SmartQuant 2006. *Introduction To The SmartQuant System Architecture.*

Wealth-Lab 2007. *Wealth-Lab Developer 4.0 User Guide.*

Wilson, S.W. 1995. "Classifier Fitness Based on Accuracy". *Evolutionary computation* 3, No.2, 149–175.

AUTHOR BIOGRAPHIES

BERND PAGE holds degrees in Applied Computer Science from the Technical University of Berlin, Germany, and from Stanford University, USA. As professor for Applied Computer Science at the University of Hamburg he researches and teaches in the field of Discrete Event Simulation as well as in Environmental Informatics.

ARNE KOORS obtained his diploma degree in Computer Science from the University of Hamburg, Germany, in 1999. Since then he has been working as a software developer and management consultant in the manufacturing industry, primarily in the field of forecasting and demand planning. Since 2007 he works on his PhD thesis in the simulation group led by Prof. Page.