Quantitative and Qualitative Support for Non-Functional Requirements in Stock Trading Systems

Jie Sun¹, Liping Zhao², Peri Loucopoulos³, Xinyu Wang¹, Shanping Li¹ and Xiaohu Yang¹

¹College of Computer Science, Zhejiang University, Hangzhou, P. R. China
{jibin, wangxinyu, shan, yangxh}@zju.edu.cn
²School of Computer Science, University of Manchester, Manchester, U.K.
{liping.zhao}@manchester.ac.uk
³Department of Informatics and Telematics, Harokopio University of Athens, Greece
{p.loucopoulos}@hua.gr

Abstract. The efficient functioning of financial markets is important for growth across the world, affecting businesses, society and individuals. Stock trading, a key activity within financial markets, is undergoing major transformation fuelled by advances in information systems techniques. Our motivation is in the development of stock trading systems that are robust and are transparent to regulators, traders and the public. Specifically, the work presented in this paper focuses on the way that quality factors may be considered at the requirements phase when designing systems to support stock trading processes. We focus on three quality factors namely performance, availability and security. We introduce a modeling paradigm that extends the well-known UML Activity model to explicitly represent these three quality factors and augment this through quantitative and qualitative reasoning. The approach is exemplified through the use of a small aspect of a stock trading process, dealing with the ‘order crossing’ part, taken from the practices of an international firm.

Keywords: business process modeling, non-functional requirements, quantitative reasoning, qualitative reasoning, stock trading processes

1 Introduction

There has been much discussion and debate recently about the contribution of technological advances in new products and services in support of financial trading. The Foresight report [1], the result of a very large project involving more than 150 leading academics from more than 20 countries, focused on the effect of Computer-Based Trading (CBT) systems, highlighting the controversy surrounding High Frequency Computer-based Trading (HFT) and Algorithmic Trading (AT). The report states “Markets are already 'socio-technical' systems, combining human and robot participants. Understanding and managing these systems to prevent undesirable behaviour in both humans and robots will be key to ensuring effective regulation”. At the same time new regulations and directives such as the European Commission’s MiFID II (Markets in Financial Instruments Directive) [2, 3] are in preparation and will seek in the future to regulate the way that firms trade using HFT and AT.
Against this backdrop, many stock trading firms are in the process of re-engineering their systems in preparation for a possibly major disruption to their usual modes of operation. It is therefore timely to consider quality aspects as an important facet of the development of such systems as much as their functionality. Traditionally, the development of stock trading systems considers Functional Requirements (FRs) and Non-Functional Requirements (NFRs) separately [4, 5]. At best FRs are analyzed formally using business process models, whereas NFRs are treated as simple textual descriptions [6, 7]. This separation of concerns makes it difficult for a business analyst to determine or reason about whether a certain stock trading process has satisfied the required NFRs [7, 8].

To overcome this shortcoming, this paper proposes an approach that (a) augments functional modeling of business processes with NFRs and (b) provides reasoning facilities to enable a business analyst to quantitatively and qualitatively evaluate business processes. We consider three desirable quality dimensions of stock trading systems namely, availability, performance and security. These three dimensions have long been considered as critical for the effective functioning of stock trading systems.

- **Performance**: In the globalization and competitive environment of the financial markets, performance directly affects the customer satisfaction and loyalty of stock trading companies [9]. Hence, it has long been recognized as a major drive in stock trading systems development [8, 10].
- **Availability**: Stock trading systems provide services to a large number of customers in different countries. Customers may require services at different times, so stock trading systems should be on service continuously for as long as possible. Should a system crash at runtime, its recovery time should be as little as possible. Availability is thus another indispensable NFR of stock trading systems [7].
- **Security**: Stock trading systems should still provide required services, even when attacked by malicious behaviors. The complex stock trading business involving financial transactions regulated by laws and directives, and interacting with other external trading systems makes security an essential and fundamental NFR of these systems [8].

In considering these three NFRs, the paper introduces and demonstrates a methodical approach that consists of three interrelated components:

1. Modeling of NFRs as an integral part of business process modeling.
2. Quantitative reasoning on the basis of three reasoning rules.
3. Qualitative reasoning on the basis of both Softgoal Interdependency Graph (SIG) [11] and Problem Interdependency Graph (PIG) [12, 13].

The systematic way of dealing with the modeling of and reasoning about three important NFRs, establishes an important foundation for the success and sustainability of stock trading systems, their development and evolution.

The paper is structured as follows: Section 2 gives a brief overview of stock trading processes in order to provide an understanding of some major activities found in these processes and also to introduce an example from a real situation, an example that is used throughout the paper to demonstrate the theoretical contributions. Section 3 focuses on the proposed approach in terms of the three aforementioned components of modeling, quantitative and qualitative reasoning. Section 4 discusses other ap-
2 Overview of Stock Trading Processes

Stock trading processes vary from one trading venue to another and are supported by a variety of systems, including order management systems, order routing systems, order crossing systems, trade printing systems, trade management systems, etc. As an example, consider the following business process supported by an order crossing system. The system is a financial information system developed for a US-based multinational financial service company to support stock trading in USA, Europe and Australia. The actual order crossing process is highly complex with a large number of activities and various process flows, but for illustration convenience, we have abstracted and simplified it into the UML activity diagram in Fig. 1.

![Activity Diagram of the Order Crossing Process](image)

**Fig. 1.** The activity diagram of the order crossing process

A Trader submits his/her trading data in the form of a buy-side or sell-side order to an Order Crossing System (OCS). A buy-side order is an order to buy some shares of a certain stock symbol, whereas a sell-side order is an order to sell some shares of a certain stock symbol. A stock symbol is a unique identifier for a group of publicly traded shares of a specific company, such as IBM, AAPL, INTC. The OCS checks the order data and customer details, and notifies Trader with confirmation or rejection. If the order were valid, the OCS would save it. Then, from the Market Data Feeder
(MDF), the OCS obtains the market price information relevant to the stock symbol of the order, and also determines a price for the order. The OCS matches the buy-side orders with sell-side orders according to their stock symbol. For example, the IBM buy-side orders can only be matched with the IBM sell-side orders, rather than with the AAPL orders or the INTC orders. Once matched, the matched order is executed into a trade. Next, the OCS updates both order information and account information, and sends the trade information to Printing Agent (PA) for verification according to financial regulations. If the trade were illegal, it would be corrected and reprinted. After printing, the trade would be allocated to sub-accounts, because several orders of all the sub-accounts are always aggregated into one order, and crossed under some parent account. Finally, OCS would notify Trader with the trade information.

For the business process represented by the activity diagram of Figure 1 and described above, there was an information system that was developed on behalf of an international financial service. The development of this system was based on a set of requirements specification with details on FRs and NFRs. The FRs were first analyzed in detail using this business process model. The resultant system supported five essential functional services: Order Entry, Order Matching, Order Pricing, Trade Printing and Trade Management. Whilst the FRs were considered at the requirements analysis phase, NFRs were considered as system-level requirements, and thus were not analyzed in depth until at the architecture design stage [7]. At the requirements analysis stage, NFRs were only listed with very simple descriptions. For instance, the system should ideally maintain a high availability equivalent to 99.99% of its operation time [7]. To the best of our knowledge this is in general the case for systems being developed to support stock trading. Our aim therefore has been to consider NFRs at an early stage alongside FRs.

3 Support of NFRs in Modeling Stock Trading Processes

We support NFRs in stock trading processes in terms of both representation and evaluation, as shown schematically in Fig. 2. In the first step, we identify appropriate measures and descriptions to specify NFRs, and also integrate functional activity diagrams and NFRs into NFR-oriented activity diagrams. Then by exploring both FRs and NFRs knowledge in the NFR-oriented activity diagrams, we perform quantitative and qualitative reasoning to evaluate the satisfaction of NFRs. Each of the three stock trading NFRs can be evaluated both quantitatively and qualitatively, but due to the space limitation, this paper demonstrates quantitative reasoning to evaluate performance and availability, and qualitative reasoning to evaluate security and its relationships with the other two NFRs. The reasoning results provide explicit, reasonable and understandable evidences to evaluate the NFRs of stock trading processes.

![Fig. 2. The support of NFRs in business processes](image-url)
3.1 NFRs-oriented Extension to UML Activity Diagrams

While UML activity diagrams can well support stock trading FRs based on a set of notations [14], they do not provide an appropriate representation for NFRs. In order to incorporate stock trading NFRs into activity diagrams, we have defined a UML profile. The profile mechanism is offered to support a lightweight extension to UML metamodels [14]: Metamodel is a model that defines the language for expressing a model. Profile is a specific package, which consists of stereotypes and tagged values defined to extend metamodels for different purposes. Stereotype is a new type of modeling element that extends the semantics of metaclasses in metamodels. The use of a stereotype must be based on a certain metaclass it extends. The notation of a stereotype can adapt from a pre-existing metaclass by attaching icons. A stereotype can also have attributes, which are referred to as tagged values.

Our NFRs-oriented profile that extends the metamodel of UML activity diagrams is shown in Fig. 3. In order to make our NFR-oriented activity diagrams easy to understand, it is better to specify the metamodel as simple as possible. Therefore, in our profile, we define two new stereotypes, extended from UML metaclasses:

- The NFRProcess (NP).
- The NFRActivityNode (NAN).

A NP is an activity diagram extended with NFRs. NP extends Activity, and therefore derives all its relationships with other concepts. A NP consists of ordered activity nodes that are incorporated with NFRs – that is NFRActivityNodes, and other modeling elements of UML activity diagrams, e.g. ControlNode, ActivityEdge and ActivityPartition.

A NAN is an activity node extended with three NFRs (performance, availability and security). NFRs are defined as the tagged values of NAN – that is, each NAN can be annotated with these NFRs. In Fig. 3, NAN extends two metaclasses – Action and StructuredActivityNode. For an action whose NFRs can be estimated, we directly annotate it with NFRs; whereas some actions’ NFRs are not able to be estimated individually. For instance, the two actions in Fig. 1 – submit order and receive order – should be executed together as an atomic business transaction. We cannot stop them halfway to estimate their individual NFRs; otherwise, they are not able to achieve consistent results. For such actions, we can group them into a StructuredActivityNode and annotate it with NFRs. In the UML metamodel, a StructuredActivityNode is aggregated from ActivityNodes. Therefore, deriving this characteristic, a NAN that extends StructuredActivityNode can also be aggregated from NANs that extend Action – that is, NANs can have nested NANs. This relationship is represented as the self-aggregation of NAN in Fig. 3.

![Fig. 3. The NFRs-oriented profile extended from the metamodel of UML activity diagrams](image-url)
We take the order crossing process (see Section 2) as an example to illustrate our NFR-oriented activity diagrams. Based on system logs and other development documents, we estimated the concrete measurements for performance, availability and security. Then we incorporate them into the original order crossing process (Fig. 1),
as shown in Fig. 4. Due to space limitation, for nested NANs, their NFRs with null values are not explicitly shown.

Each NAN is annotated with the three NFRs of performance, availability and security with the letters ‘P’, ‘A’ and ‘S’ respectively.

Performance – measured by two frequently-used metrics – throughput and response time. The former is usually used to measure the overall performance of a stock trading process. It is hard to estimate the throughput for specific activity node(s). The response time of activity node(s) is much easier to be obtained from domain experts based on their experiences and system history data.

Availability – usually measured by a simple uptime rate of the overall stock trading process. But for specific activity node(s), this metric is unmeasurable. The domain experts advised us to use the tolerant down time to measure the availability of activity node(s), which is also frequently used and much easier to be estimated.

Security – qualitatively specified by security goals to be achieved or security problems to be avoided.

In the modeling of NFRs, our NFR-oriented activity diagrams bridge the gap between NFRs and FRs for stock trading processes, by explicitly representing NFRs in the functional activity nodes. The incorporation of NFRs with individual activity nodes is also helpful to trace how specific activities affect NFRs of the whole process. Based on the traceability between NFRs and FRs, business analysts may easily find specific activities to be improved and thus save effort for process redesign.

3.2 Quantitative Reasoning for Stock Trading NFRs

Quantitative reasoning is performed in order to evaluate the performance and availability of a stock trading process as a whole. They are quantified by aggregating the NFRs of all NANs in the NFRProcess.

3.2.1 Rules for Measuring NFRs

As business process models may involve different process flows according to their FRs, we define three reasoning rules for the five basic flow patterns: Sequence, Alternative, Parallel, Join, and Merge [15].

Rule 1: sequence process flow. Fig. 5 shows an example of this type of process flow. In this case, performance/availability is reasoned by adding together the performance/availability values of all NANs in a sequence process flow.

\begin{align*}
P_{seq} &= \sum_{i=0}^{n} P_{NAN_i} \\
A_{seq} &= \sum_{i=0}^{n} A_{NAN_i}
\end{align*}

where, \( n \) represents the total number (usually \( n \geq 0 \)) of Nan in a sequence flow and \( i \) points to the \( i \)th Nan. If there is no Nan in a certain sequence process flow – that is, its \( n \) is equal to zero, then its performance and availability are equal to zero.
**Rule 2: alternative/merge process flow.** Fig. 6 shows an example of this type of process flow. In this case performance/availability is reasoned by adding together the performance/availability value of each branch according to the occurrence probability of the branch. Formulae (3) and (4) are, respectively, used to calculate performance and availability.

\[
P_{alt} = \sum_{i=1}^{m} q_i \cdot P_{seq} \tag{3}
\]

\[
A_{alt} = \sum_{i=1}^{m} q_i \cdot A_{seq} \tag{4}
\]

where \( m \) represents the total number (usually \( m \geq 2 \)) of branches in an alternative/merge process flow; \( i \) points to the \( i \)th branch; \( q_i \) represents the occurrence probability of the \( i \)th branch (usually \( \sum_{i=1}^{m} q_i = 1 \)); \( n_i \) represents the number (usually \( n_i \geq 0 \) ) of sequential NANs in the \( i \)th branch.

In some alternative process flows, not all branches that split from one alternative node would be finally brought together into one merge node. This means that (a) one or more branches may be ended with final node(s) or (b) one or more branches may go back to previous NAN(s) and thus indicate a loop (Fig. 7). In the first situation, formulae (3) and (4) can be directly applied to calculate performance and availability, respectively; whereas in the second situation, we should first transform the loop flow into another form, as shown in Fig. 8. Then for the alternative/merge flow in this figure, we can use Formulae (3) and (4) to calculate its performance and availability, respectively. In Fig. 8, \( l \) represents the iteration times (usually \( l \geq 0 \)) of the loop.

**Rule 3: parallel/merge process flow.** Fig. 9 shows an example of this type of process flow. In this case, performance/availability is determined by the maximum performance/availability value of all branches. Formulae (5) and (6) are, respectively, used to calculate performance and availability.

\[
P_{par} = \max_{i=1}^{m} P_{seq} \tag{5}
\]

\[
A_{par} = \max_{i=1}^{m} A_{seq} \tag{6}
\]
Qualitative Reasoning for Stock Trading NFRs

3.2.2 A Method for Measuring NFRs

Based on the reasoning rules defined in section 3.2.1, performance and availability of a business process are quantified in the following three steps:

**Step 1: Business process simplification.** A NFRProcess is first simplified into a business process, consisting only of ordered NANs that are annotated with concrete performance and availability values. We still take the order crossing process as an example to illustrate the quantitative reasoning. In Fig. 10, we created a business process by simplifying the NFRProcess in Fig. 4. In the simplified process, each alternative branch is attached with a value of its occurrence probability. The probability can be estimated based on the statistics of system history data. Due to space imitation, Fig. 10 does not repeatedly show the performance and availability values of NANs.

**Step 2: Business process partitioning.** The simplified business process is partitioned into individual sub-processes, according to the five basic process flow patterns. In this way, the reasoning rules can be directly applied inside each sub-process to calculate its performance and availability. In Fig. 10, the simplified order crossing process is partitioned into a set of basic processes flows with solid lines.

**Step 3: NFRs calculation.** By repeatedly applying the reasoning rules, we can calculate both performance and availability for each sub-process, and then add them together to obtain both performance and availability value of the overall business process. In Fig. 10, for each sub-process, we list the rule(s) that is/are used and also describe how performance (P) and availability (A) are calculated, respectively. Finally, the performance (P_{oc}) and availability (A_{oc}) calculation for the overall order crossing (OC) process are listed in the outermost box.

3.3 Qualitative Reasoning for Stock Trading NFRs

The qualitative reasoning is intended to (a) evaluate the security achievements of the overall stock trading process and (b) analyze the dependencies between security and the other two stock trading NFRs. NFRs are generally treated as softgoals to be achieved, and modeled by Softgoal Interdependency Graph (SIG) [11], but in addition to the goal-oriented knowledge, there are other knowledge components that cannot be described by clear-cut goals, such as problems/obstacles to be avoided [12]. To model such knowledge, Problem Interdependency Graph (PIG) [12] was proposed as a complementary approach. The combination of SIG and PIG can comprehensively describe and analyze the knowledge of NFRs [16]. Therefore, based on both SIG and PIG, we conduct the qualitative reasoning in four steps (Fig. 11). A SIG/PIG models non-functional goals to be satisfied/non-functional problems to be avoided, and their solutions (i.e operationalizing goals) through And/Or-decomposition and make(++)/help(+) /hurt(-)/break(--) contribution. How well solutions satisfy goals is
indicated by make(++)/help(+) contribution, whereas how effectively solutions address problems is attributed with hurt(-)/break(--) contribution. A solution may also have some positive (make/+ or help/) or negative (hurt/ or break/) side-effect toward other goals/problems.

![Fig. 10. Quantitative reasoning for performance and availability of the order crossing process](image)

![Fig. 11. Steps of NFRs-oriented qualitative reasoning (Step 4 is implicitly shown by labels)](image)

**Step 1: Problem and Goal Organization.** For a NFRProcess (NP), all of its security goals (SG) and security problems (SP) attached to NANs are, respectively, collected and organized into a SIG and a PIG. Individual goals/problems are connected through And/Or-decomposition to a root goal/problem. The root goal/problem is the security goal (SG[NP])/security problem (SP[NP]) of the overall NP. For example, if a certain NP has five problems (P[NAN]) and five goals (G[NAN]) attached to NANs, in this step, they are respectively organized into a SIG and a PIG, as shown in Fig. 11. Prob-
lems are connected using Or-decomposition to \(SP[NP]\), whereas goals are connected using And-decomposition to \(SG[NP]\).

**Step 2: Solution Identification.** To address/satisfy each \(P[NAN]/G[NAN]\), solution(s) are explored with the help of problem/goal refinement. The problems/goals may be And/Or-decomposed repeatedly until solutions are identifiable. For example, in Fig. 11, \(G1[NAN1]\) is And-decomposed into \(G11\) and \(G12\). They are sufficiently satisfied \((\text{make}++)\) by operationalizing goals \((\text{Op}) – G11\) and \(G12\), respectively.

**Step 3: NFRs Dependency Analysis.** The dependencies between NFRs are analyzed by exploring how each security solution influences performance and availability of the same NP. Solutions are related to performance \((\text{Performance}[NP])\) and availability \((\text{Availability}[NP])\) through side-effect. In Fig. 11, both Op51 and Op11 have a negative \((\text{hurt}/-)\) side-effect on \(\text{Performance}[NP]\), whereas Op51 has a positive \((\text{help}/+)\) side-effect on \(\text{Availability}[NP]\).

**Step 4: Problem Resolution and Goal Satisfaction.** How well \(SG[NP]\) and \(SP[NP]\) are addressed and satisfied by solutions are evaluated through label evaluation \([11, 12]\). During the evaluation, we first label each solution, and then, based on label propagation rules for different links \([12, 13]\), evaluate the label to determine how it impacts the resolution/achievement of its parent problem/goal. The label evaluation continues upward the SIG/PIG until \(SG[NP]/SP[NP]\) is evaluated. In Fig. 11, each solution is first labeled with \(\text{satisfied}\) (denoted by a check mark). A \(\text{satisfied}\) label on a solution is reversed to a \(\text{denied}\) label (denoted by a cross mark) on its parent problem/goal over a \(\text{break}(-)\) contribution; whereas over a \(\text{make}(++)\) contribution, its parent problem/goal is instead labeled as \(\text{satisfied}\). Hence, \(P51\) and \(P52\) are labeled as \(\text{denied}\), whereas \(G11\) and \(G12\) are labeled as \(\text{satisfied}\). Over an And-decomposition, a problem is denied when any of its sub-problems is denied, whereas a goal is satisfied when all of its sub-goals are satisfied. Thus, \(P5[NAN5]\) is labeled as \(\text{denied}\), whereas \(G1[NAN1]\) is labeled as \(\text{satisfied}\). Neither \(SG[NP]\) nor \(SP[NP]\) is labeled because one of the sub-problems/goals is not labeled. A \(\text{satisfied}\) label on a solution leads to a \(\text{weaklyDenied}\) label \((\text{w})\) on its side-effect goal over a \(\text{hurt}(-)\) side-effect; whereas over a \(\text{help}(+)\) side-effect, its side-effect goal is instead labeled as \(\text{weaklySatisfied}(\text{w})\). So \(\text{Performance}[NP]\) is labeled as \(\text{weaklyDenied}\), whereas \(\text{Availability}[NP]\) is labeled as \(\text{weaklySatisfied}\).

We have applied the above qualitative reasoning steps to the order crossing process. Fig. 12 shows the evaluation result of security and its dependencies with performance and availability. In the figure, the root problem/goal is the security problem/goal of the overall order crossing (OC) process, represented as \(SP[OC]/SG[OC]\). Through the And/Or decomposition, \(SP[OC]\) and \(SG[OC]\) are, respectively, connected to all sub-problems (e.g. malicious code spread [OE]) and sub-goals (e.g. identity legality [OE]) attached to NANs in Fig. 4 (Step 1). Note that NANs are denoted by their initials. The sub-problems and sub-goals are then connected to identified solutions (e.g. encryption [transmitted data]) through various \(\text{contribution}\) links (Step 2). Influences of the solutions on performance and availability are also represented through different \(\text{side-effect}\) links (Step 3). \(\text{Performance}[OC]\) may be hurt by some solutions; whereas \(\text{Availability}[OC]\) can be enhanced by some solutions. After the PIG and SIG are created, the resolution of problems and the satisfaction of goals are evaluated bottom-up from solutions (Step 4). The evaluation results show that \(SP[OC]\) and \(SG[OC]\) can be, respectively, addressed and satisfied by solutions.
4 Related Work and Discussion

To date, a number of approaches have been proposed to consider non-functional requirements with particular emphasis on business processes. Different quality metrics (e.g., size, coupling, cohesion, compactness, and structuredness) were identified to measure the modularity, complexity, and understandability of business processes [17-19]. These metrics were also integrated with different business process modeling languages [20, 21] and used to help the business process redesign [22-24]. All these works concentrate on the evolution qualities [25], such as maintainability, modifiability and extensibility, whereas our approach focuses on the execution qualities (i.e., performance, availability and security). Although we do not explicitly discuss in this paper evolution qualities, our representation of NFRs in business processes can also improve evolution qualities, such as maintainability and extensibility.

For execution qualities [25], Donzelli and Bresciani [26] proposed an approach in which quality modeling extends the utility of goal-oriented, agent-based RE techniques, enabling the resolution of organizational and soft systems issues. Al-Balushi et al [27] focused on the identification of conflicts in quality definitions using an analysis tool supported by domain ontology. They separately considered execution qualities, whereas we put more emphasis on supporting them in business processes. Various attempts have been made to describe NFRs in different business process models, such as Event-driven Process Chain [28, 29], Business Process Modeling Notation [28], and UML-driven Process Chain [28]. Specifically, Heravizadeh et al [30] paid attention to the capturing of quality dimensions of business processes. Heinrich [32] et al collected quality characteristics and attributes of processes, and BPM languages are enhanced in order to express these attributes. They adopt the ISO/IEC9126 standard (quality of software) for evaluating business processes. Pavlovski [33] proposes new artifacts to model the constraints associated with completion time, security privilege, availability of a business process and regularity or organization constraints. However, these approaches provide little support for the evaluation of NFRs. Our approach not only incorporates NFRs into business processes, but also evaluates them quantitatively and qualitatively.

Fig. 12. Qualitative reasoning for the order crossing process
There are also extensive works on evaluation for NFRs of business processes. For example, Jansen-Vullers et al [34] developed an approach to quantify business processes from different performance dimensions with various measures. Pourshahid et al [35] and Kavakli et al [40] introduced a framework for measuring and aligning processes with goals. Compared with our approach, their focus is on business process as a whole rather than its constituent parts. Saeedi et al [36] put forward an approach for incorporating a set of quality requirements into BPMN, and evaluated them quantitatively. Their quality metrics are specific for service-oriented processes, and the security requirement is absent from their evaluation. Damasceno et al [37] analyzed security for business processes, but their evaluation is performed after implementation. Aburub et al [38] presented an approach to the identification and inclusion of NFRs in business processes. They put more emphasis on qualitative analysis for NFRs, whereas our approach provides a more complete analysis both quantitatively and qualitatively. Heinrich et al [39] considered concepts such as activities, actors, information and physical objects as well as required resources for evaluation. In our approach, only activities and process flows are analyzed. Hence, a possible future work for us is to support NFRs with more process concepts.

5 Conclusion

This paper has made a significant contribution to the field of Requirements Engineering by showing in detail how to model and reason about NFRs in one important aspect of the financial sector namely that of stock trading. Specifically, the paper incorporates three key stock trading NFRs – performance, availability and security – into functional stock trading processes in an elegant and rigorous way, which provides an explicit traceability between NFRs and FRs. The paper also provides a quantitative and qualitative reasoning approach that step-by-step evaluates the satisfaction of individual NFRs, and analyzes their relationships for stock trading processes. Stock trading systems include most of the important characteristics of financial systems, such as the importance of customer satisfaction, and the involvement of national and international financial regulatory bodies. Therefore, the three NFRs discussed in this paper are as well crucial for other financial systems (e.g. banking systems and other financial trading systems). We believe that our approach can be applicable to other financial systems and in the future we plan to critically examine this claim by applying our approach to a banking application. Furthermore, we also believe that the conceptual framework of our approach is not confined to just the UML activity diagrams and in our future work we plan to demonstrate the approach’s utility using other well known methods such as BPMN. We also plan to extend the idea of combining qualitative and quantitative reasoning for a whole host of other NFRs and through this to develop a generic and at the same time pragmatic method. Our vision is for NFRs to be considered as an integral part of Requirements Engineering. To this end we plan to extend the quantitative reasoning to deal with more complex flow patterns, and to develop support tools in order to make our NFRs Requirements Engineering method widely and practically usable.
6  Acknowledgements

This work was jointly supported by State Street Corporation in the USA, and by the National Natural Science Foundation of China under Grant No. 61103032.

7  Reference