The TSE Trading Rules[1]

Dines Bjørner
Fredsvej 11, DK-2840 Holte, Danmark
E–Mail: bjorner@gmail.com, URL: www.imm.dtu.dk/~db

February 22, 2010

[1]This is version 2 of a document first released on January 28, 2010. The current version 2 corrects typos and mistakes of version 1. I thank Prof. Tetsuo Tamai, Tokyo University for commenting on version 1: clarifying issues and identifying mistakes and typos. Change bars designate changes wrt. version 1.

The reason why these notes are written is the appearance of [1].
I have taken the liberty of including that paper in this document, cf. Appendix A.
I had the good fortune of visiting Prof. Tetsuo Tamai, Tokyo Univ., 8–Dec.8, 2009.
I read [1] late November.
I then had wished that Tetsuo had given it to me upon my arrival.
I was, obviously ignorant of its publication some five months earlier.
I have now reread [1] (late January 2010).

I mentioned to Tetsuo that I would try my hand on a formalisation.
A description, both by a narrative, and by related formulas.

What you see here, in Chap. 1, is a first attempt[1].

[1]Earlier versions of this document will have Chap. 1 being very incomplete.

Contents

Chapter 1

The Tokyo Stock Exchange

This chapter was begun on January 24. It is being released, first time, January 28.

1.1 Introduction

This chapter shall try describe: narrate and formalise some facets of the (now "old"[1]) stock trading system of the TSE: Tokyo Stock Exchange (especially the 'matching' aspects).

1.2 The Problem

The reason that I try tackle a description (albeit of the "old" system) is that Prof. Tetsuo Tamai published a delightful paper [1, IEEE Computer Journal, June 2009 (vol. 42 no. 6) pp. 58-65)], Social Impact of Information Systems, in which a rather sad story is unfolded: a human error key input: an offer for selling stocks, although "ridiculous" in its input data (*"sell 610 thousand stocks, each at one (1) Japanese Yen"*, whereas one stock at 610,000 JPY was meant), and although several immediate — within seconds — attempts to cancel this "order", could not be cancelled ! This lead to a loss for the selling broker at around 42 Billion Yen, at today's exchange rate, 26 Jan. 2010, 469 million US $s ![2] Prof. Tetsuo Tamai's paper gives a, to me, chilling account of what I judge as an extremely sloppy and irresponsible design process by TSE and Fujitsu. It also leaves, I think, a strong impression of arrogance on the part of TSE. This arrogance, I claim, is still there in the documents listed in Footnote 1.

So the problem is a threefold one of

[1] We write "old" since, as of January 4, 2010, that 'old' stock trading system has been replaced by the so-called arrowhead system. We refer to the following documents:

- `http://www.tse.or.jp/english/rules/equities/arrowhead/pamphlet.html`
- `http://www.tse.or.jp/english/rules/equities/arrowhead/pamphlet-e.pdf`
- `http://www.tse.or.jp/english/rules/equities/arrowhead/pamphlet1e.pdf`
- `http://www.tse.or.jp/english/rules/equities/arrowhead/pamphlet2e.html`

[2] So far three years of law court case hearing etc., has, on Dec. 4, 2009, resulted in complainant being awarded 10.7 billion Yen in damages. See `http://www.ft.com/cms/s/0/e9d89050-e0d7-11de-9f58--00144feab49a.html`.

- **Proper Requirements:** How does one (in this case a stock exchange) prescribe (to the software developer) what is required by an appropriate hardware/software system for, as in this case, stock handling: acceptance of buy bids and sell offers, the possible withdrawal (or cancellation) of such submitted offers, and their matching (i.e., the actual trade whereby buy bids are marched in an appropriate, clear and transparent manner).
- **Correctness of Implementation:** How does one make sure that the software/hardware system meets customers' expectations.
- **Proper Explanation to Lay Users:** How does one explain, to the individual and institutional customers of the stock exchange, those offering stocks for sale of bids for buying stocks – how does one explain – in a clear and transparent manner the applicable rules governing stock handling.[3]

I shall only try contribute, in this document, to a solution to the first of these sub-problems.

1.3 A Domain Description

1.3.1 Market and Limit Offers and Bids

1. A market sell offer or buy bid specifies
   (a) the unique identification of the stock,
   (b) the number of stocks to be sold or bought, and
   (c) the unique name of the seller.
2. A limit sell offer or buy bid specifies the same information as a market sell offer or buy bid (i.e., Items 1a–1c), and
   (d) the price at which the identified stock is to be sold or bought.
3. A trade order is either a (mkMkt marked) market order or (mkLim marked) a limit order.
4. A trading command is either a sell order or a buy bid.
5. The sell orders are made unique by the mkSell "make" function.
6. The buy orders are made unique by the mkBuy "make" function.

type
√ 1 Market = Stock_id × Number_of_Stocks × Name_of_Customer
1a Stock_id
√ 1b Number_of_Stocks = {|n•n:**Nat**∧n≥1|}
1c Name_of_Customer
2 Limit = Market × Price
√ 2d Price = {|n•n:**Nat**∧n≥1|}

[3]The rules as explained in the Footnote 1 on the previous page listed documents are far from clear and transparent: they are full of references to fast computers, overlapping processing, etc., etc.: matters with which these buying and selling customers should not be concerned — so, at least, thinks this author !

3 Trade == mkMkt(m:Market) | mkLim(l:Limit)
4 Trading_Command = Sell_Order | Buy_Bid
5 Sell_Order == mkSell(t:Trade)
6 Buy_Bid == mkBuy(t:Trade)

1.3.2 Order Books

7. We introduce a concept of linear, discrete time.
8. For each stock the stock exchange keeps an order book.
9. An order book for stock $s_{id} : SI$ keeps track of limit buy bids and limit sell offers (for the *id*entified *s*tock, s_{id}), as well as the market buy bids and sell offers; that is, for each price
   (d) the number of stocks, designated by unique order number, offered for sale at that price, that is, limit sell orders, and
   (e) the number of stocks, by unique order number, bid for buying at that price, that is, limit buy bid orders;
   (f) if an offer is a market sell offer, then the number of stocks to be sold is recorded, and if an offer is a market buy bid (also an offer), then the number of stocks to be bought is recorded,
10. Over time the stock exchange displays a series of full order books.
11. A trade unit is a pair of a unique order number and an amount (a number larger than 0) of stocks.
12. An amount designates a number of one or more stocks.

type
√ 7 T, On [Time, Order number]
8 All_Stocks_Order_Book = Stock_Id $\overrightarrow{m}$ Stock_Order_Book
9 Stock_Order_Book = (Price $\overrightarrow{m}$ Orders) × Market_Offers
9 Orders:: so:Sell_Orders × bo:Buy_Bids
9d Sell_Orders = On $\overrightarrow{m}$ Amount
9e Buy_Bids = On $\overrightarrow{m}$ Amount
9f Market_Offers :: mkSell(n:**Nat**) × mkBuy(n:**Nat**)
10 TSE = T $\overrightarrow{m}$ All_Stocks_Order_Book
11 TU = On × Amount
12 Amount = {|n•**Nat**∧n≥1|}

1.3.3 Aggregate Offers

13. We introduce the concepts of aggregate sell and buy orders for a given stock at a given price (and at a given time).
14. The aggregate sell orders for a given stock at a given price is

(g) the stocks being market sell offered and

(h) the number of stocks being limit offered for sale at that price or lower

15. The aggregate buy bids for a given stock at a given price is

(i) including the stocks being market bid offered and

(j) the number of stocks being limit bid for buying at that price or higher

```
value
  14 aggr_sell: All_Stocks_Order_Book × Stock_Id × Price → Nat
  14 aggr_sell(asob,sid,p) ≡
  14   let ((sos,_),(mkSell(ns),_)) = asob(sid) in
  14g    ns +
  14h    all_sell_summation(sos,p) end
  15 aggr_buy: All_Stocks_Order_Book × Stock_Id × Price → Nat
  15 aggr_buy(asob,sid,p) ≡
  15   let ((_,bbs),(_,mkBuy(nb))) = asob(sid) in
  15i    nb +
  15j    nb + all_buy_summation(bbs,p) end

  all_sell_summation: Sell_Orders × Price → Nat
  all_sell_summation(sos,p) ≡
    let ps = {p′|p′:Prices • p′ ∈ dom sos ∧ p′ ≥ p} in accumulate(sos,ps)(0) end

  all_buy_summation: Buy_Bids × Price → Nat
  all_buy_summation(bbs,p) ≡
    let ps = {p′|p′:Prices • p′ ∈ dom bos ∧ p′ ≤ p} in accumulate(bbs,ps)(0) end
```

The auxiliary accumulate function is shared between the all_sell_summation and the all_buy_summation functions. It sums the amounts of limit stocks in the price range of the accumulate function argument ps. The auxiliary sum function sums the amounts of limit stocks — "pealing off" the their unique order numbers.

```
value
  accumulate: (Price m→ Orders) × Price-set → Nat → Nat
  accumulate(pos,ps)(n) ≡
    case ps of {} → n, {p}∪ ps′ → accumulate(pos,ps′)(n+sum(pos(p)){dom pos(p)}) end

  sum: (Sell_Orders|Buy_Bids) → On-set → Nat
  sum(ords)(ns) ≡
    case ns of {} → 0, {n}∪ ns′ → ords(n)+sum(ords)(ns′) end
```

To handle the sub_limit_sells and sub_limit_buys indicated by Item 17c on the facing page of the Itayose "algorithm" we need the corresponding sub_sell_summation and sub_buy_summation functions:

```
value
```

sub_sell_summation: Stock_Order_Book × Price → **Nat**
sub_sell_summation(((sos,_),(ns,_)),p) ≡ ns +
  let ps = {p′|p′:Prices • p′ ∈ **dom** sos ∧ p′ > p} **in** accumulate(sos,ps)(0) **end**

sub_buy_summation: Stock_Order_Book × Price → **Nat**
sub_buy_summation(((_,bbs),(_,nb)),p) ≡ nb +
  let ps = {p′|p′:Prices • p′ ∈ **dom** bos ∧ p′ < p} **in** accumulate(bbs,ps)(0) **end**

1.3.4 The TSE Itayose "Algorithm"

16. The TSE practices the so-called Itayose "algorithm" to decide on opening and closing prices[4]. That is, the Itayose "algorithm" determines a single so-called 'execution' price, one that matches sell and buy orders[5]:

17. The "matching sell and buy orders" rules:

    (a) *All market orders must be 'executed'*[6].

    (b) *All limit orders to sell/buy at prices higher/lower*[7] *than the 'execution price'*[8] *must be executed.*

    (c) *The following amount of limit orders to sell or buy at the execution prices must be executed: the entire amount of either all sell or all buy orders, and at least one 'trading unit'*[9] *from 'the opposite side of the order book'*[10].

- The 28 January 2010 version had lines
  - ⋆ $17c'_{\exists}$ name some_priced_buys, should have been, as now, some_priced_sells and
  - ⋆ $17c''_{\forall}$ name all_priced_buys, should have been, as now, all_priced_sells.
- My current understanding of *and assumptions* about the TSE is
  - ⋆ that each buy bid or sell order concerns a number, n, of one or more of the same kind of stocks (i.e. sid).
  - ⋆ that each buy bid or sell order when being accepted by the TSE is assigned a unique order number *on*, and
  - ⋆ that this is reflected in some Sell_Orders or Market_Bids entry being augmented.[11]

[4] [1, pp 59, col. 1, lines 4-3 from bottom, cf. Page 59]
[5] [1, pp 59, col. 2, lines 1–3 and Items 1.–3. after yellow, four line 'insert', cf. Page 59] These items 1.–3. are reproduced as "our" Items 17a–17c.
[6] To execute an order: ?????
[7] Yes, it should be: "higher/lower"
[8] Execution price: ?????
[9] Trading unit: ?????
[10] The opposite side of the order book: ?????
[11] The present, 22.2.2010, model "lumps" all market orders. This simplification must be corrected, as for the Sell_Orders and Market_Bids, the Market_Offers must be modelled as are Orders.

- For current (Monday 22 Feb., 2010) lack of a better abstraction[12], I have structured the Itayose "Algorithm" as follows:
  - ⋆ (17′) either a match can be made based on
    - ◇ all buys and
    - ◇ some sells,
  - ⋆ ($17'_\vee$) or
  - ⋆ (17″) a match can be made based on
    - ◇ aome buys and
    - ◇ all sells.

value
17 match: All_Stocks_Order_Book × Stock_Id → Price-**set**
17 match(asob,sid) **as** ps
17 **pre**: sid ∈ **dom** asob
17 **post**: ∀ p′:Price • p′ ∈ ps ⇒
17′ all_buys_some_sells(p′,ason,sid,ps) ∨
$17'_\vee$ ∨
17″ some_buys_all_sells(p′,ason,sid,ps)

- (17′) The all_buys_some_sells part of the above disjunction "calculates" as follows:
  - ⋆ The all_buys... part includes
    - ◇ all the market_buys
    - ◇ all the buys properly below the stated price, and
    - ◇ all the buys at that price.
  - ⋆ The ...some_sells part includes
    - ◇ all the market_sells
    - ◇ all the sells properly below the stated price, and
    - ◇ some of the buys at that price.

17′ all_buys_some_sells(p′,ason,sid,ps) ≡
17′ ∃ os:On-**set** •
17a′ all_market_buys(asob(sid))
17b′ + all_sub_limit_buys(asob(sid))(p′)
17c′ + all_priced_buys(asob(sid))(p′)
17a′ = all_market_sells(asob(sid))
17b′ + all_sub_limit_sells(asob(sid))(p′)
$17c'_\exists$ + some_priced_sells(asob(sid))(p′)(os)

- (17″) As for the above, only "versed".

[12] One that I am presently contemplating is based on another set of **pre/post** conditions.

```
17″ some_buys_all_sells(p′,ason,sid,ps) ≡
17″   ∃ os:On-set •
17a″        all_market_buys(asob(sid))
17b″      + all_sub_limit_buys(asob(sid))(p′)
17c″      + some_priced_buys(asob(sid))(p′)(os)
17a″    =   all_market_sells(asob(sid))
17b″      + all_sub_limit_sells(asob(sid))(p′)
17c″∀     + all_priced_sells(asob(sid))(p′) ∨
```

The match function calculates a set of prices for each of which a match can be made. The set may be empty: there is no price which satisfies the match rules (cf. Items 17a–17c below). The set may be a singleton set: there is a unique price which satisfies match rules Items 17a–17c. The set may contain more than one price: there is not a unique price which satisfies match rules Items 17a–17c. The single (′) and the double (″) quoted (17a–17c) group of lines, in the match formulas above, correspond to the Itayose "algorithm"s Item 17c *'opposite sides of the order book'* description. The existential quantification of a set of order numbers of lines 17′ and 17″ correspond to that "algorithms" (still Item 17c) point of *at least one 'trading unit'*. It may be that the **post** condition predicate is only fullfilled for all trading units – so be it.

```
value
  all_market_buys: Stock_Order_Book → Amount
  all_market_buys((_,(_,mkBuys(nb))),p) ≡ nb

  all_market_sells: Stock_Order_Book → Amount
  all_market_sells((_,(mkSells(ns),_)),p) ≡ ns

  all_sub_limit_buys: Stock_Order_Book → Price → Amount
  all_sub_limit_buys(((,bbs),_))(p) ≡ sub_buy_summation(bbs,p)

  all_sub_limit_sells: Stock_Order_Book → Price → Amount
  all_sub_limit_sells((sos,_))(p) ≡ sub_sell_summation(sos,p)

  all_priced_buys: Stock_Order_Book → Price → Amount
  all_priced_buys((_,bbs),_)(p) ≡ sum(bbs(p))

  all_priced_sells: Stock_Order_Book → Price → Amount
  all_priced_sells((sos,_),_)(p) ≡ sum(sos(p))

  some_priced_buys: Stock_Order_Book → Price → On-set → Amount
  some_priced_buys((_,bbs),_)(p)(os) ≡
    let tbs = bbs(p) in if {}≠os∧os⊆dom tbs then sum(tbs)(os) else 0 end end

  some_priced_sells: Stock_Order_Book → Price → On-set → Amount
  some_priced_sells((sos,_),_)(p)(os) ≡
    let tss = sos(p) in if {}≠os∧os⊆dom tss then sum(tss)(os) else 0 end end
```

The formalisation of the Itayise "algorithm", as well as that "algorithm" [itself], does not guarantee a match where a match "ought" be possible. The "stumbling block" seems to be

the Itayose "algorithm"s Item 17c. There it says: *'at least one trading unit'*. We suggest that a match could be made in which some of the stocks of a candidate trading unit be matched with the remaining stocks also being traded, but now with the stock exchange being the buyer and with the stock exchange immediately "turning around" and posting those remaining stocks as a TSE marked trading unit for sale.

18. It seems to me that the Tetsuo Tamai paper does not really handle
    (a) the issue of order numbers,
    (b) therefore also not the issue of the number of stocks to be sold or bought per order number.
19. Therefore the Tetsuo Tamai paper does not really handle
    (a) the situation where a match "only matches" part of a buy or a sell order.

Much more to come: essentially I have only modelled column 2, rightmost column, Page 59 of [1, Tetsuo Tamai, "TSE"]. Next to be modelled is column 1, leftmost column, Page 60 of [1]. See these same page numbers of the present document !

1.3.5 Match Executions

```
to come
```

1.3.6 Order Handling

```
to come
```

Chapter 2

Bibliographical Notes

Bibliography

[1] T. Tamai. Social Impact of Information System Failures. *Computer, IEEE Computer Society Journal*, 42(6):58–65, June 2009.

Appendix A

Tetsuo Tamai's Paper

For private, limited circulation only, I take the liberty of enclosing Tetsuo Tamai's IEEE Computer Journal paper.

COVER FEATURE

SOCIAL IMPACT OF INFORMATION SYSTEM FAILURES

Tetsuo Tamai, *University of Tokyo*

The social impact of information systems becomes visible when serious system failures occur. A case of mistyping in entering a stock order by Mizuho Securities and the following lawsuit between Mizuho and the Tokyo Stock Exchange sheds light on the critical role of software in society.

Almost daily, we hear news of system failures that have had a serious impact on society. The ACM Risks Forum moderated by Peter Neumann is an informative source that compiles various reported instances of computer-related risks (http://catless.ncl.ac.uk/risks).

One of journalism's shortcomings is that it makes a loud outcry when trouble occurs with a computer-based system, but it remains silent when nothing goes wrong. This gives the general public the wrong impression that computer systems are highly unreliable. Indeed, as software is invisible and not easy for ordinary people to understand, they generally perceive software to be something unfathomable and undependable.

Another problem is that when a system failure occurs, news sources offer no technical details. Reporters usually don't have the knowledge about software and information systems needed to report technically significant facts, and the stakeholders are generally reluctant to disclose details. The London Ambulance Service failure case is often cited in software engineering literature because its detailed inquiry report is open to the public, which only emphasizes how rare such cases are (www.cs.ucl.ac.uk/staff/a.finkelstein/las/lascase0.9.pdf).

MIZUHO SECURITIES VERSUS THE TOKYO STOCK EXCHANGE

The case of Mizuho Securities versus the Tokyo Stock Exchange (TSE) is archived in the 12 December 2005 issue of the *Risks Digest* (http://catless.ncl.ac.uk/risks/24.12.html), and additional information can be obtained from sources such as the *Times* (www.timesonline.co.uk/tol/news/world/asia/article755598.ece) and the *New York Times* (www.nytimes.com/2005/12/13/business/worldbusiness/13glitch.html?_r=1), among others.

The incident started with the mistyping of an order to sell a share of J-Com, a start-up recruiting company, on the day its shares were first offered to the public. An employee at Mizuho Securities, intending to sell one share at 610,000 yen, mistakenly typed an order to sell 610,000 shares at 1 yen.

 Published by the IEEE Computer Society 0018-9162/09/$25.00 © 2009 IEEE

What happened after that was beyond imagination. The order went through and was accepted by the Tokyo Stock Exchange Order System. Mizuho noticed the blunder and tried to withdraw the order, but the cancel command failed repeatedly. Thus, it was obliged to start buying back the shares itself to cut the loss. In the end, Mizuho's total loss amounted to 40 billion yen ($225 million). Four days later, TSE called a news conference and admitted that the cancel command issued by Mizuho failed because of a program error in the TSE system. Mizuho demanded compensation for the loss, but TSE refused. Then, Mizuho sued TSE for damages.

When such a case goes to court, we can gain access to documents presented as evidence, which provides a rare opportunity to obtain information about the technical details behind system failures. Still, requesting and acquiring documents from the court requires considerable effort by the third party. As it happened, Mizuho contacted me to give an expert opinion, thus I had access to all materials presented to the court. Admittedly, there is always the possibility of bias, but as a scientist, I have endeavored to report this case as impartially as possible.

Another reason for examining this case is that it involved several typical and interesting software engineering issues including human interface design, fail-safety issues, design anomalies, error injection by fixing code, ambiguous requirements specification, insufficient regression testing, subcontracting, product liability, and corporate governance.

WHAT HAPPENED

J-Com was initially offered on the Tokyo Stock Exchange Mother Index on 8 December 2005. On that day, a Mizuho employee got a call from a client telling him to sell a single share of J-Com at 610,000 yen. At 9:27 a.m., the employee entered an order to sell 610,000 shares at 1 yen through a Fidessa (Mizuho's securities ordering system) terminal. Although a "Beyond price limit" warning appeared on the screen, he ignored it (pushing the Enter key twice meant "ignore warning" by the specification), and the order was sent to the TSE Stock Order System. J-Com's outstanding shares totaled 14,500, which means the erroneous order was to sell 42 times the total number of shares.

At 9:28 a.m., this order was displayed on the TSE system board, and the initial price was set at 672,000 yen.

Price determination mechanism

TSE stock prices are determined by two methods: *Itayose* (matching on the board) and *Zaraba* (regular market). The Itayose method is mainly used to decide opening and closing prices; the Zaraba method is used during continuous auction trading for the rest of the trading session. In the J-Com case, the Itayose method was used as it was the first day of determining the J-Com stock price.

There are two order types for selling or buying stocks: *market orders* and *limit orders*. Market orders do not specify the price to buy or sell and accept the price the market determines, while limit orders specify the price. When sell and buy orders are matched to execute trading, market orders of both sell and buy are always given the first priority.

Market participants generally want to buy low and sell high. But when the Itayose method is applied, there is no current market price to refer to, and thus there can be a variety of sell/buy orders, resulting in a wide range of prices. With the Itayose method, a single execution price is determined that matches sell and buy orders by satisfying the following rules:

An employee at Mizuho Securities, intending to sell one share at 610,000 yen, mistakenly typed an order to sell 610,000 shares at 1 yen.

1. All market orders must be executed.
2. All limit orders to sell/buy at prices lower/higher than the execution price must be executed.
3. The following amount of limit orders to sell or buy at the execution price must be executed: the entire amount of either all sell or all buy orders, and at least one trading unit from the opposite side of the order book.

The third rule is complicated but functions as a tie-breaker when the first two rules do not determine a unique price. Looking at an example helps to understand how the rules work.

Table 1 represents an instance of the order book. The center column gives the prices. The left center column shows the volume of sell offers at the corresponding price, while the right center column shows the volume of the buy bids. The volume of the market sell orders and the market buy orders is displayed at the bottom line and at the top line, respectively. The leftmost column shows the aggregate volume of sell offers (working from the bottom to the top in the order of priority), and the rightmost column gives the aggregate volume of buy bids (working from the top to the bottom in the order of priority).

We start by focusing on rules (1) and (2) to determine the opening price. First, the price level is searched where the amounts of the aggregated sell and the aggregated buy cross over. In this case, the line is between 500 yen and 499

COVER FEATURE

Table 1. Order book example illustrating Itayose method.

Sell offer			Buy bid	
Aggregate sell orders	Shares offered at bid	Price (yen)	Buy offers at bid	Aggregate buy orders
		Market	4,000	
48,000	8,000	502	0	4,000
40,000	20,000	501	2,000	6,000
20,000	5,000	500	3,000	9,000
15,000	6,000	499	15,000	24,000
9,000	3,000	498	8,000	32,000
6,000	0	497	20,000	52,000
	6,000	Market		

yen. These two prices satisfy conditions (1) and (2), so they are the opening price candidates. Then, applying rule (3), the price is finally determined as 499 yen.

Of course, this algorithm does not always determine the price. For example, if the orders are all buy and no sell, there is no solution that satisfies all three rules. An additional mechanism that holds back transactions even if the matching price is found by the Itayose method is a measure to prevent sudden price leaps or drops. On the TSE, an immediate execution only takes place if the next execution price is within a certain range from the previous execution price. The price level determines the range. For example, if the most recently executed price was 500 yen, the next execution price must be within the range of 490-510 yen. In other words, it can only fluctuate up to 10 yen in either direction.

Suppose the matched price is beyond this range—for example, 550 yen when the previous price was 500 yen. Then, execution does not take place; instead a special bid quote of 510 yen is indicated to call for offers at this price. If no offers at this price are received, the special bid quote will be raised to 520 yen after 5 minutes, and so on until equilibrium is achieved. This mechanism is intended to make a smooth transition between widely divergent prices.

But on the morning of 8 December, J-Com had no previous price. In such cases, the publicly assessed value is used in place of the previous price, which was 610,000 yen. Because the matched price was much higher, a special bid quote of 610,000 yen was shown at 9:00 a.m., then raised to 641,000 yen at 9:10 a.m., which means the range was ± 31,000 yen, and raised again to 672,000 yen at 9:20 a.m. Table 2 shows the order book at that moment, when the 1-yen sell offer came in.

Initial price determination

The term "reverse special quote" denotes this particularly rare event. It means that when a special buy bid quote is displayed, a sell order of low price with a significant amount that reverses the situation to a special sell offer quote comes in (or conversely a special sell offer quote is reversed to a special buy bid quote). TSE has another rule that applies to such a case. This rule stipulates that the previous special quote is fixed as the execution price, and the transaction proceeds. Thus, the initial price of J-Com was now determined to be 672,000 yen. In addition to the step price range set for reducing sudden price change, there is also a price limit range for a day. The upper and lower limits of the price for each stock are defined based on the initial price of the day. In the J-Com case, the limits were defined at the moment when the initial price was determined: The upper limit was 772,000 yen, and the lower limit was 572,000 yen.

In regular trading, the price limits are fixed at the start of the market day, and orders with prices exceeding the limit (either upper or lower) are rejected. But when the initial price is determined during the market time, as in the J-Com case, orders received before the price limits are set are not ignored. Instead, the price of an order exceeding the upper limit is adjusted to the upper price limit, and an order under the lower limit is adjusted to the lower price limit. Thus, the 1-yen order by Mizuho was adjusted to 572,000 yen.

Noticing the mistake, Mizuho entered a cancel command through a Fidessa terminal at 9:29:21, but it failed. Between 9:33:17 and 9:35:40, Mizuho tried to cancel the order several times through TSE system terminals that are installed at the Mizuho site, but the cancellations failed. Mizuho called TSE asking for a cancellation on the TSE side, but the answer was no.

At 9:35:33, Mizuho started to buy back J-Com shares. In the end, it could only buy back 510,000 shares; nearly 100,000 shares were bought by others and never restored.

Aftermath

On 12 December, four days after the incident, TSE president Takuo Tsurushima held a press conference and admitted that the order cancellation by Mizuho failed because of a defect in the TSE Stock Order System.

Table 2. Order book for J-Com stock at 9:20 a.m.

Sell offer			Buy bid	
Aggregate	Amount	Price (yen)	Amount	Aggregate
		Market	253	
1,432	695	OVR*	1,479	1,732
737		6,750	4	1,736
737		6,740	6	1,742
737		6,730	6	1,748
737	3	6,720**	28	1,776
734		6,710	2	1,778
734		6,700	3	1,781
734	1	6,690	1	1,782
733		6,680	1	1,783
733	114	UDR***	120	1,903
	619	Market		

* More than 675,000 yen ** Special buy quote *** Less than 668,000 yen

Mizuho could not buy back 96,236 shares, and it was impossible for Mizuho to deliver real shares to those who had bought them. An exceptional measure was taken to settle trading by paying 912,000 yen per share in cash. The result was a 30-billion-yen loss to Mizuho. Mizuho had already suffered a loss of 10 billion yen by buying back 510,000 shares, thus the total loss amounted to 40 billion yen.

Mizuho and TSE started negotiations on compensation for damages in March 2006, but they failed to reach an agreement. Mizuho sent a formal letter to TSE in August 2006 requesting compensation, which TSE declined by sending a letter of refusal.

Mizuho filed a suit against TSE in the Tokyo District Court on 27 October 2006, demanding compensation of 41.5 billion yen. The first oral pleadings took place on 15 December 2006, and trials were held 13 times in two years, the last on 19 December 2008. The court's decision in that trial was scheduled to be given on 27 February 2009, but the court decided to postpone the decision.

In the contract between TSE and each user of the TSE Stock Order System, including Mizuho, there is a clause on exemption from responsibility on the TSE side except when a serious mistake is attributed to TSE. The crucial issue was whether the damage caused by the system defect was due to a serious mistake beyond the range of exemption. TSE also argued that as the incident started with a mistake on the Mizuho side, the mistakes and the resulting damages should be canceled out.

PROBABLE CAUSE

The TSE system unduly rejected the Mizuho order cancellation because the module for processing order cancellation erroneously judged that the J-Com target sell order had been completely executed, thus leaving no transactions to be canceled. This bug had been hiding for five years.

Fujitsu developed the system under contract with TSE and released it for use in May 2000. An evidence document submitted to the court reported that a similar error was found during integration testing in February 2000 and that the current fault occurred as a result of fixing that error.

But there are several mysteries surrounding this apparently simple failure case. Initially, TSE maintained that the target cancellation order could not be found because its price had been changed from 1 yen to the adjusted price of 572,000 yen, whereas the designated cancel price command was the original 1 yen. This explanation is bizarre as it implies that the order data is searched in the database using price as a key when it is obvious that price cannot be a key because there can be multiple orders with the same price. In addition, as this case shows, the price of the same order can be modified during the transaction. This explanation turned out to be wrong, but it came from the fact that there was indeed a logic in the procedure that partly used price to search order data. TSE also maintained that if buy orders did not flow in continuously and thus the target sell orders were not always being matched to buy orders, the order cancel module would not have been invoked within the order matching module but instead invoked in the order entry module, and then the cancellation would have succeeded. However, this explanation implies that different cancel modules are called or the same module behaves differently according to when it is invoked.

The third question, and probably the most crucial one with respect to the direct cause of the error, is how data handling identifies orders causing a reverse special quote. That information is written into a database containing

the order book data, but once the information is used in determining the execution price, it is immediately cleared. The rationale behind this design decision is mysterious. The programmer who was charged with fixing the February 2000 bug intended to use this data to judge the type of order to be canceled but he did not know that the data no longer existed.

TSE and Fujitsu claimed that this incident occurred in a highly exceptional situation when the following seven conditions held at the same time:

1. The daily price limits have not been determined.
2. The special quote is displayed.
3. The reverse special quote occurs.
4. The price of the order that has caused the reverse special quote is out of the newly defined daily price limits.
5. The target order of cancellation caused the reverse special quote.
6. The target cancellation order is in the process of sell and buy matching, which forces the cancellation process to wait.
7. The target order is continually being matched.

The order cancellation module appears to have insufficient cohesion as different functions are overloaded.

A general procedure for the order cancellation module would be as follows:

1. Find the order to be canceled.
2. Determine if the order satisfies conditions for cancellation.
3. Execute cancellation if the conditions are met.

Because each order has a few simple attributes—stock name, sell or buy, remaining number of shares to be processed (if 0, the order is completed), and price—the condition that an order can be canceled is straightforward: "the remaining number of shares to be processed is greater than zero." There is only one other condition that cannot be determined by the order attribute data but can be determined by its execution state: If the target order is in the process of matching, the cancel process must wait.

A remarkable point to note is that factors such as undefined limit price, display of special quote, reversing special quote, price adjustment to the limit, and so forth have no influence on the cancellation judgment. Thinking in this way, it seems that the system design artificially introduced the seven complicated conditions listed by TSE and Fujitsu.

DESIGN ANOMALY

Figure 1 shows a flowchart of the module that handles order cancellation. Because order cancellation and order change are processed in the same way, the two functions are overloaded in this same module, but for the sake of simplicity I only deal with order cancellation.

The flowchart is not shown to provide details but to illustrate the kind of documents presented to the court. It is extracted and modified from a document submitted as evidence by the defendant, which was an analysis of the error reported by a TSE system engineer. The plaintiff required the defendant to provide the entire design specification and source code, but the defendant refused and the judge did not force the issue, being reluctant to go into technical details in court.

Part A of the flowchart deals with the logic of price adjustment to limit if necessary. The decision logic is as follows:

- if the order to cancel is sell and the price is lower than the lower limit, it is adjusted to the lower limit; and
- if the order to cancel is buy and the price is higher than the upper limit, it is adjusted to the upper limit.

Part B of the flowchart is the logic inserted in February 2000 when an error was found during testing and caused a failure in December 2005. Its logic is as follows: If called in the order matching process; and limit prices are already set; and the order to cancel is a buy over the limit price or a sell under the limit price and is not a reverse special quote order; then a cancellation is infeasible because all shares are already executed. Although this logic is unduly complicated, it is sound only if all the if-conditions are correctly judged. Unfortunately, the judgment on "if not a reverse special quote" gave a wrong answer of "true" in this Mizuho case, and the decision erroneously judged that the cancellation was infeasible.

Insufficient information is available to allow capturing details of the system design, but from what is available we can infer the following design flaws.

Problems in database design

Three databases are related to the problem in this case: Order DB, Sell/Buy Price DB, and Stock Brand DB. The Order DB stores data of all entered orders. This database should include the current attributes of each order, including those necessary for judging whether the designated order can be canceled. For example, because there is a record field for the executed shares in this database, determining if all the shares of the order have been executed or not should be a trivial process. However, due to the time gap between usage and update of the data, the process is much more complicated. If the principle of database integrity is respected, the logic would be much clearer, but performance seems to be given higher priority than integrity.

Part A of the flowchart in Figure 1 calculates price adjustment within the cancellation handling module, which implies that the price data in the Order DB does not reflect the current status.

The Sell/Buy Price DB sorts sell/buy orders by price for each stock brand. This is by nature a secondary database constructed from the Order DB. The secondary index is price, but identifying an order uniquely in the database requires the order ID. The explanation that price is used to search the database must refer to search in this database, and the price adjustment logic embedded in the order cancel module should be related to it. The data handling over the Price DB and the Order DB appears to be unduly complicated.

The Stock Brand DB corresponds to a physical order book for each stock, but its substantial data is stored in the Sell/Buy Price DB and only some specific data for each stock brand is kept here. However, to implement a rule that an order that has caused a reverse special quote has an exceptional priority in matching—lower than the regular case—the customer ID and order ID of such a stock is written in this database, and they are cleared as soon as the matching is done. This kind of temporary usage of a database goes against the general principle that a database should save persistent data accessed by multiple modules.

Problems in module design.

The part of the system that handles order cancellation appears to have low modularity. The logic in part B of the flowchart made a wrong judgment because the information telling it that the target order had induced the reverse special quote had been temporarily written on the Stock Brand DB by the order matching module and had already been cleared. This implies an accidental module coupling between the order matching and order cancelling modules.

The order cancellation module appears to have insufficient cohesion as different functions are overloaded. It is not clear how the tasks of searching the target order to be canceled, determining cancellability, executing cancellation, and updating the database are this module's responsibility.

LESSONS LEARNED

In addition to the insights into the associated software design problems, this case provides lessons learned with regard to software engineering technologies, processes, and social aspects.

Safety and human interface

If the order entry system on either the Mizuho or TSE side had been equipped with more elaborate safety measures, the accident could have been avoided. It was not the first time that the mistyping of a stock order resulted in a big loss. For instance, in December 2001, a trader at UBS

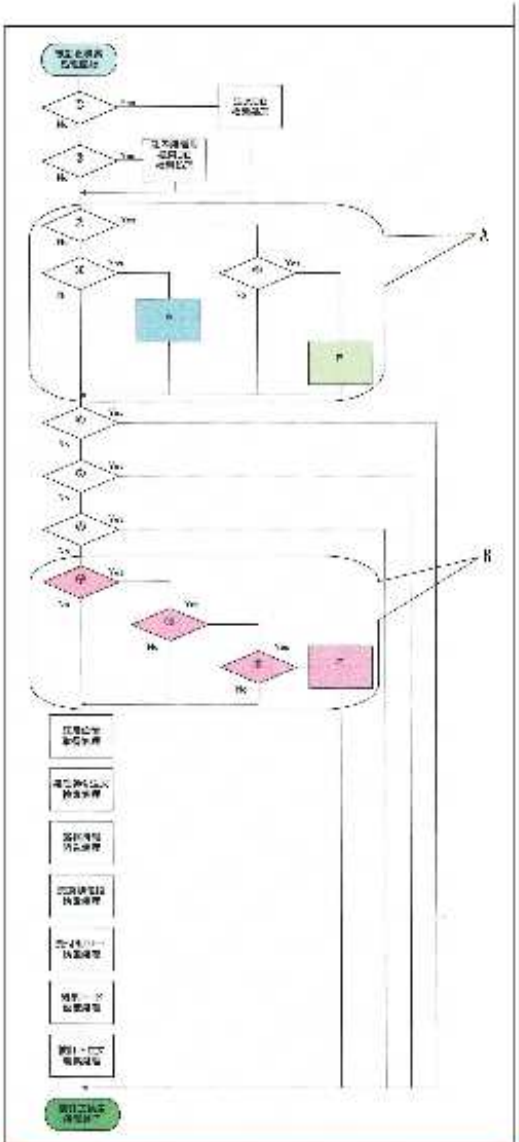

Figure 1. Flowchart of the order cancellation module.

Warburg, the Swiss investment bank, lost more than 10 billion yen while trying to sell 16 shares of the Japanese advertising company Dentsu at 610,000 yen each. He sold 610,000 shares at six yen each. (The similarity between these two cases, including the common figure of 610,000, is remarkable.)

Table 3. Associations between the Software Engineering Code of Ethics and Professional Practice and the TSE-Mizuho case.

Engineering issue	Applicable ACM/IEEE-CS Principle	Ethics clause
Design anomaly	3.01	Strive for high quality, acceptable cost and a reasonable schedule, ensuring significant tradeoffs are clear to and accepted by the employer and the client, and are available for consideration by the user and the public.
	3.14	Maintain the integrity of data, being sensitive to outdated or flawed occurrences.
Safety and human interface	1.03	Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment. The ultimate effect of the work should be to the public good.
Requirements specification	3.07	Strive to fully understand the specifications for software on which they work.
	3.08	Ensure that specifications for software on which they work have been well documented, satisfy the users' requirements and have the appropriate approvals.
Verification and validation	3.10	Ensure adequate testing, debugging, and review of software and related documents on which they work.
Role of user and developer	4.02	Only endorse documents either prepared under their supervision or within their areas of competence and with which they are in agreement.
	5.01	Ensure good management for any project on which they work, including effective procedures for promotion of quality and reduction of risk.
Chain of subcontracting	2.01	Provide service in their areas of competence, being honest and forthright about any limitations of their experience and education.
	3.04	Ensure that they are qualified for any project on which they work or propose to work by an appropriate combination of education and training, and experience.

The habit of ignoring warning messages is common, but it was a critical factor in these cases. It raises the question of how to design a safe—but not clumsy—human interface.

Requirements specification

Development of the current TSE Stock Order System started with the request for proposal (RFP) that TSE presented to the software industry in January 1998. Two companies submitted proposals, and TSE selected Fujitsu as the vendor with which to contract. After several discussions between TSE and Fujitsu, Fujitsu wrote the requirements specification, which TSE approved.

With respect to the order cancellation requirement, it is only mentioned as a function to "Cancel order" in the RFP, and no further details are given there. In the requirements specification, six conditions are listed when cancel (or change) orders are not allowed, but none of them fit the Mizuho case. The document also states that "in all the other cases, change/cancel condition checking should be the same as the current system." Here, "current system" refers to the prior version of the TSE Stock Order System, also developed by Fujitsu, which had been in use until May 2000.

The phrase "the same as the current system" frequently appears in this requirements specification, which was criticized by software experts after the Mizuho incident was publicized. The phrase may be acceptable if there is a consensus between the user and the developer on what it means in each context, but when things go wrong, the question arises whether the specification descriptions were adequate.

Verification and validation

The fact that an error was injected while fixing a bug found in testing is so typical that every textbook on testing warns about this possibility. It is obvious that regression testing was not properly done. It is perhaps too easy to criticize this oversight, but it would be worthwhile to study why it happened in this particular case. So far, not many details have been disclosed.

Role of user and developer

It is conceivable that communication between the user and the developer was inadequate during the TSE system development. The user, TSE, basically did not participate in the process of design and implementation. More involvement of the user during the entire development process would have promoted deeper understanding of the requirements by the developer, and the defect injected during testing might have been avoided.

Subcontracting chain

As in many large-scale information system development projects, the TSE system project was organized in a hierarchical subcontracting structure. The engineer who was in charge of fixing the code in question had a low position in the subcontracting chain. This organizational structure was the likely cause of the misunderstanding about database usage. Such a subcontract structure has often been studied from the industry and labor problem point of view, but it is also important to examine it from the engineering point of view.

Product liability

The extent to which software is regarded as a product amenable to product liability laws may depend on legal and cultural boundaries, but there is a general worldwide trend demanding stricter liability for software. More lawsuits are being filed, and thus software engineers must be more knowledgeable about software product liability issues.

ETHICAL ISSUES

This case raises several questions about professional ethics. However, we should be careful in relating ethical issues and legal matters. Illegal conduct and unethical conduct are of course not equivalent. Moreover, the Mizuho incident is a civil case, not a criminal case.

The Software Engineering Code of Ethics and Professional Practice developed by an ACM and IEEE Computer Society joint task force provides a good framework for discussing ethical issues. The Code comprises eight principles, and each clause is numbered by its principle category and the sequence within the principle. The principles are numbered as 1: Public, 2: Client and Employer, 3: Product, 4: Judgment, 5: Management, 6: Profession, 7: Colleagues, and 8: Self.

As Table 3 shows, some clauses in the Code have relatively strong associations with various aspects of the TSE-Mizuho case. However, this discussion is by no means intended to blame the software engineers who participated in planning, soliciting requirements, designing, implementing, testing, or maintaining the TSE system or other related activities, or to suggest negligence of ethical obligations. First, the Code was not intended to be used in this fashion. Second, the collected facts and disclosed materials are insufficient to precisely judge what kind of specific conduct caused the unfortunate result. However, linking the problems in this case with plausibly related ethical obligation clauses as shown in Table 3 can provide a basis for considering the ethical aspects of this incident and other similar cases.

In addition to individual ethical conduct, the Mizuho-TSE case raises issues pertaining to corporate governance. Why did such an erroneous order by a trader go through unnoticed at Mizuho? Did the TSE staff respond appropriately when they were consulted about the order cancellation? How did Fujitsu manage subcontractors? Corporate governance is another domain where software engineering must deal with social and ethical issues.

If we can learn valuable lessons from this unfortunate incident, it would be beneficial. We should also encourage people who have access to information about similar system failures having significant social impact to analyze and report those cases.

Tetsuo Tamai is a professor in the Graduate School of Arts and Sciences at the University of Tokyo. His research interests include requirements engineering and formal and informal approaches to domain modeling. He received a DrS in mathematical engineering from the University of Tokyo. He is a member of the IEEE Computer Society, the ACM, the Information Processing Society of Japan, and the Japan Society for Software Science and Engineering. Contact him at tamai@graco.c.u-tokyo.ac.jp.