CSE4312F12 Project Solution
ROI

Damien Gruel (cse23089@cse.yorku.ca)
Ludovic Lavalette (cse23088@cse.yorku.ca)

December 3, 2012

Note

- A customer elicitation session was held during class on Tuesday November 6, 2012. If you were not there sure to catch up with a fellow student who was there.
- This template is handed out *caveat emptor*. There may be errors and wrong information. It is ultimately your responsibility to elicit the correct requirements from the customer and to ensure that you satisfy the customer goals and specify correct output from the input.
- Your are required to correct any errors or ambiguities in this template and use this template to produce your final requirements document.

Revisions

Date	Revision	Description
10 October 2012	1.0	Initial customer elicitation
15 November 2012	2.0	Initial Student solution
1 December 2012	3.0	Final Student solution

Contents

List of Figures

List of Tables

1 Context Diagram

The following diagram is the context diagram for the ROI system.

The monitored variables (which are the content of the CSV file, provided by the user), are :

- an header, which is composed of a required name, an optional description of the file and optional information about the customer (account number, email, address and phone number)
- the evaluation dates (*start* and *end*)
- the tuple data (*date*, *market value*, *cash flow*, *agent fees* and *benchmark*).

The format of the output is the following (whole input = everything between the earliest date and the latest date in the sequence of tuple data):

Name: ??

Whole input: yyyy-mm-dd to yyyy-mm-dd
TWR: ?? %
ROI: ?? %
Benchmark: ?? %

Evaluation Period: yyyy-mm-dd to yyyy-mm-dd
TWR: ?? %
ROI: ?? %
Benchmark: ?? %

The controlled variables are also a warning (if a calculation is not possible, if the evaluation period is not valid, or if the portfolio history has no name) and an error (if the CSV file is not valid).

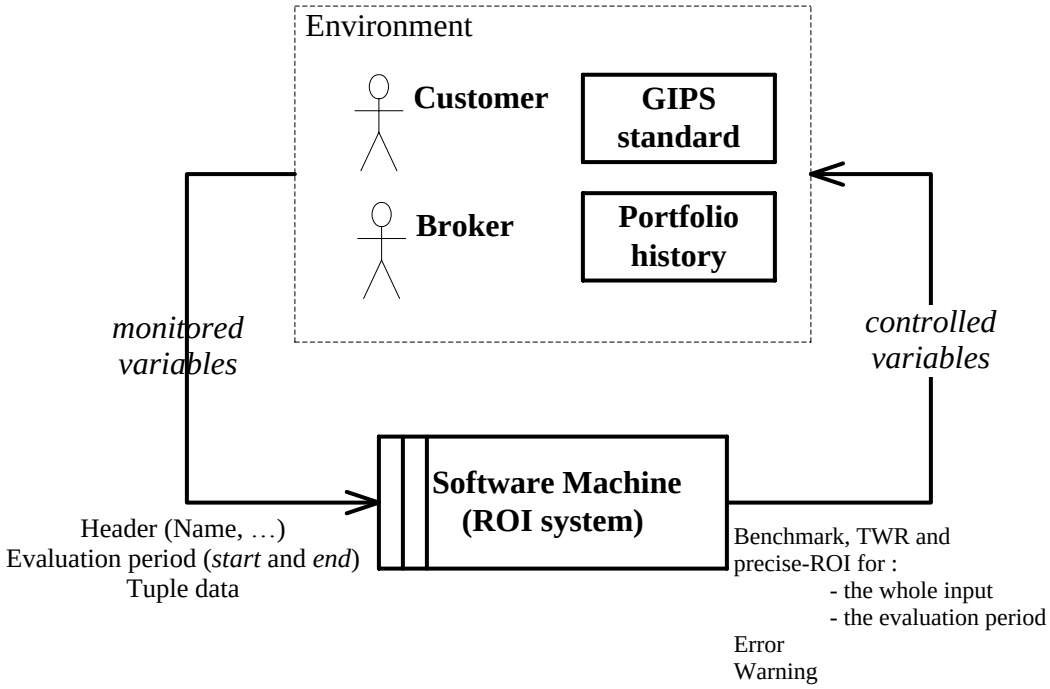

Figure 1: Context diagram for the ROI system

2 Dictionary

Agent fees: Money that the customer pays to the investment advisor to run the account.

Benchmark: Standard used as a point of reference for evaluating performance.

Cash Flow: Revenue or expense stream that changes a cash account over a given period.

CSV: Comma Separated Value file format used to store tabular data in which numbers and text are stored in plain-text form that can be easily written and read in a text editor.

Customer: The user of the software system.

Evaluation Period: a start and end date (provided by the user) for the portfolio history over which the return on investment is calculated.

GIPS: Global Investment Performance Standards

Investment broker: Runs the portfolio on behalf of the customer and supplies portfolio accounts.

Portfolio statement: List of all investments and current value.

Portfolio History: the historical data of investment performance over time that the customer stores about their investments as gleaned from their monthly or yearly investment accounts. Usually stored by customers in a CSV file (see Figure 1).

ROI: Return On Investment: Performance measure used to evaluate the efficiency of an investment.

TWR: Time Weighted Return: Measure of the compound rate of growth in a portfolio.

Tuple data: *date*, *market value*, *cash flow*, *agent fees* and *benchmark*.

3 E/R-descriptions

3.1 E-descriptions

ID	Description	Comment
E1	Customers create and store a portfolio history, i.e. the historical data of their investment performance as determined from portfolio statements.	
E2	Customers store their portfolio history as a CSV text file. CSV files may be prepared on editors of any operating system and encoded as ANSI or UTF-8.	

Header of the CSV file		
E3.1	Every portfolio history has a name.	
E3.2	Optionally, every portfolio history has a description, account number, email, address, and phone number fields.	

Evaluation period in the CSV file		
E4.1	Optionally, every portfolio has an evaluation period that is between the start and end date of the historical performance data.	See Invariant 1 of TWR_ROI_CALCULATION (Fig. 11)

E4.2	The start date and the end date must be in ISO format (yyyy-mm-dd).	
E4.3	The evaluation period is in range.	See Invariant 1 of TWR_ROI_CALCULATION (Fig. 11)

Data in the CSV file		
E5.1	A portfolio history records investment performance in a non-empty sequence of tuple data, each tuple having the fields: date (required), market value (required), cash flow (optional), agent fees (optional) and benchmark (optional).	See *tr* of TWR_ROI_CALCULATION (Fig. 11)
E5.2	For each tuple, the dates must be in ISO format (yyyy-mm-dd).	
E5.3	When there is a customer contribution, the cash flow is a positive number. For a withdrawal, the number is negative.	
E5.4	Agent fees can be internal (deducted from within the portfolio) or external (additional amounts paid by the customer to the investment broker). The portfolio history reflects only external agent fees, always reported as a non-negative amount.	
E5.5	Every data tuple (row in the CSV file) has a date and a non-negative market value.	See Invariant 2 of TWR_ROI_CALCULATION (Fig. 11)
E5.6	Dates in the tuples are unique and ordered.	See Invariant 3 of TWR_ROI_CALCULATION (Fig. 11)
E5.7	No withdrawal in the tuple data can be greater than the market value.	See Invariant 4 of TWR_ROI_CALCULATION (Fig. 11)
E5.8	An account cannot grow from zero market value and cash flow.	See Invariant 5 of TWR_ROI_CALCULATION (Fig. 11)

E5.9	For each tuple, the market value plus cash-flow plus agent-fees must be non-zero.	See precondition 3 of feature *twr* of TWR_ROI_CALCULATION (Fig. 11)

3.2 R-descriptions

ID	Description	Comment
R1	All return on investment calculations shall follow the GIPS standard.	See twr, roi, benchmark (Fig. 11)

Evaluation period		
R2.1	If no evaluation period is provided, then the start date is the earliest date and the end date the latest date in the sequence of tuple data.	See Start_Valid and End_Valid conditions of the function tables
R2.2	Warning message: If the evaluation dates are not valid, then the following error message shall be displayed to the user: "Invalid evaluation period"	See Function tables

CSV file		
R3.1	Error message: If the CSV file is not valid (i.e. if any of the conditions mentioned above do not hold), then the following error message shall be displayed to the user: "Invalid file".	See Function tables
R3.2	Warning message: If the CSV file does not contain a name, then the following error message shall be displayed to the user: "Incomplete file: absence of name".	See Function table 20

Calculation of the TWR		
R4.1	The system shall provide two TWRs (if each one is calculable) : one for the evaluation period, and one for the whole input.	See Function table 17
R4.2	The TWRs shall be rounded to two decimal places.	

R4.3	If the evaluation period is less than a year, then the TWR shall be reported in absolute terms as a percentage return (i.e. it is not annualized). If the evaluation period is a year or more, then the TWR is annualized to a percentage per year.	See postcondition of *annual_compounded_TWR* of TWR_ROI_CALCULATION (Fig. 11)
R4.4	The annualized TWR shall be reported as a percentage.	See *annual_compounded_TWR* of TWR_ROI_CALCULATION (Fig. 11)
R4.5	Agent fees are treated like a deposit (the agent fees are added to the market value and the cash flow).	See *twr* of TWR_ROI_CALCULATION (Fig. 11)
R4.6	Warning message: If the TWR is not calculable, then a warning message shall be displayed to the user.	See Function table 17

Calculation of the ROI		
R5.1	The system shall provide two ROIs : one for the evaluation period, and one for the whole input.	See Function table 18
R5.2	The ROIs shall be rounded to two decimal places.	
R5.3	The ROI shall be reported as a percentage.	See *roi* of TWR_ROI_CALCULATION (Fig. 11)
R5.4	Agent fees are treated like a deposit (the agent fees are added to the cash flow).	See *roi* of TWR_ROI_CALCULATION (Fig. 11)

Calculation of the Benchmark		
R6.1	The system shall provide two benchmarks (if each one is calculable) : one for the evaluation period, and one for the whole input.	See Function table 19
R6.2	The benchmarks shall be rounded to two decimal places.	

R6.3	The benchmark shall be reported as a compounded ROI, if the benchmark figures are available for the evaluation period.	See *benchmark* of TWR_ROI_CALCULATION (Fig. 11)
R6.4	Warning message: If the benchmark is not calculable, then a warning message shall be displayed to the user.	See Function table 19

4 Mathematical model

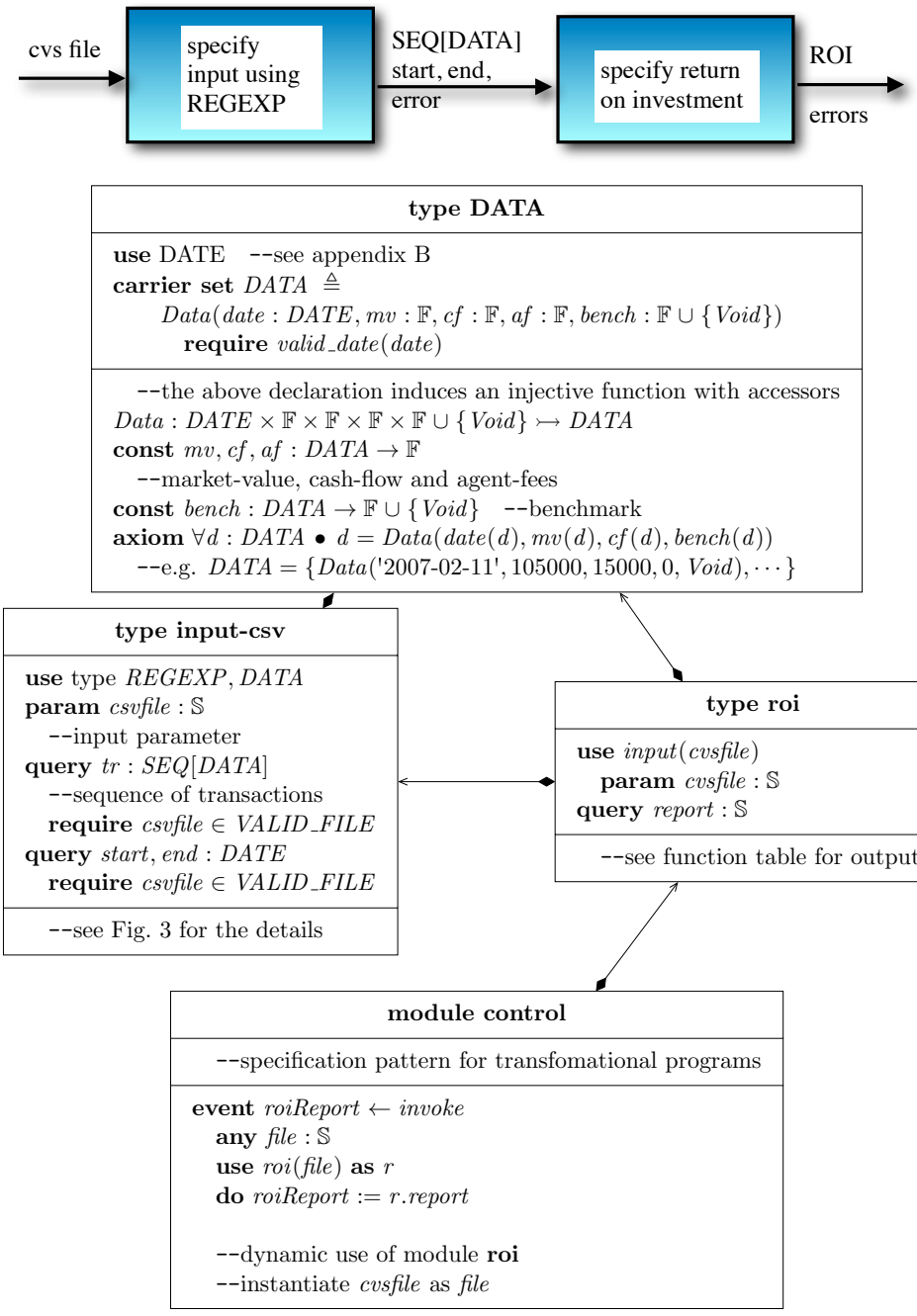

Figure 2: Module specification of return on investment

type input-csv

use type $REGEXP, DATA, DATE$ --we let $\epsilon = \{\text{“”}\}$, eol = {“\n”} etc.
carrier set $DATA \triangleq Data(date : DATE, mv : \mathbb{F}, cf : \mathbb{F}, af : \mathbb{F}, bench : \mathbb{F} \cup \{Void\})$
param $csvfile : \mathbb{S}$ --input parameter
query $tr : SEQ[DATA]$ --sequence of transactions defined by axiom below
  require $csvfile \in VALID\_FILE$
query $start, end : DATE$
  require $csvfile \in VALID\_FILE$

const $VALID\_FILE : REGEXP$
 $\triangleq HEADER \cdot PARAMETERS \cdot \text{eol} \cdot ROW \cdot *(\text{eol} \cdot ROW) \cdot *(\text{“,”}|\text{eol})$
const $HEADER : REGEXP$
 $\triangleq *(HLINE \cdot eol)$
const $HLINE : REGEXP$
 $\triangleq *(\Sigma \backslash \text{eol}) \backslash (EV\_PER \cdot *\Sigma)$
const $PARAMETERS : REGEXP$
 $\triangleq EV\_PER \cdot DATE\_STR \cdot \text{“\_to\_”} \cdot DATE\_STR \cdot *\text{“,”} \cdot \text{eol} \cdot COL\_HEAD$
const $COL\_HEAD : REGEXP$
 $\triangleq + \text{“,”} \cdot eol \cdot$
  “Transaction_Date,Market_Value,Cash_Flow,Agent_Fees,Benchmark” $\cdot *\text{“,”}$
const $EV\_PER : REGEXP \triangleq$ “Evaluation_Period:_”
const $ROW : REGEXP$
 $\triangleq (DATE\_STR \cdot \text{“,”} \cdot FLOAT \cdot \text{“,”} \cdot (FLOAT|\epsilon) \cdot \text{“,”} \cdot (FLOAT|\epsilon) \cdot$
  $\text{“,”} \cdot (FLOAT \cdot \text{“\%”}|\epsilon) \cdot *\text{“,”})$
const $s2d : DATE\_STR \to DATE$ --see birthday book for $DATE$
const $s2f : FLOAT \to \mathbb{F}$ --deferred, $FLOAT$ is the string version of $\mathbb{F}$
const $f2s : \mathbb{F} \to FLOAT$ --deferred, see your favourite programming language
const $d2s : DATE \to DATE\_STR$ --deferred
const $s2optf[G] : (FLOAT|\epsilon) \times G \to \mathbb{F} \cup G$ --string-to-optional float
  where $\forall G \bullet s2optf \in (FLOAT|\epsilon) \times G \to \mathbb{F} \cup G$
  --parameter G is a set such as $\{Void\}$ or a default value such as $\{0\}$
const $f : ROW \to DATA$
dummy $w : ROW$ and $s_0, s_1, s_2, s_3 : \mathbb{S}$
axiom 1: --definition of function f that maps a row string to data
  $w \in (d2s(d) \cdot \text{“,”} \cdot s_0 \cdot \text{“,”} \cdot s_1 \cdot \text{“,”} \cdot s_2 \cdot \text{“,”} \cdot s_3 \cdot *\text{“,”})$
  $\wedge\ (s_4 \cdot \text{“\%”} = s_3 \vee s_4 = s_3 = \epsilon)$
  $\Rightarrow f(w) = Data(d, s2f(s_0), s2optf(s_1, 0), s2optf(s_2, 0), s2optf(s_4, \text{Void}))$
query $error : \mathbb{B} \triangleq textfile \notin VALID\_FILE$ --definition of $tr, start, end$
axiom 2: --definition of $tr, start, end$
  $csvfile \in VALID\_FILE \Rightarrow$
  $(\exists h, foot, s, e : \mathbb{S}; data : SEQ[ROW]$
   $\mid\ h \in HEADER \cdot EV\_PER \cdot s \cdot \text{“\_to\_”} \cdot e \cdot *\text{“,”} \cdot \text{eol} \cdot COL\_HEAD$
   $\wedge\, data \in SEQ[ROW]$
   $\wedge\, end \in *(\text{','}|eol)$
   $\wedge\, textfile \in h \cdot (\cdot i | 0 \le i < \#data \bullet eol \cdot data(i)) \cdot foot$
   $\bullet\ \ tr = (\cdot i | 0 \le i < \#data \bullet < f(data(i)) >$
    $\wedge (start = s2d(s)) \wedge (end = s2d(e))$
)

Figure 3: Type input-csv

TWR_ROI_CALCULATION

--input (*input.csv*)

tr: SEQ[DATA]
--sequence of data [*date, market_value, cash_flow, agent_fees, benchmark*]
--*tr.domain* = {1,2,...,*tr.count*}

count: INTEGER ≜ *tr.count*

dates: SET[DATE] ≜ {$t \in tr \bullet t.date$}

start : DATE ≜ *tr*[1] --first date of the file
end : DATE ≜ *tr*[*count*] --last date of the file

duration: VALUE ≜ *days(end - start)*÷(365.2422)
--years between *start* and *end* calculated by days
--*days*(x) similar to Excel

--output calculation (*input.out.csv*)

di (d:DATE): INTEGER
--index into sequence of transaction for date *d*
require $d \in dates$
ensure $Result \in tr.domain \wedge tr[Result].date = d$

TODO: DANS REQUIRE I VARIE DANS L'INTERVALLE D'INDICE DEFINIT PAR S ET E
IL FAUT VERIFIER QUE CE SOIT BIEN CA DANS LES E/R DESCRIPTIONS
--TWR for the period *s .. e*
twr (s, e: DATE): VALUE
require
$s, e \in dates$
$e > s$
$\forall i \in di(s)+1..di(e) \bullet tr[i-1].mv + tr[i-1].cf + tr[i-1].af \neq 0$
ensure
$Result \triangleq (\Pi i{:}INTEGER \mid di(s) < i \leq di(e) \bullet wealth(i)) - 1$

where *wealth(i)* $\triangleq$ *tr[i].mv÷(tr[i-1].mv + tr[i-1].cf + tr[i-1].af)*

annual_compounded_TWR (s, e: DATE): VALUE
ensure
(duration$\geq 1) \Rightarrow$ *Result* $= ((1 + twr(s, e))^{1 \div duration} - 1) * 100$
(duration<1)$\Rightarrow$ *Result= twr(s, e) * 100*

TODO: IL FAUT RETIRER DES E-R DESCRIPTION QUE
AF EST UN DEPOSIT AILLEUR QUE POUR LE TWR
TODO:VERIFIER QUE LE + POUR LES AGENT FEES
CI DESSOUS EST CORRECT
roi (s, e: DATE): VALUE
require
$s, e \in dates$
$e > s$
ensure
(tr[m].mv+tr[m].cf) $*(1 + Result \div 100)^{days(e-s) \div 365.2422}$
$+ (\Sigma i : \mathbb{N} | m < i < n \bullet (tr[i].cf + tr[i].af) *$
$(1 + Result \div 100)^{days(e-tr[i].date) \div 365.2422}) - tr[n].mv = 0$
where
$m = di(s)$
$n = di(e)$

year(d : DATE): INTEGER
require
$d =$ "$yyyy - mm - dd$"
ensure
$Result = yyyy$

mon(d : DATE): INTEGER
require
$d =$ "$yyyy - mm - dd$"
ensure
$Result = mm$

day(d : DATE): INTEGER

require
$d = "yyyy - mm - dd"$
ensure
$Result = dd$

TODO FAIRE LES FONCTION TABLE POUR MIN ET MAX
TODO VOIR POUR LE REQUIRE DE MIN ET MAX
$\uparrow (f, s$: DATE): DATE
require
ensure
<< see table...>>

$\downarrow (f, s$: DATE): DATE
require
ensure
<< see table...>>

TODO FAIRE ET NUMEROTER LA FUNCTION TABLE
DE LA FCT SUIVANTE
at (d: DATE): DATA
ensure
<< *see table...* >>

bm_calculable (s, e: DATE): BOOL
require
ensure
$Result = at[end \downarrow a].bench \neq void$
$\wedge(\forall d \in DATE | mon(d) = day(d) = 1 \wedge year(s) \leq year(d) \leq year(e)$
$\bullet\ at[d].bench \neq void)$
$\wedge((C \wedge at[s].bench \neq void) \vee (\neg C \wedge at[end \downarrow b].bench \neq void))$
where
$a = "year(s) + 1 - 01 - 01"$
$b = "year(e) + 1 - 01 - 01"$
$C = mon(e) = 1 \wedge day(e) = 1$

TODO RAJOUTER LA FUNCTION TABLE
--the function below return a set of DATA with a benchmark $\neq$ void
bm_seq (s, e: DATE): SEQ[DATA]

require
$s, e \in dates$
$bm\_calculable(s,\ e)$
ensure
$Result = (\oplus i | di(s) < i < di(e) \wedge tr[i].bench \neq void \bullet < tr[i] >) \oplus < t >$
where
t = << see table ...>>

TODO DEFINIR LA FONCTION PO()
$bm\_final\_value(s, e$: DATE): VALUE
require
$s, e \in dates$
$bm\_calculable(s, e)$
ensure
$Result \triangleq$
$tr[m].mv * (\Pi i, seq, t | seq = bm\_seq(s, e) \wedge a \oplus seq[0] \wedge i \geq 1$
$\wedge t = seq[i] \bullet t.bench^{po(s,e,seq[i-1].date,t.date)})$
$+(\Sigma k | m \leq k \leq n \bullet (tr[k].cf - tr[k].af) *$
$(\Pi i, seq, t | seq = bm\_seq(tr[k].date, e) \wedge b(k) \oplus seq \wedge i \geq 1$
$\wedge t = seq[i] \bullet t.bench^{po(s,e,seq[i-1].date,t.date)}))$
where
$m = di(s)$
$n = di(e)$
$--a$ defines seq[0] in the first product
$--b(k)$ defines seq[0] in the second product
$a =< (" year(s) - 01 - 01", 0, 0, 0, void) >$
$b(k) =< (" year(tr[k].date) - 01 - 01", 0, 0, 0, void) >$

$benchmark(s, e$: DATE): VALUE
require
$s, e \in dates$
$bm\_calculable(s, e)$
ensure
$tr[m].mv * (Result + 1)^{days(e-s) \div 365.2422})$
$+(\Sigma k | m \leq k \leq n \bullet tr[k].cf *$
$(Result + 1)^{days(e-tr[k].date) \div 365.2422}) - FV = 0$
where

$m = di(s)$
$n = di(e)$
$FV \triangleq bm\_final\_value(s, e)$

Invariants

(1) *(start<end)∧(start,end∈dates)*
--metadata evaluation period is in range and valid

(2) $\forall t \in tr \bullet t.date \neq Void \wedge t.mv \geq 0$
--every row has a date and a non-negative market value

(3) $\forall i \in 2..count \bullet tr[i].date > tr[i-1].date$
--date are unique and ordered

(4) $\forall t \in tr \bullet t.mv + t.cf \geq 0$
--Cannot withdraw more than the market value

(5) $\forall i \in 2..count \mid tr[i-1].mv = 0 \wedge tr[i-1].cf = 0 \bullet tr[i].mv = 0$
--account coannot grow from zero market value and cash flow

Table 11: Mathematical model for the ROI system

TODO : A CHECKER !!! + TITRE

fi, se : DATE			Min	Max
year(fi) < year(se)			fi	se
year(fi) > year(se)			se	fi
year(fi) = year(se)	mon(fi) < mon(se)		fi	se
	mon(fi) > mon(se)		se	fi
	mon(fi) = mon(se)	day(fi) < day(se)	fi	se
		day(fi) ≥ day(se)	se	fi

Table 12: Function table for ROI system (calculation of the benchmarks)

		at(d)
d∈dates		∃! i | 1≤i≤count • Result = tr[i]∧Result.date = d
d∉dates	mon(d)=1 ∧ day(d)=1	Result = (d,0,0,0,void)
	¬(mon(d)=1 ∧ day(d)=1)	null

Table 13: Function table for ROI system (calculation of the ROIs)

4.1 Function tables

4.1.1 Abbreviations, conditions and messages

Abbreviation	Description
first	tr[1].date
last	tr[tr.count].date
a_c_TWR	annual_compounded_TWR (see TWR_ROI_CALCULATION (Fig. 11))
b	benchmark (see TWR_ROI_CALCULATION (Fig. 11))

Condition	Description
C1	$\forall i \in 2..count \bullet tr[i\text{-}1].mv + tr[i\text{-}1].cf + tr[i\text{-}1].af \neq 0$
C2	$\forall i \in di(start)+1..di(end) \bullet tr[i\text{-}1].mv + tr[i\text{-}1].cf + tr[i\text{-}1].af \neq 0$
C3	benchmark_calculable(first,last) (see TWR_ROI_CALCULATION (Fig. 11))
C4	benchmark_calculable(start,end) (see TWR_ROI_CALCULATION (Fig. 11))
Start_Valid	(start $\in$ dates$\cup${null}) $\wedge$ (start in ISO format) (if Start_Valid $\wedge$ start=null, then start=tr[1].date)
End_Valid	(end $\in$ dates$\cup${null}) $\wedge$ (end in ISO format) (if End_Valid $\wedge$ end=null, then end=tr[tr.count].date)
Name	(csvfile.name = null)

Message	Description
E	"Invalid file"
W	"Invalid evaluation period"
W1	"The TWR for the whole input is not calculable"
W2	"The TWR's are not calculable"
W3	W + W1
W4	"Benchmark for the whole input is not calculable"
W5	"The benchmarks are not calculable"
W6	W + W4
W7	"Incomplete file: absence of name"

4.1.2 Calculation of the TWRs

				Error	Warning	TWR (whole input)	TWR (evaluation period)
Valid_CSV	¬(Start_Valid) ∨ ¬(End_Valid) ∨ end ≤ start	C1		—	W	a_c_TWR(first,last)	—
		¬C1		—	W3	—	—
	Start_Valid ∧ End_Valid ∧ end>start	C1		—	—	a_c_TWR(first,last)	a_c_TWR(start,end)
		¬C1	C2	—	W1	—	a_c_TWR(start,end)
			¬C2	—	W2	—	—
Invalid_CSV				E	—	—	—

Table 17: Function table for ROI system (calculation of the TWRs)

4.1.3 Calculation of the ROIs

		Error	Warning	ROI (whole input)	ROI (evaluation period)
Valid_CSV	¬(Start_Valid) ∨ ¬(End_Valid) ∨ end ≤ start	—	W	—	—
	Start_Valid ∧ End_Valid ∧ end>start	—	—	roi(first,last)	roi(start,end)
Invalid_CSV		E	—	—	—

Table 18: Function table for ROI system (calculation of the ROIs)

4.1.4 Calculation of the benchmarks

				Error	Warning	Benchmark (whole input)	Benchmark (evaluation period)
Valid_CSV	¬(Start_Valid) ∨ ¬(End_Valid) ∨ end ≤ start	C3		—	W	b(first,last)	—
		¬C3		—	W6	—	—
	Start_Valid ∧ End_Valid ∧ end>start	C3		—	—	b(first,last)	b(start,end)
		¬C3	C4	—	W4	—	b(start,end)
			¬C4	—	W5	—	—
Invalid_CSV				E	—	—	—

Table 19: Function table for ROI system (calculation of the benchmarks)

4.1.5 Name of the portfolio history

	Warning
Name	—
¬Name	W7

Table 20: Function table for ROI system (name of the portfolio history)

5 Acceptance tests

Test Case ID	T1 - test_date_invalid.csv
Description	Verify that an invalid date raises an error.
Requirement IDs tested	R3.1
Type	Negative
Initial State	A directory containing the CSV file.
Action	Execute the ROI system on the CSV file
Consequences	Output : "Error: Invalid file."
Test Case ID	T2 - test_date_invalid_February.csv
Description	Verify that an invalid date (because of the leap years) raises an error.
Requirement IDs tested	R3.1
Type	Negative
Initial State	A directory containing the CSV file.
Action	Execute the ROI system on the CSV file
Consequences	Output : "Error: Invalid file."

Test Case ID	T3 - test_tuple_without_date.csv
Description	Verify that a tuple without a date raises an error.
Requirement IDs tested	E5.5 - R3.1
Type	Negative
Initial State	A directory containing the CSV file.
Action	Execute the ROI system on the CSV file
Consequences	Output : "Error: Invalid file."

Test Case ID	T4 - test_negative_market_value.csv
Description	Verify that a negative market value raises an error.
Requirement IDs tested	E5.5 - R3.1
Type	Negative
Initial State	A directory containing the CSV file.
Action	Execute the ROI system on the CSV file
Consequences	Output : "Error: Invalid file."

Test Case ID	T5 - test_dates_non_unique.csv
Description	Verify that two tuples with the same date raise an error.
Requirement IDs tested	E5.6 - R3.1
Type	Negative
Initial State	A directory containing the CSV file.
Action	Execute the ROI system on the CSV file
Consequences	Output : "Error: Invalid file."

Test Case ID	T6 - test_dates_non_ordered.csv
Description	Verify that tuples which are not ordered raise an error.
Requirement IDs tested	E5.6 - R3.1
Type	Negative
Initial State	A directory containing the CSV file.
Action	Execute the ROI system on the CSV file
Consequences	Output : "Error: Invalid file."

Test Case ID	T7 - test_withdraw.csv
Description	Verify that a withdraw which is greater that the market value raises an error.
Requirement IDs tested	E5.7 - R3.1
Type	Negative
Initial State	A directory containing the CSV file.
Action	Execute the ROI system on the CSV file
Consequences	Output : "Error: Invalid file."

Test Case ID	T8 - test_grow.csv
Description	Verify that an account which grow from zero market value and cash flow raises an error.
Requirement IDs tested	E5.8 - R3.1
Type	Negative
Initial State	A directory containing the CSV file.
Action	Execute the ROI system on the CSV file
Consequences	Output : "Error: Invalid file."

6 Requirements Traceability matrix

Requirement ID	**Test Case IDs**
R3.1	T1, T2, T3, T4, T5, T6, T7, T8

A REGEXP

A set of strings is used as the model for regular expressions. We use prefix operators for the Kleene closure (e.g. $*x$ where x is a regular expression such as {'hello'}) and iteration at least one or more (e.g. $+x$) rather than suffix operators. Note that where there is no confusion we use 'hello' instead of {'hello'} where the set is a singleton.

We may use type REGEXP to specify a *FLOAT_STRING* as follows.

$$
\begin{aligned}
\mathit{FLOAT\_STRING} = {} & \text{'+'}\mathit{Inf} && (1)\\
| {} & \text{'-'}\mathit{Inf} && (2)\\
| {} & \mathit{NaN} && (3)\\
| {} & (\text{'-'}|\text{'+'}|\epsilon) \cdot ({}^{*}d \cdot \text{'.'}|\epsilon) \cdot {}^{*}d \cdot ((\text{'e'} \cdot (\text{'-'}|\epsilon) \cdot {}^{+}\, d) \,|\, \epsilon) && (4)\\
d = {} & \text{'0'}|\text{'1'}|\cdots|\text{'9'} && (5)
\end{aligned}
$$

In the above we use the convention that 'e', for example, really stands for the single set {'e'}.

type REGEXP

carrier set $REGEXP$ --set of all regular string expressions
axiom $REGEXP \subseteq \mathbb{P}(\mathbb{S})$
carrier set $\Sigma \triangleq$ {"0", "1", "2", $\cdots$, "a", "b", etc., all printing characters}

dummy $x, y, z : REGEXP$
dummy $s, t, u : \mathbb{S}$
axiom $\forall s \in \Sigma \bullet \{s\} \in REGEXP$
const $0 : REGEXP \triangleq \{\}$ --zero is the unit element of alternation
const $1 : REGEXP \triangleq \{\text{""}\}$ --1 is the unit element of concatenation
--we also use ϵ instead of 1
const infix "|" : $REGEXP \times REGEXP \to REGEXP$
--alternation
const infix " $\cdot$ " : $REGEXP \times REGEXP \to REGEXP$
--concatenation
const prefix " $*$ " : $REGEXP \times REGEXP \to REGEXP$
--iteration zero or more times
const prefix "+" : $REGEXP \times REGEXP \to REGEXP$
--iteration one or more times
axiom $s \in x|y \equiv s \in x \vee s \in y$
theorem $x|0 = 0|x = x$
axiom $s \in x \cdot y \equiv (\exists t, u | s = t \cdot u \bullet t \in x \wedge u \in y)$
--note that $t \cdot u$ is concatenation over $SEQ[\mathbb{S}]$
theorem $1 \cdot x = x \cdot 1 = 1$ --1 is the identity of concatenation
const infix "^": $REGEXP \times \mathbb{N} \to REGEXP$
--use this operator by raising the second argument like an exponent
axiom $x^n = (\cdot i \mid 0 \leq i \leq n \bullet x)$ --concatenation quantifier
--e.g. $x^3 = x \cdot x \cdot x$
theorem $x^0 = 1$
axiom $s \in *x \equiv (\exists n : \mathbb{N} \bullet s \in x^n)$
axiom $s \in +x \equiv (\exists n : \mathbb{N}_1 \bullet s \in x^n)$

Figure 4: Type REGEXP for regular expressions over printing characters

B DATE

The specification of the date module is provided on the following page.

type DATE
carrier set DATE date $\triangleq$ (year, month, day : $\mathbb{N}$) – injective constructor **require** valid_date(year, month, day) **const** year, month, day $\in$ DATE -$\rightarrow$ $\mathbb{N}$ **axiom** $\forall$ d $\in$ DATE $\bullet$ d = date(year(d), month(d), day(d)) **query** valid_date(y,m, d:$\mathbb{N}$) : $\mathbb{B}$ $\triangleq$ <<Table below>> **query** leap_year(y : $\mathbb{N}$) : $\mathbb{B}$ $\triangleq$ mod(y, 4) = 0 $\wedge$ mod(y, 400) $\notin$ 100,200,300) **require** y $\geq$ 1583

Below: *ly* abbreviates *leap_year*

				valid_date
$(1583 \leq y \leq 9999)$	$m \in \{1,3,5,7,8,10,12\}$			true
$\wedge(1 \leq m \leq 12)$	$m \in \{4,6,9,11\}$	$d \leq 30$		true
$\wedge(1 \leq d \leq 31)$		$d > 30$		false
	$m = 2$	$ly(y)$	$d \leq 29$	true
			$d > 29$	false
		$\neg ly(y)$	$d \leq 28$	true
			$d > 28$	false
not the above				false