

2IW81 Final Examination
Software Specification and Architecture
Mathematics and Computer Science
Eindhoven University of Technology
April 15, 2014, 9:00-12:00

Note: It is not allowed to use study material, lecture notes, computers, calculators or mobile phones during the examination.

You should indicate your answers in the exam form itself. Should you require additional paper, please ask the invigilator.

For your convenience summary of Event-B syntax is included as an appendix to this exam.

Part I (40 points)

Answer the following multiple-choice questions. Each question might have multiple correct answers; you should encircle all of them in the exam text below. For each question you get

- 2 points if all correct answers have been indicated and no incorrect answers have been indicated;
- 1 point if some correct answers have been missed but no incorrect answers have been indicated;
- 0 points if at least one incorrect answer has been indicated.

1) VEMUS, Virtual European Music School, was a big European project targeting distance music learning by designing a software system to support both music teachers and learners. Music pieces played by the learners are subject to evaluations both by teachers and by the automated systems. One of the requirements pertaining to the latter reads: "The system should allow for automatic performance evaluation".

- a. This requirement is specific.
- b. This requirement is a functional requirement.
- c. Quality attribute related to this requirement is performance.
- d. This requirement records traceability information.

2) Use case descriptions commonly contain:



- a. Trigger, associations, guarantee.
- b. Precondition, generalizations, main scenario.
- c. Precondition, main scenario, alternative scenarios.
- d. Aggregation, composition, generalization.

3) In an e-commerce application the customers have an option of checking the status of their orders. To check the status of an order customer should be logged in. Use case diagram describing this situation contains two use cases, Login and CheckOrderStatus, and

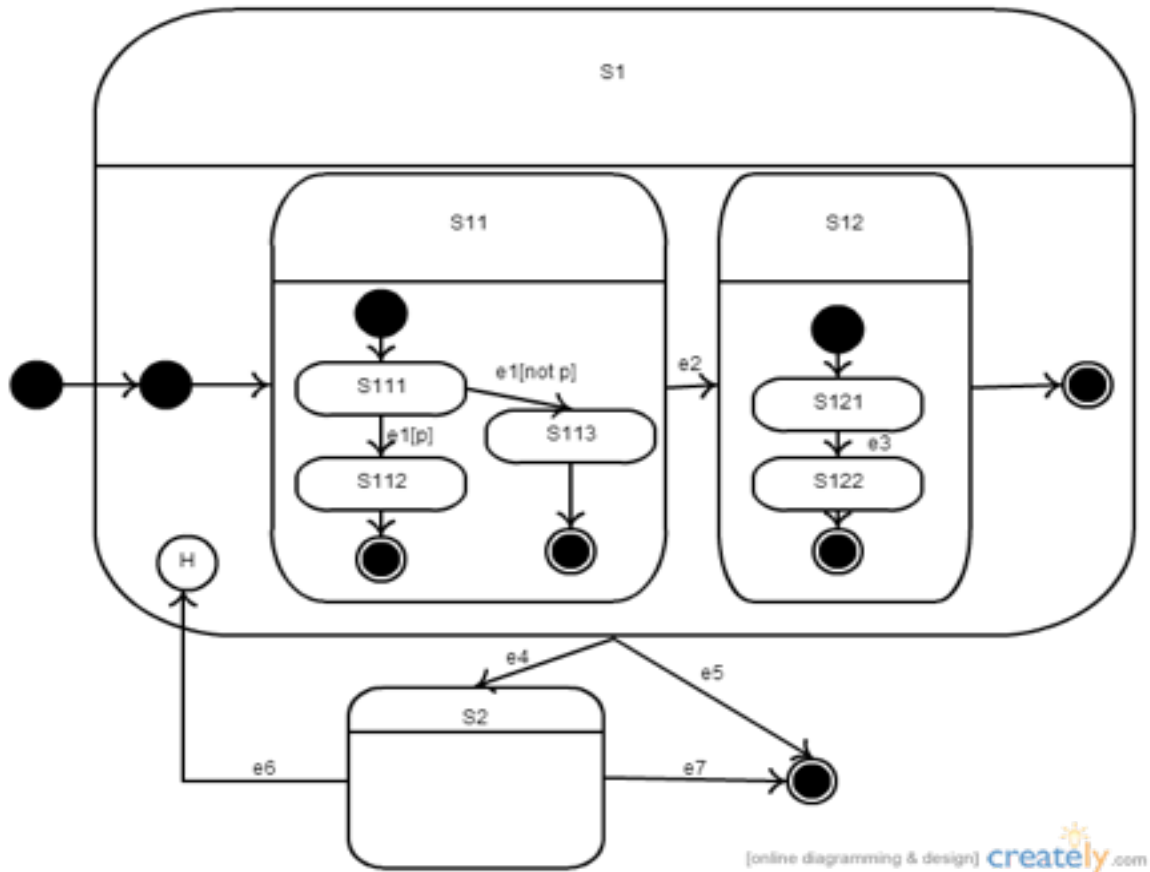
- a. a dashed arrow from Login to CheckOrderStatus with a stereotype <<include>>
- b. a dashed arrow from Login to CheckOrderStatus with a stereotype <<extend>>
- c. a dashed arrow from CheckOrderStatus to Login with a stereotype <<include>>
- d. a dashed arrow from CheckOrderStatus to Login with a stereotype <<extend>>

4) Identify correct statements about activity diagrams:

- a. Activity diagrams have only one ActivityFinal marker  and may have multiple FlowFinal markers .

- b. If an ExpansionRegion is present, it should have at least one FlowFinal marker .
- c. InterruptibleActivityRegion is indicated using the rake symbol .
- d. Pins represent input/output parameters.

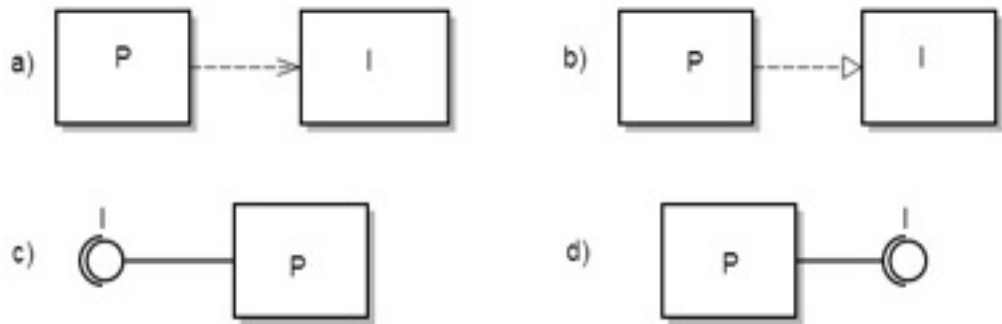
5) Consider the following diagram



What state would be reached after the following sequence of events: e1, e2, e3, e4, e6?

- a. S111
- b. S112 or S113 depending on the value of the guard p
- c. S121
- d. S122

6) Which of the following notations indicate(s) that interface I is provided by class P?

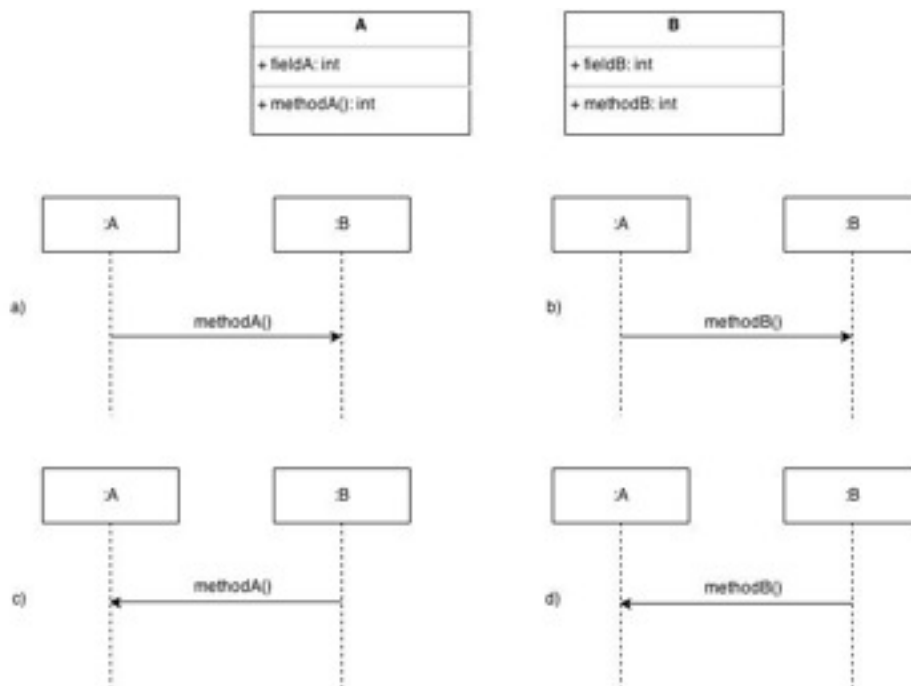


7) Which statements hold for the following diagram



- a. One flight can be assigned only to zero or one planes.
- b. One flight can be assigned to any number of planes.
- c. One plane can be assigned only to zero or one flights.
- d. One plane can be assigned to any number of flights.

8) Which one(s) of the following sequence diagram fragments is consistent with the class diagram fragment?

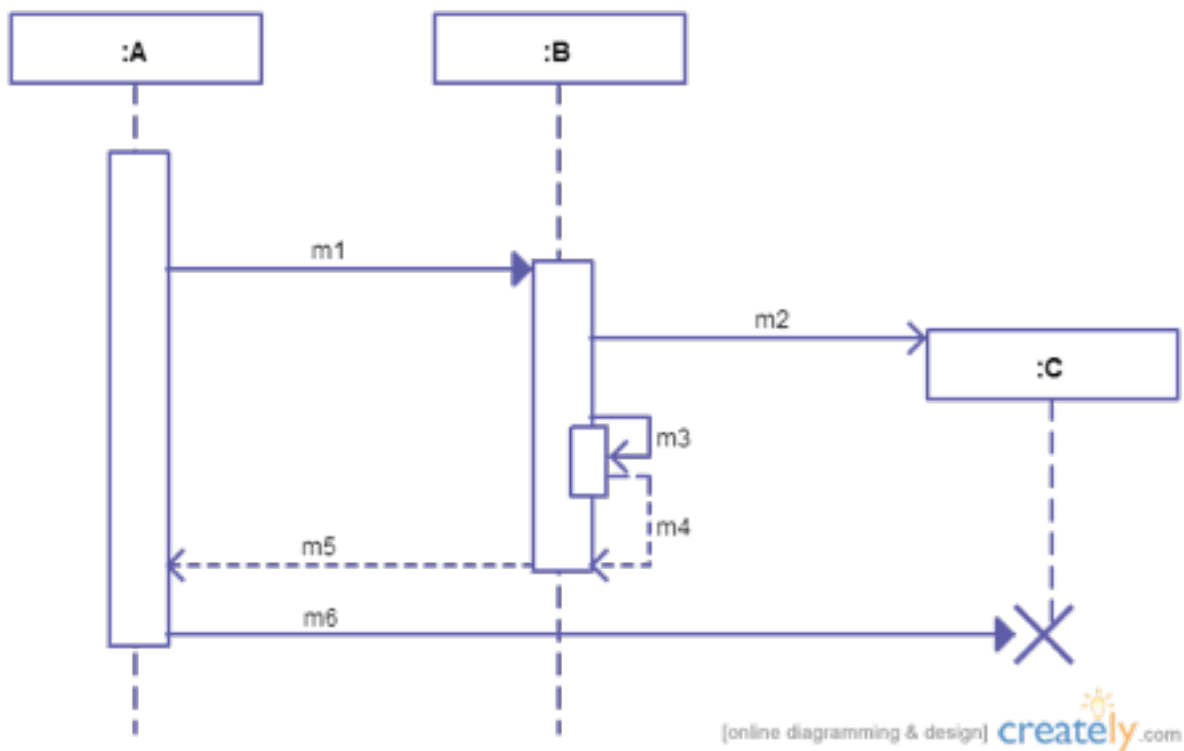


9) Identify correct statements pertaining to deployment diagrams:

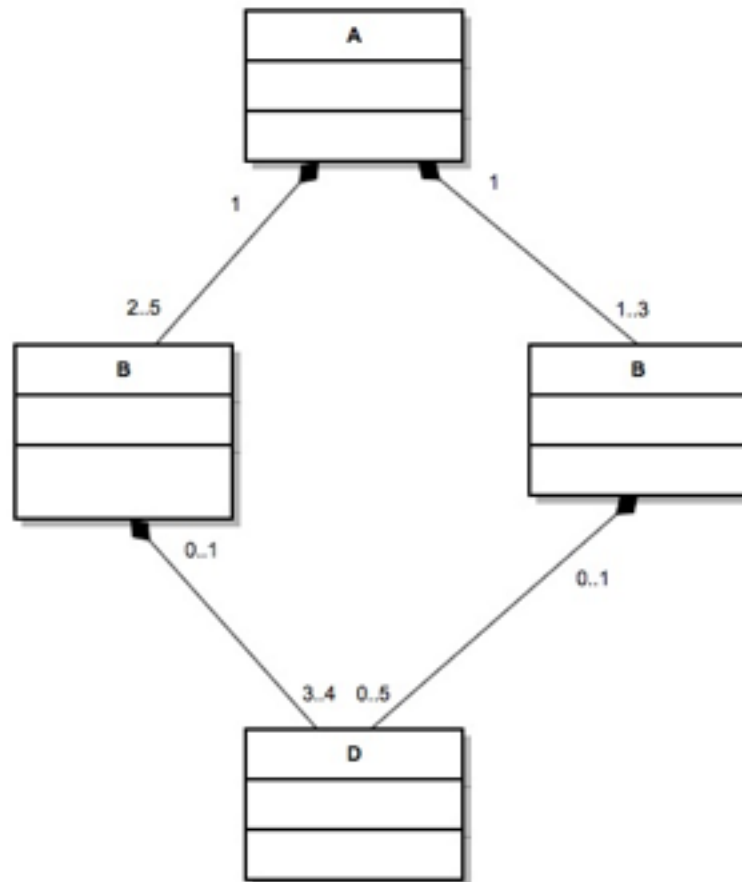
- a. Artefacts are physical elements of the system such as devices and execution environments.
- b. Artefacts: information items produced during software development or when operating the system.
- c. Manifestation maps artefacts to components, use cases, classes, components, packages.
- d. Manifestation maps components, use cases, classes, components to artefacts.

10) Inspect the following diagram: some messages are incorrect. Identify them:

- a. m1
- b. m2
- c. m3
- d. m6



11) See the diagram below. What is the maximum number of instances of D that can be connected to one instance of A?



- a. 12
- b. 0
- c. 30
- d. 35

12) In a timing diagram...

- a. horizontal lines indicate instantaneous change of a state;
- b. a timing ruler should always be present;
- c. time constraint {t1..t2} notation indicates that the event takes place between t1 and t2;
- d. multiple timelines might be present.

13) Choose informal description that corresponds to the following Event-B specification

```
authorized ∈ User ↔ Activity
takeplace ∈ Room ↔ Activity
location ∈ User ↔ Room
```

- A user is authorized to engage in several activities. Each activity takes place in a specific room.
- A user is authorized to engage in several activities. Activities take place in rooms.
- Activities take place in rooms. All users are located in rooms.
- Each user is authorized to engage in only one activity. Users can be located in rooms.

14) Consider the following fragment of an Event-B specification:

```
VARIABLES
  authorized
  takeplace
  location
INVARIANTS
  inv1 : authorized ∈ User ↔ Activity
  inv2 : takeplace ∈ Room ↔ Activity
  inv3 : location ∈ User ↔ Room
  inv4 : ∀u,r · u ↦ r ∈ location ⇒ takeplace[{r}] ⊆ authorized[{u}]
EVENTS
  EnterRoom ▴
  ANY
  user
  room
  WHERE
  grd1 : user ∈ User
  grd2 : room ∈ Room
  THEN
  act1 : location(user) = room
  END
```

Here “ \Rightarrow ” denotes implication. The verification of this specification fails. Choose the guard that should be added to the event “EnterRoom” to guarantee verification of this specification.

- grd3 : $\forall act \cdot room \mapsto act \in takeplace \Rightarrow user \mapsto act \in authorised$
- grd3 : $user \mapsto room \in location$
- grd3 : $takeplace[\{room\}] \subseteq authorized[\{user\}]$
- grd3 : $room \mapsto act \in takeplace \wedge user \mapsto act \in authorized$

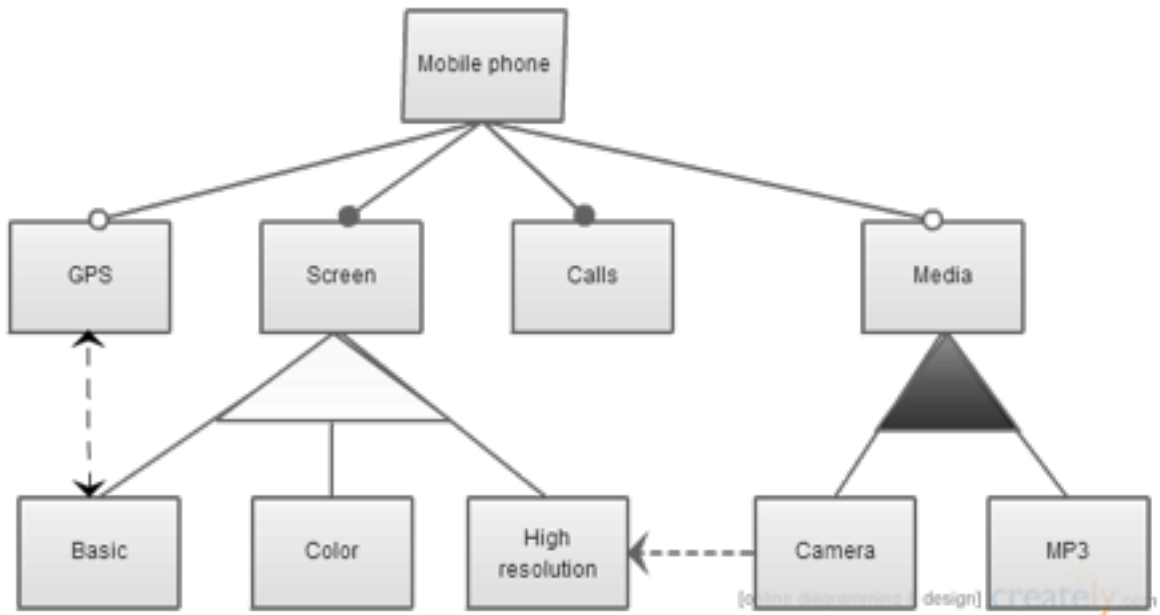
15) [Rozanski and Woods] Company C, an established financial organization, wishes to expand its presence on the Internet with the ability to market a range of financial services to members of the public. These services are aimed at residents of the country where Company C is based, as well as some international customers. Company C plans to contract out the development and operation of the system to an established Web developer.

- a. Assessors of the system are staff from the Web development company.
- b. Assessors of the system include senior managers who will authorize funding for the project.
- c. Assessors of the system include Company C's internal accounting and legal staff, as well as external financial regulators from any country in which Company C wants to trade.
- d. Assessors of the system include ordinary members of the public, who will access the public-facing Web site, along with internal administrative staff, who will carry out its back-office functions.

16) Kruchten's 4+1 process view

- a. covers aspects related to the concurrency viewpoint of Taylor, Medvidović and Dashofy;
- b. covers aspects related to the operational viewpoint of Rozanski and Woods;
- c. can be represented by a UML component diagram;
- d. is a viewpoint in terms of the ISO/IEC/IEEE 42010 standard.

17) Consider the following variability model.



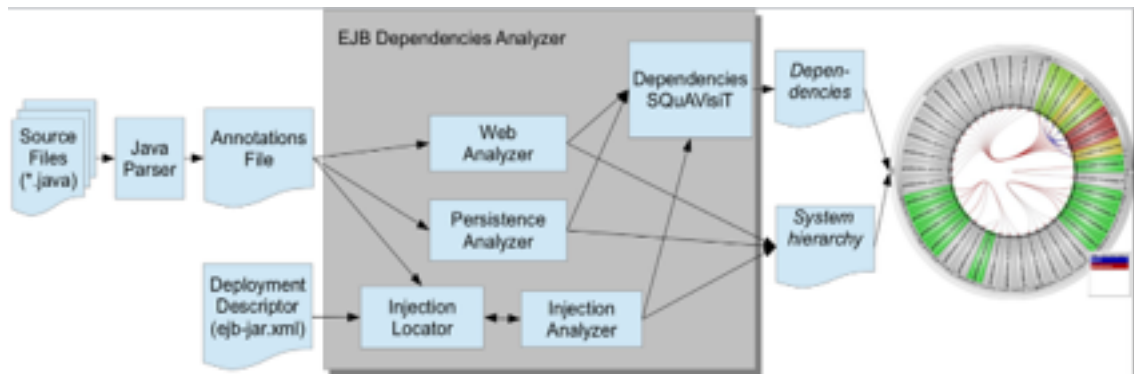
Which combinations of mobile phones' features are allowed by this model?

- a. GPS, Basic screen, Calls
- b. Basic screen, High resolution screen, Calls, MP3
- c. GPS, Color screen, Calls, Camera
- d. High resolution screen, Calls, Camera

18) Indicate true statements pertaining to the Model-View-Controller architectural pattern

- a. Model corresponds to a Boundary class
- b. Model can change without involving Controller
- c. Model-View-Controller enforces event-driven behavior
- d. Model-View-Controller supports only one View per Model.

19) Which architectural style is represented by the following diagram? Circular figure on the right represents both the visualization tool and the visualization obtained.



- Layered style
- Pipes and filters style
- Blackboard style
- Mobile code style

20) The Gnutella protocol is an open decentralized group membership and search protocol. Gnutella has been designed to operate in a dynamic environment, where hosts can join and leave the network frequently. Furthermore, Gnutella is expected to be scalable and reliable, i.e., external attacks should not cause significant data or performance loss. Which one of the following architectural styles would you apply if you have been designing Gnutella?

- Peer-to-peer
- Mobile code
- Client-server
- Blackboard

Part II (60 points)

Part II contains five modeling exercises, by solving each one of them you can earn up to 15 points. You should solve **four exercises out of five**. Please read all the exercises first, choose the ones you are most confident in to work on, and work on those exercises. If you solve all five exercises, we will only grade the four exercises you have solved first.

1. [15 points] Design a variability model corresponding to the following description by Hundt, Mehner, Pfeiffer and Sokenou (2007).

“The security component is intended to provide access control by user authentication and authorization. A user management stores login names and passwords.

Authentication is the minimal function required for access control. If only authentication is requested, the authenticated user has complete access to the application. The minimal requirement for authentication is the login feature. Login is carried out either through a user interface or through operating system user identification. An optional logout feature supports to explicitly log out from the system. After logout, a new user or the same must be able to re-login. Optionally, a timeout is available to logout a user automatically after a configurable time span has elapsed without user interaction. Logout and timeout are not supported if login is realized by operating system user identification.

Authorization is dependent on authentication. For authorization, it is assumed that access restrictions for each user can be specified. Authorization is supposed to cover business objects containing data (per value) but also work flows (per work flow) or certain operations (per operation). Different access rights and different user roles are distinguished.”

2. [15 points] Suppose we are developing a non-web-based information system. This software will be used to collect and process data about certain entities. The requirements on the user interface are not completely clear. We would like to minimize the risks regarding these requirements by splitting each interface into frames. We have two types of interfaces:
- data-entry interfaces: consist of two frames. The main frame displays existing data items in a data-grid. By selecting an item from this data-grid, the second frame will display a detailed view of the item and will allow the user to edit/remove the item. After making a change in the second frame, this change should be reflected immediately in the main frame.
 - information display interfaces: consist of several frames. Each frame offers a different representation for the same data items, e.g., spreadsheet, pie chart, bar chart.

Which architectural patterns/styles would be useful in this situation?

What components of the system would you associate with each part of the chosen architectural patterns/styles?

How would you address the problems in this description by applying these patterns/styles?

3. [15 points] Consider the following ATM system.

The ATM system allows clients to work with bank accounts. To use the ATM a client has to perform authentication. After authentication has been performed, the client gets access to the accounts that he/she is assigned to. The client can perform the following operations with his/her accounts:

- withdraw money from his/her account;
- transfer money from his/her account to any other account;
- deposit money in his/her account;
- stop working with the ATM and with his/her accounts.

The balance variable defines the amount of money stored on accounts. There are two types of accounts: debit and credit. Balance of a debit account must be greater than or equal to zero. Balance of a credit account can be negative, but must not exceed a predefined maximum credit sum.

Money is stored on accounts in different currency. There are four types of currency available: Dollar, Euro, Rupee, and Yuan. The exchange rate defines the rate at which one currency will be exchanged for another when withdrawing, transferring, or depositing money of different currency.

Compose an Event-B specification of this ATM system. Use the following Event-B specification as an initial version (here the specification is not split into the context and the machine for the sake of brevity):

SETS

Clients
Accounts
AccountType

CONSTANTS

Debit
Credit
MaxCredit

AXIOMS

axm1 : partition(AccountType, {Debit}, {Credit})
axm2 : MaxCredit = 1000

VARIABLES

authorized
atm_user // the authorized client of the ATM
clients2accounts // assigning accounts to clients
balance
access // accounts that the atm_user can access

INVARIANTS

inv1 : clients2accounts \in Clients \leftrightarrow Accounts
inv2 : authorized \in BOOL
inv3 : atm_user \in Clients
inv4 : access \subseteq Accounts
inv5 : authorized = FALSE \Rightarrow access = \emptyset
inv6 : authorized = TRUE \Rightarrow access = {x · atm_user \mapsto x \in clients2accounts | x}
inv7 : balance \in Accounts \rightarrow Z

EVENTS

Authentication
ANY client
WHERE ...
THEN ...

```
END
Transfer
  ANY source, destination, amount
  WHERE ...
  THEN ...
  END
Deposit
  ANY destination, amount
  WHERE ...
  THEN ...
  END
Withdraw
  ANY source, amount
  WHERE ...
  THEN ...
  END
Stop
  WHERE ...
  THEN ...
  END
END
```

Add guards (WHERE-section) and actions (THEN-section) to the listed events to specify the ATM functionality. Specify types of the proposed parameters in the guards of the events. Ensure that the invariants hold after each of these events is executed. Add necessary variables, invariants and guards to describe debit/credit types of accounts. Add necessary sets, constants, axioms, variables, invariants, parameters and guards to describe currencies and exchange of currencies.

4. [15 points] Give a class diagram that models the domain of the card game Klondike. Give only classes, associations, and multiplicities. No attributes or methods required.

Klondike is a solitaire game where part of the cards are initially placed on the table in a harp shape, the so-called tableau (see picture). The rest of the cards are placed face down in the deck pile and one by one turned over and either added to the tableau or to the discard pile. Cards have a rank and a suit. During the game, four suit piles are built (in the top right of the picture), one for each suit.



5. [15 points] Consider the following required functionality of an online DVD shop:

Customers should be able to search for DVDs by Title and Category (some examples of categories are: Series, Movies, Music). Once they find a product they like, they can add a DVD to their shopping cart. This requires that they are logged in, therefore if they do not yet have an account, they should be able to create one. When creating an account, a username and password should be chosen, and optionally, they can enter their home address and/or the credit card information they want to use for paying. Credit card information consists of the name on the credit card, the credit card company, the credit card number, and the expiration date.

They can add as many DVDs to the shopping cart as they want, and they can also remove DVDs. If they decide at some point to check out the shopping cart, then they are asked to check the shipping address and credit card information supplied, and if either of those was not supplied, they are asked to supply it. Next, they can review the contents of the shopping cart, and if they agree, they can send the order. Then, the credit card information is sent to the bank, and, if approved, the order is finalised. If the information was not approved, then the customer is made aware of this, so he/she can change the payment option.

A delivery agent can check the status of online orders, and process an order, meaning that it changes its status from “pending” to “delivered”.

Finally, the administrator is responsible for the DVD inventory. She can check the current inventory, order new DVDs from the supplier, and add DVDs to the inventory.

a. Create a UML Use Case Diagram for the online DVD shop. In addition, give a detailed description (pre-condition, trigger, guarantee, main scenario,...) of the use case for “check out shopping cart”.

b. Based on your use case description of “check out shopping cart”, create a UML Sequence Diagram.

c. Create a UML Activity Diagram to depict the business process for processing a DVD order. There are three parties involved in processing an order: Shipping, Online Sales, and Accounting. The process starts when Online Sales receives an order for DVDs from a user. To complete the order, the store needs to charge the credit card and deliver the DVDs. To charge the card, Online Sales sends the credit card information to Accounting, who will then validate and process the credit card. To deliver the DVDs, Shipping will first fill the order, then prepare the package, and finally deliver it. Once the DVDs are delivered and the credit card is charged, the order is closed.

This page has been intentionally left blank

A Concise Summary of the Event-B mathematical toolkit ¹

Each construct will be given in its presentation form, as displayed in the Rodin toolkit, followed by the ASCII form that is used for input to Rodin.

In the following: P, Q and R denote *predicates*;

x and y denote single variables;

z denotes a single or comma-separated list of variables;

p denotes a pattern of variables, possibly including \mapsto and parentheses;

S and T denote set expressions;

U denotes a set of sets;

m and n denote integer expressions;

f and g denote functions;

r denotes a relation;

E and F denote expressions;

E, F is a recursive pattern, *ie* it matches e_1, e_2 and also $e_1, e_2, e_3 \dots$; similarly for x, y ;

Freeness: The meta-predicate $\neg free(z, E)$ means that none of the variables in z occur free in E . This meta-predicate is defined recursively on the structure of E , but that will not be done here explicitly. The base cases are: $\neg free(z, \forall z \cdot P \Rightarrow Q)$, $\neg free(z, \exists z \cdot P \wedge Q)$, $\neg free(z, \{z \cdot P \mid F\})$, $\neg free(z, \lambda z \cdot P|E)$, and $free(z, z)$.

In the following the statement that P *must constrain* z means that the type of z must be at least inferrable from P .

In the following, parentheses are used to show syntactic structure; they may of course be omitted when there is no confusion.

Note: Event-B has a formal syntax and this summary does not attempt to describe that syntax. What it attempts to do is to *explain* Event-B *constructs*. Some words like *expression* collide with the formal syntax. Where a syntactical entity is intended the word will appear in *italics*, e.g. *expression*, *predicate*.

1 Predicates

2 Sets

- | | | | |
|---|---|---|---|
| 1. False \perp | false | 1. Singleton set: $\{E\}$ | {E} |
| 2. True \top | true | 2. Set enumeration: $\{E, F\}$ | {E, F} |
| 3. Conjunction: $P \wedge Q$
Left associative. | P & Q | See note on the pattern E, F at top of summary. | |
| 4. Disjunction: $P \vee Q$
Left associative. | P or Q | 3. Empty set: \emptyset | { } |
| 5. Implication: $P \Rightarrow Q$
Non-associative: this means that $P \Rightarrow Q \Rightarrow R$ must be parenthesised or an error will be diagnosed. | P => Q | 4. Set comprehension: $\{z \cdot P \mid F\}$ | { z . P F } |
| 6. Equivalence: $P \Leftrightarrow Q$
$P \Leftrightarrow Q = P \Rightarrow Q \wedge Q \Rightarrow P$
Non-associative: this means that $P \Leftrightarrow Q \Leftrightarrow R$ must be parenthesised or an error will be diagnosed. | P <=> Q | General form: the set of all values of F for all values of z that satisfy the <i>predicate</i> P . P must <i>constrain</i> the variables in z . | |
| 7. Negation: $\neg P$ | not P | 5. Set comprehension: $\{F \mid P\}$ | { F P } |
| 8. Universal quantification:
$\forall z \cdot P \Rightarrow Q$
Strictly, $\forall z \cdot P$, but usually an implication.
<i>For all values of z, satisfying P, Q is satisfied.</i>
The types of z must be inferrable from the <i>predicate</i> P . | !z.P => Q | Special form: the set of all values of F that satisfy the <i>predicate</i> P . In this case the set of bound variables z are all the free variables in F .
$\{F \mid P\} = \{z \cdot P \mid F\}$, where z is all the variables in F . | |
| 9. Existential quantification:
$\exists z \cdot P \wedge Q$
Strictly, $\exists z \cdot P$, but usually a conjunction.
<i>There exist values of z, satisfying P, that satisfy Q.</i>
The type of z must be inferrable from the <i>predicate</i> P . | #z.P & Q | 6. Set comprehension: $\{x \mid P\}$ | { x P } |
| 10. Equality: $E = F$ | E = F | A special case of item 5: the set of all values of x that satisfy the <i>predicate</i> P .
$\{x \mid P\} = \{x \cdot P \mid x\}$ | |
| 11. Inequality: $E \neq F$ | E /= F | 7. Union: $S \cup T$ | S \vee T |
| | | 8. Intersection: $S \cap T$ | S /\ T |
| | | 9. Difference: $S \setminus T$
$S \setminus T = \{x \mid x \in S \wedge x \notin T\}$ | S \ T |
| | | 10. Ordered pair: $E \mapsto F$
$E \mapsto F \neq (E, F)$
Left associative.
In all places where an ordered pair is required, | E -> F |

¹Version January 23, 2014©1996-2014 Ken Robinson

$E \mapsto F$ must be used. E, F will not be accepted as an ordered pair, it is always a list. $\{x, y \cdot P \mid x \mapsto y\}$ illustrates the different usage.

11. Cartesian product: $S \times T$ **S ** T**
 $S \times T = \{x \mapsto y \mid x \in S \wedge y \in T\}$
 Left-associative.
12. Powerset: $\mathbb{P}(S)$ **POW(S)**
 $\mathbb{P}(S) = \{s \mid s \subseteq S\}$
13. Non-empty subsets: $\mathbb{P}_1(S)$ **POW1(S)**
 $\mathbb{P}_1(S) = \mathbb{P}(S) \setminus \{\emptyset\}$
14. Cardinality: $\text{card}(S)$ **card(S)**
 Defined only for *finite*(S).
15. Generalized union: $\text{union}(U)$ **union(U)**
 The union of all the elements of U .
 $\forall U \cdot U \in \mathbb{P}(\mathbb{P}(S)) \Rightarrow$
 $\text{union}(U) = \{x \mid x \in S \wedge \exists s \cdot s \in U \wedge x \in s\}$
 where $\neg \text{free}(x, s, U)$
16. Generalized intersection: $\text{inter}(U)$ **inter(U)**
 The intersection of all the elements of U .
 $U \neq \emptyset,$
 $\forall U \cdot U \in \mathbb{P}(\mathbb{P}(S)) \Rightarrow$
 $\text{inter}(U) = \{x \mid x \in S \wedge \forall s \cdot s \in U \Rightarrow x \in s\}$
 where $\neg \text{free}(x, s, U)$
17. Quantified union: **UNION z.P | S**
 $\cup z \cdot P \mid S$
 P must *constrain* the variables in z .
 $\forall z \cdot P \Rightarrow S \subseteq T \Rightarrow$
 $\cup(z \cdot P \mid E) = \{x \mid x \in T \wedge \exists z \cdot P \wedge x \in S\}$
 where $\neg \text{free}(x, z, T), \neg \text{free}(x, P), \neg \text{free}(x, S),$
 $\neg \text{free}(x, z)$
18. Quantified intersection: **INTER z.P | S**
 $\cap z \cdot P \mid S$
 P must *constrain* the variables in z ,
 $\{z \mid P\} \neq \emptyset,$
 $(\forall z \cdot (P \Rightarrow S \subseteq T)) \Rightarrow$
 $\cap z \cdot P \mid S = \{x \mid x \in T \wedge (\forall z \cdot P \Rightarrow x \in S)\}$
 where $\neg \text{free}(x, z), \neg \text{free}(x, T), \neg \text{free}(x, P),$
 $\neg \text{free}(x, S).$

2.1 Set predicates

1. Set membership: $E \in S$ **E : S**
2. Set non-membership: $E \notin S$ **E /: S**
3. Subset: $S \subseteq T$ **S <: T**
4. Not a subset: $S \not\subseteq T$ **S /<: T**
5. Proper subset: $S \subset T$ **S <<: T**
6. Not a proper subset: $s \not\subset t$ **S /<<: T**
7. Finite set: *finite*(S) **finite(S)**
 $\text{finite}(S) \Leftrightarrow S$ is *finite*.
8. Partition: *partition*(S, x, y) **partition(S,x,y)**
 x and y partition the set S , ie $S = x \cup y \wedge x \cap y = \emptyset$
 Specialised use for enumerated sets:
 $\text{partition}(S, \{A\}, \{B\}, \{C\}).$
 $S = \{A, B, C\} \wedge A \neq B \wedge B \neq C \wedge C \neq A$

3 BOOL and bool

BOOL is the enumerated set: {FALSE, TRUE} and bool is defined on a predicate P as follows:

1. P is provable: $\text{bool}(P) = \text{TRUE}$
2. $\neg P$ is provable: $\text{bool}(P) = \text{FALSE}$

4 Numbers

The following is based on the set of integers, the set of natural numbers (non-negative integers), and the set of positive (non-zero) natural numbers.

1. The set of integer numbers: \mathbb{Z} **INT**
2. The set of natural numbers: \mathbb{N} **NAT**
3. The set of positive natural numbers: \mathbb{N}_1 **NAT1**
 $\mathbb{N}_1 = \mathbb{N} \setminus \{0\}$
4. Minimum: $\text{min}(S)$ **min(S)**
 $S \subseteq \mathbb{Z}$ and *finite*(S) or S must have a lower bound.
5. Maximum: $\text{max}(S)$ **max(S)**
 $S \subseteq \mathbb{Z}$ and *finite*(S) or S must have an upper bound.
6. Sum: $m + n$ **m + n**
7. Difference: $m - n$ **m - n**
 $n \leq m$
8. Product: $m \times n$ **m * n**
9. Quotient: m/n **m / n**
 $n \neq 0$
10. Remainder: $m \text{ mod } n$ **m mod n**
 $n \neq 0$
11. Interval: $m .. n$ **m .. n**
 $m .. n = \{i \mid m \leq i \wedge i \leq n\}$

4.1 Number predicates

1. Greater: $m > n$ **m > n**
2. Less: $m < n$ **m < n**
3. Greater or equal: $m \geq n$ **m >= n**
4. Less or equal: $m \leq n$ **m <= n**

5 Relations

A relation is a set of ordered pairs; a many to many mapping.

1. Relations: $S \leftrightarrow T$ **S <-> T**
 $S \leftrightarrow T = \mathbb{P}(S \times T)$
 Associativity: relations are *right associative*:
 $r \in X \leftrightarrow Y \leftrightarrow Z = r \in X \leftrightarrow (Y \leftrightarrow Z).$
2. Domain: $\text{dom}(r)$ **dom(r)**
 $\forall r \cdot r \in S \leftrightarrow T \Rightarrow$
 $\text{dom}(r) = \{x \cdot (\exists y \cdot x \mapsto y \in r)\}$
3. Range: $\text{ran}(r)$ **ran(r)**
 $\forall r \cdot r \in S \leftrightarrow T \Rightarrow$
 $\text{ran}(r) = \{y \cdot (\exists x \cdot x \mapsto y \in r)\}$

4. Total relation: $S \leftrightarrow T$ $\boxed{S \leftrightarrow T}$
if $r \in S \leftrightarrow T$ then $\text{dom}(r) = S$
5. Surjective relation: $S \leftrightarrow T$ $\boxed{S \leftrightarrow T}$
if $r \in S \leftrightarrow T$ then $\text{ran}(r) = T$
6. Total surjective relation: $S \leftrightarrow T$ $\boxed{S \leftrightarrow T}$
if $r \in S \leftrightarrow T$ then $\text{dom}(r) = S$ and $\text{ran}(r) = T$
7. Forward composition: $p ; q$ $\boxed{p ; q}$
 $\forall p, q. p \in S \leftrightarrow T \wedge q \in T \leftrightarrow U \Rightarrow$
 $p ; q = \{x \mapsto y \mid (\exists z. x \mapsto z \in p \wedge z \mapsto y \in q)\}$
8. Backward composition: $p \circ q$ $\boxed{p \circ q}$
 $p \circ q = q ; p$
9. Identity: id $\boxed{\text{id}}$
 $S \triangleleft \text{id} = \{x \mapsto x \mid x \in S\}$.
 id is generic and the set S is inferred from the context.
10. Domain restriction: $S \triangleleft r$ $\boxed{S \triangleleft r}$
 $S \triangleleft r = \{x \mapsto y \mid x \mapsto y \in r \wedge x \in S\}$.
11. Domain subtraction: $S \triangleleft r$ $\boxed{S \triangleleft r}$
 $S \triangleleft r = \{x \mapsto y \mid x \mapsto y \in r \wedge x \notin S\}$.
12. Range restriction: $r \triangleright T$ $\boxed{r \triangleright T}$
 $r \triangleright T = \{x \mapsto y \mid x \mapsto y \in r \wedge y \in T\}$.
13. Range subtraction: $r \triangleright T$ $\boxed{r \triangleright T}$
 $r \triangleright T = \{x \mapsto y \mid y \in r \wedge y \notin T\}$.
14. Inverse: r^{-1} $\boxed{r^{-1}}$
 $r^{-1} = \{y \mapsto x \mid x \mapsto y \in r\}$.
15. Relational image: $r[S]$ $\boxed{r[S]}$
 $r[S] = \{y \mid \exists x. x \in S \wedge x \mapsto y \in r\}$.
16. Overriding: $r_1 \triangleleft r_2$ $\boxed{r_1 \triangleleft r_2}$
 $r_1 \triangleleft r_2 = r_2 \cup (\text{dom}(r_2) \triangleleft r_1)$.
17. Direct product: $p \otimes q$ $\boxed{p \otimes q}$
 $p \otimes q = \{x \mapsto (y \mapsto z) \mid x \mapsto y \in p \wedge x \mapsto z \in q\}$.
18. Parallel product: $p \parallel q$ $\boxed{p \parallel q}$
 $p \parallel q = \{x, y, m, n. x \mapsto m \in p \wedge y \mapsto n \in q \mid (x \mapsto y) \mapsto (m \mapsto n)\}$.
19. Projection: prj_1 $\boxed{\text{prj}_1}$
 prj_1 is generic.
 $(S \times T) \triangleleft \text{prj}_1 = \{(x \mapsto y) \mapsto x \mid x \mapsto y \in S \times T\}$.
20. Projection: prj_2 $\boxed{\text{prj}_2}$
 prj_2 is generic.
 $(S \times T) \triangleleft \text{prj}_2 = \{(x \mapsto y) \mapsto y \mid x \mapsto y \in S \times T\}$.

5.1 Iteration and Closure

Iteration and closure are important functions on relations that are not currently part of the kernel Event-B language. They can be defined in a Context, but not polymorphically.

Note: iteration and irreflexive closure will be implemented in a proposed extension of the mathematical language. The operators will be non-associative.

1. Iteration: r^n $\boxed{r^n}$
 $r \in S \leftrightarrow S \Rightarrow r^0 = S \triangleleft \text{id} \wedge r^{n+1} = r ; r^n$.
Note: to avoid inconsistency S should be the finite *base* set for r , ie the smallest set for which all $r \in S \leftrightarrow S$.
Could be defined as a function $\text{iterate}(r \mapsto n)$.
2. Reflexive Closure: r^* $\boxed{r^*}$
 $r^* = \cup n. (n \in \mathbb{N} \mid r^n)$.
Could be defined as a function $\text{rclosure}(r)$.
Note: $r^0 \subseteq r^*$.
3. Irreflexive Closure: r^+ $\boxed{r^+}$
 $r^+ = \cup n. (n \in \mathbb{N}_1 \mid r^n)$.
Could be defined as a function $\text{iclosure}(r)$.
Note: $r^0 \not\subseteq r^+$ by default, but may be present depending on r .

5.2 Functions

A function is a relation with the restriction that each element of the domain is related to a unique element in the range; a many to one mapping.

1. Partial functions: $S \mapsto T$ $\boxed{S \mapsto T}$
 $S \mapsto T = \{r. r \in S \leftrightarrow T \wedge r^{-1} ; r \subseteq T \triangleleft \text{id}\}$.
2. Total functions: $S \rightarrow T$ $\boxed{S \rightarrow T}$
 $S \rightarrow T = \{f. f \in S \mapsto T \wedge \text{dom}(f) = S\}$.
3. Partial injections: $S \mapsto T$ $\boxed{S \mapsto T}$
 $S \mapsto T = \{f. f \in S \mapsto T \wedge f^{-1} \in T \mapsto S\}$.
One-to-one relations.
4. Total injections: $S \mapsto T$ $\boxed{S \mapsto T}$
 $S \mapsto T = S \mapsto T \cap S \rightarrow T$.
5. Partial surjections: $S \mapsto T$ $\boxed{S \mapsto T}$
 $S \mapsto T = \{f. f \in S \mapsto T \wedge \text{ran}(f) = T\}$.
Onto relations.
6. Total surjections: $S \rightarrow T$ $\boxed{S \rightarrow T}$
 $S \rightarrow T = S \mapsto T \cap S \rightarrow T$.
7. Bijections: $S \mapsto T$ $\boxed{S \mapsto T}$
 $S \mapsto T = S \mapsto T \cap S \rightarrow T$.
One-to-one and onto relations.
8. Lambda abstraction: $\boxed{\lambda p. P \mid E}$
 $(\lambda p. P \mid E)$
 P must *constrain* the variables in p .
 $(\lambda p. P \mid E) = \{z. P \mid p \mapsto E\}$, where z is a list of variables that appear in the pattern p .
9. Function application: $f(E)$ $\boxed{f(E)}$
 $E \mapsto y \in f \Rightarrow E \in \text{dom}(f) \wedge f \in X \mapsto Y$, where $\text{type}(f) = \mathbb{P}(X \times Y)$.
Note: in Event-B, relations and functions only ever have one argument, but that argument may be a pair or tuple, hence $f(E \mapsto F)$ $\boxed{f(E \mapsto F)}$
 $f(E, F)$ is never valid.

6 Models

1. Contexts: contain sets and constants used by other contexts or machines.

CONTEXT	Identifier
EXTENDS	Machine_Identifiers
SETS	Identifiers
CONSTANTS	Identifiers
AXIOMS	Predicates
END	

Note: *theorems* can be presented in the AXIOMS part of a context.

2. Machines: contain events.

MACHINE	Identifier
REFINES	Machine_Identifiers
SEES	Context_Identifiers
VARIABLES	Identifiers
INVARIANT	Predicates
VARIANT	Expression
EVENTS	Events
END	

Note: *theorems* can be presented in the INVARIANT section of a machine and the WHERE part of an event.

6.1 Events

Event_name	
REFINES	Event_identifiers
ANY	Identifiers
WHERE	Predicates
WITH	Witnesses
THEN	Actions
END	

There is one distinguished event named *INITIALISATION* used to initialise the variables of a machine, thus establishing the invariant.

Acknowledgement: Jean-Raymond Abrial, Laurent Voisin and Ian Hayes have all given valuable feedback and corrections at various stages of the evolution of this summary.

6.2 Actions

Actions are used to change the state of a machine. There may be multiple actions, but they take effect concurrently, that is, in parallel. The semantics of events are defined in terms of *substitutions*. The substitution $[G]P$ defines a predicate obtained by replacing the values of the variables in P according to the action G . General substitutions are not available in the Event-B language.

Note on concurrency: any single variable can be modified in at most one action, otherwise the effect of the actions would, in general, be inconsistent.

1. *skip*, the null action:
skip denotes the empty set of actions for an event.
2. Simple assignment action: $z := E$ $x := E$
 $:=$ = “becomes equal to”: replace free occurrences of x by E .
3. Choice from set: $x \in S$ $x :: S$
 \in = “becomes in”: arbitrarily choose a value from the set S .
4. Choice by predicate: $z \mid P$ $z \mid P$
 \mid = “becomes such that”: arbitrarily choose values for the variable in z that satisfy the predicate P . Within P , x refers to the value of the variable x before the action and x' refers to the value of the variable after the action.
5. Functional override: $f(x) := E$ $f(x) := E$
Substitute the value E for the function/relation f at the point x .
This is a shorthand:
 $f(x) := E = f := f \triangleleft \{x \mapsto E\}$.