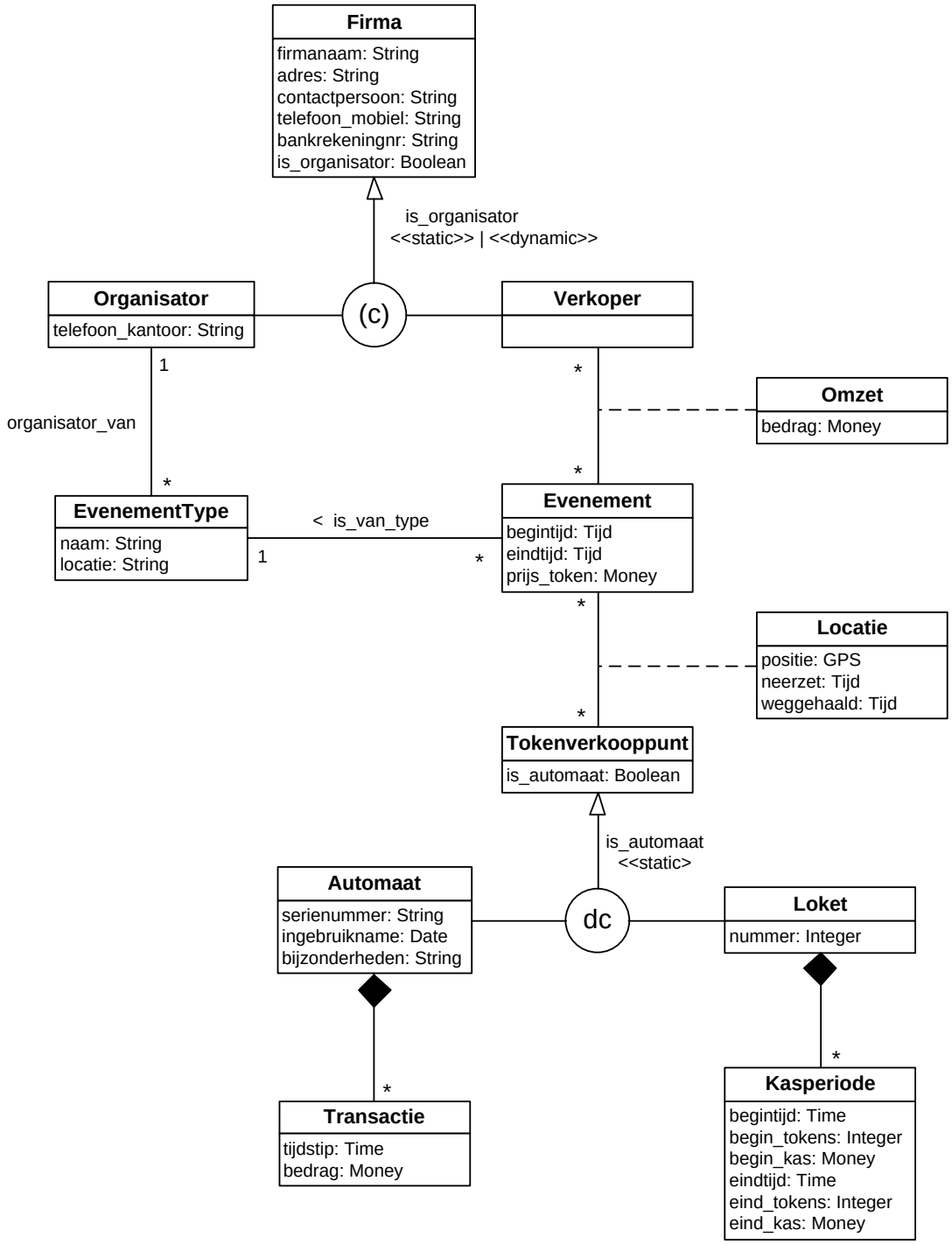
Uitwerking 201300180 Data & Informatie, toets 1, 02.05.2014

1 – Class diagram



Toelichting:

Het ligt voor de hand om Organisator en Verkoper samen te nemen (levert bonuspunten op, als je dit niet hebt kan je nog steeds 40 punten halen). Of deze generalisatie static / dynamic en als dan niet covering is is niet duidelijk, beide is dan goed.

2 – Quality requirements

Er zijn allerlei quality characteristics die voor ieder betaalsysteem van belang zijn, zoals integrity, confidentiality, recoverability. Van systemen die LOC7000 zou kunnen gaan gebruiken kan je eigenlijk verwachten dat ze hieraan voldoen. LOC7000 zal niet een eigen systeem gaan ontwikkelen, maar een systeem gaan gebruiken dat door een of andere financiële dienstverlener wordt aangeboden en ook elders gebruikt wordt. Omdat het gebruik op festivals andere eisen stelt dan gebruik in winkels, is het van belang om juist de contextspecifieke eisen te noemen.
Eisen die in dit verband de hoogste prioriteit hebben zijn

1. **Time behaviour**: Tijdverlies is omzetverlies. Hoe sneller er betaald wordt, hoe meer kan worden verkocht.
2. **Operability** of **User error protection**. Ook bij dronken klanten moet betalen makkelijk zijn en, minstens even belangrijk, volstrekt duidelijk zijn of een betaling wel/niet gelukt is. Tijdens het project werd dit ook wel omschreven als “Idiot-proofness”.
3. **Availability**: Het is zeer ongewenst dat het systeem op een piekmoment uit de lucht is.
4. **Capacity**: Er moet voldoende capaciteit zijn om veel betalingen in korte tijd te kunnen verwerken. (Een centrale instantie die de betalingen verwerkt, zoals Equens, zal daar geen problemen mee hebben, maar locaal zou er een bottleneck kunnen zijn, bijv. door een matige internetverbinding).
5. **Accessability**: Als het de bedoeling is de plastic token af te schaffen, is iedereen dan in staat om op deze manier te betalen (niet iedereen heeft bijvoorbeeld een smartphone).

Andere goed gemotiveerde quality characteristics (zoals Integrity etc.) zijn gedeeltelijk goed gerekend. Confidentiality is helemaal goed gerekend als in de motivatie ingegaan wordt op specifieke omstandigheden bij festivals.
Voorbeelden van goede requirements zijn de volgende (er zijn natuurlijk ook andere mogelijk)

1. 98 % van de betalingen (transacties) dient binnen 2 sec te zijn afgehandeld, *of*
   Elektronisch betalen moet minstens even snel kunnen als betalen met tokens
   (prima requirement, al is het feitelijk onjuist – zie P.S.)
2. Het systeem moet door dronken mensen goed gebruikt kunnen worden
3. Het systeem moet een beschikbaarheid hebben van 99.9 %
4. Het systeem moet minstens 100 transacties tegelijk kunnen verwerken zonder dat de performance beïnvloed wordt.
5. 95 % van de festivalbezoekers moet in staat zijn om het systeem te gebruiken.

P.S.: in 2006 heeft een afstudeerder van de UT daadwerkelijk bij LOC7000 een afstudeerproject gedaan om naar elektronische betaalsystemen te kijken. De conclusie was dat plastic tokens op de middellange termijn zeker niet gaan verdwijnen. Op de genoemde kwaliteitskenmerken zijn ze voorlopig onverslaanbaar, maar ooit zal de druk toenemen om op een goedkoper systeem over te gaan (door de benodigde menskracht is het token-systeem relatief duur). LOC7000 werd geadviseerd door middel van pilots wel ervaring op te doen met nieuwe betaalsystemen, die dan naast tokens gebruikt zouden kunnen worden.

3 – Database schema

```
A(a1, a2,
   primary key (a1)
);
B (a1, b1, b2,
  primary key (a1),
  foreign key (a1) references A(a1)
);
C (a1, c1, c2, ba1, d1, d2,
  primary key (a1),
  unique (c1), check (c1 is not null),
           alternatief: voor deze regel:
           primary key (c1), unique (a1), check (a1 is not null),
  foreign key (a1) references A(a1),
  foreign key (ba1) references B(a1),
  check (ba1 is not null)
);
```

Toelichting:

`T` en `D` zijn identiek. Het is in strijd met de opdracht "geen redundantie" om voor `D` of `T` een aparte tabel te hebben, want vanwege de multipliciteit kan alle `D`- en `T`-data in `C` opgeslagen worden.

De tabel voor `A` kan niet weggelaten worden door zijn attributen in `B` en `C` op te nemen, om twee redenen: (1) het is niet gegeven dat `B` en `C` tezamen `A` overdekken, dus kan sommige `A`-data niet opgeslagen worden, en (2) het is niet gegeven dat `B` en `C` elkaar niet overlappen, dus wordt sommige `A`-data gedupliceerd.

De relaties `R, S , T (= D)` zijn redundant:
Relatie `R` (gezien als een verzameling van paren keys) is als volgt te verkrijgen uit de database:

```
  R = select A.a1 as keyA, B.a1 as keyB from A, B where A.a1 = B.a1;
```

Net zo voor `S , T :`

```
  S = select A.a1 as keyA, C.a1 as keyC from A, C where A.a1 = C.a1;
  T = select C.ba1 as keyB, c.a1 as keyC from C;
```

Relatie `D` gezien als een verzameling van paren keys, is identiek aan `T.`
Relatie `D` gezien als een verzameling van paren keys tezamen met attributen `d1, d2`, is als volgt te verkrijgen uit de database:

```
  D = select C.ba1 as keyB, C.a1 as keyC, C.d1 as d1, C.d2 as d2
from C;
```