Disseny

Lectura dels DC i construcció del MComp

Josep M Merenciano
meren@cs.upc.edu
Departament de Ciències de la Computació
EPSEVG-UPC

Novembre de 2019

Sumari

1 Una observació preliminar

En el que segueix es presenten de forma esquemàtica els aspectes teòrics necessaris per llegir el DC, i per construir a partir d'ells el MComp.

El redactat i l'estructuració permet usar-ho com material de referència. El preu a pagar és que el resultat pot resultar més feixuc de llegir del que potser caldria.

La lectura simultània, en paral·lel, d'aquests apunts teòrics i d'alguna de les solucions pas a pas publicades, ha donar llum al redactat.

2 Lectura dels diagrames de col·laboració (DC)

2.1. Qüestió de noms

Àmbit de visibilitat

- En principi cada DC és un àmbit de visibilitat independent
  - Res impedeix usar el mateix nom en DC diferents per a referenciar diferents elements
- Malgrat tot, sovint s'usa la coincidència de noms en DC diferents per a expressar que hi ha una comunicació entre els ES
  - És a dir, que l'element és el mateix en ambdós DC
- *Davant d'un nom compartit entre dos DC cal:*
  - Observar si hi ha algun motiu clar per considerar que són elements diferents
    - Per exemple, en un DC un nom s'usa per a un identificadr, i en un altre DC s'usa per a referenciar un objecte
  - Determinar si per la semàntica del problema té sentit que siguin el mateix element
    - Exemple. En un ES cerquem un objecte, i en el següent ES iterat, l'anem modificant

Anòmies

- Un objecte sense nom és una anòmia
- L'aparició d'una anòmia en un DC pot significar que:
  - Estem davantd'un singleton

  - En el DC hi ha prou context per saber de quin objecte estem parlant
    - En aquest cas el nom és redundant, i per simplicitat el dissenyador ha preferit ometre'l
    - Compte però: l'objecte té nom; simplement no l'explicitem
- Generalment el component que rep els ES és un singleton
- La resolució de les anòmies (això és, decidir quin és el nom no explicitat) es fa tenint en compte:
  - La semàntica del problema
    - Hem creat un objecte i ara li volem delegar determinada tasca, significa que l'objecte creat i l'objecte que rep el missatge de delegació han de ser el mateix
  - Necessitats de visibilitat
    - Suposem que per poder enviar un determinat missatge necessitem un enllaç dirigit amb destinació `:B`
    - Suposem que en tot el nostre disseny del CU estem usant un únic objecte `:B`, que en un altre moment hem anomenat `b`
    - Llavors l'objecte `:B` que és destinació de l'enllaç dirigit considerat només pot ser `b`

2.2. Objectes vs identificadors

Objectes

- Tot aquell qui en un diagrama de col·laboració (DC) rep un missatge és un objecte
- El `find()` retorna objectes
- L'argument d'un `add()` és un objecte

Identificadors

- Els arguments del `find()` són identificadors

2.3. Components

Detecció de components

- Tot objecte que apareix en algun diagrama de col·laboració (DC) és realització d'un component.
  - Els multiobjectes són una ficció emprada pels DC. No són la realització de cap component.
- Tot identificador ho és d'algun component
  - El nom del component és el que apareix en el mateix DC

2.4. Multiobjectes

Els multiobjectes en els models

- Els `multiobjectes` s'expressen en els DC en forma de pila d'objectes, o com un objecte de tipus `<List>B`
- Els `multiobjectes` no s'expressen en forma de component a MComp.
- La presència d'un `multiobjecte` en un DC indica una visibilitat multiavaluada a MComp

2.5. Visibilitats

Visibilitats de paràmetre

- Les visibilitats de paràmetre s'obtenen directament de les interfícies dels diferents components
  - D'aquí la importància d'explicitar quins són els missatges que pot rebre un component; i si els arguments del missatge són valors, identificadors o objectes

Visibilitats d'atribut

Definició visibilitat d'atribut

- Les visibilitats d'atribut tenen una vigència superior a la del context d'emissió

Casos particulars de visibilitat d'atribut

- Visibilitats d'activació
  - Per comunicar ES diferents
- Visibilitat d'emmagatzematge
  - Un ES emmagatzema un objecte que serà usat en futurs ES, del mateix CU o no
  - Els magatzems, repositoris o similars, són un cas especial d'emmagatzematge. Són objectes la finalitat dels quals és emmagatzemar-ne d'altres
    - Una visibilitat local sobre el magatzem significa que tot el que aquest emmagatzema s'ha de crear i destruir en aquest àmbit local
    - Per tant *cal tenir visibilitat d'atribut sobre els magatzems.*
    - Tot magatzem té una visibilitat multiavaluada d'atribut sobre el component que emmagatzema
      - Tota dada usada en una creació ha de ser recuperable
  - Visibilitat de recuperació
    - Tota dada usada per a l'emmagatzematge ha de ser recuperable

Visibilitats locals

- Tota visibilitat que no sigui de paràmetre i que no tinguem raons per considerar-la d'atribut és una visibilitat local

2.6. Multiobjectes i comunicació

Recepció de missatges

- Quan s'envia un missatge a un multiobjecte, o és un missatge de gruo o és un missatge d'enllaç
- Missatge de grup
  - Són una llista tancada de missatges, que afecten el multiobjecte
  - La semàntica de cadascun dels missatges està definida prèviament
  - La llista engloba:
    - `find()`
    - `add()`
    - `create()`
- Missatge d'enllaç
  - Tot missatge que no sigui de grup és d'enllaç
  - Qui rep el missatge d'enllaç és cadascun dels objectes que conté el multiobjecte
  - El nom, els arguments, i la semàntica dels missatges d'enllaç les defineix el dissenyador

Emissió de missatges

- Si en un DC un multiobjecte emet un missatge, qui realment l'emet és cadascun dels objectes del multiobjecte
- Els arguments explicitats en un missatge que emet un multiobjecte poden ser:
  - Fix
    - Cada objecte del multiobjecte, en enviar el missatge usa el mateix valor com a argument que han usat els altres objectes del multiobjecte
  - Variable
    - Cada objecte del multiobjecte, en enviar el missatge pot usar un valor diferent com a argument del que han usat els altres objectes del multiobjecte

  - Com a argument variable es pot enviar el propi emissor. Per exemple, un multiobjecte `List<A>` pot enviar com a argument `a:A` (o `a` si no es pot produir confusió) per indicar que s'envia a si mateix
  - Similarment podem enviar un atribut de l'emissor. Per exemple, un multiobjecte `List<A>` pot enviar com a argument `a.atr` per indicar que cada objecte emissor envia el seu valor de l'atribut `atr`

Quants multiobjectes

- Un multiobjecte no és un component
- Donat un component `A` podem tenir tants multiobjectes `List<A>` com convingui
- Cal llegir atentament els DC per discernir si dos multiobjectes `List<A>` són dos multiobjectes diferents o no

3 Construcció del Model de components

3.1. Anàlisi dels DC

Procediment de construcció de MComp

- Per cada missatge de cada diagrama de col·laboració cal analitzar i estudiar, no necessàriament en aquest ordre:
  1. Si la comunicació és possible
  2. Arguments d'entrada
  3. Arguments de sortida

Comunicació possible

- Hi ha una visibilitat que permet l'enllaç per on es transmet el missatge
- Tot argument del missatge és accessible des de l'emissor
  - És un literal, un identificador, o bé l'emissor en té una visibilitat d'algun tipus

Arguments d'entrada

- Cal analitzar la tipologia de cada argument:
  - Valor o literal

    No ens diu res
  - Identificador
    - Ens indica l'existència d'un component, però no ens diu res de visibilitats
  - Objecte

  - L'emissor del missatge n'ha de tenir algun tipus de visibilitat
  - El receptor del missatge en té una visibilitat de paràmetre

Arguments de sortida

- L'emissor del missatge té una visibilitat de l'objecte retornat
- El receptor del missatge té una visibilitat (mínim de paràmetre, tot i que també la pot tenir d'atribut) sobre l'objecte retornat
- Cal tenir en compte que tot missatge de creació (`create()`) té com a argument de sortida l'objecte creat

3.2. Dibuix de MComp

Visibilitats vs dependències

- Dependència: Visibilitat local o de paràmetre
  - Es dibuixen en línia discontínua: $-->$
- Visibilitat d'atribut
  - Es dibuixen en línia contínua: $\longrightarrow$

4 Contractes a partir del disseny

Idea bàsica

- Cal assegurar que tot missatge emès té les condicions per ser emès
  - Això afecta a les PRE
- Cal explicar què fa cada missatge
  - Això afecta a les POST
- Cal propagar aquestes condicions als ES
  - De cara a l'especificació només ens interessen els contractes dels ES

Condicions que impedeixen l'emissió d'un missatge

- No existeix l'enllaç dirigit pel qual es transmet
  - Cal una PRE que asseguri l'existència de l'enllaç dirigit
- Per a algun dels seus arguments no existeix l'enllaç dirigit que permet accedir-hi
  - Cal una PRE que asseguri l'existència de l'enllaç dirigit

Utilitat dels arguments

- Tot argument (sigui d'entrada o de sortida) ha de tenir alguna utilitat. És a dir, s'ha d'usar, comunicar o emmagatzemar
  - Ús. Li enviem un missatge
  - Comunicació. Es passa com a argument d'un missatge, o es retorna com a argument de sortida del missatge
  - Emmagatzematge. En el cas dels multiobjectes hi ha un `add()`. En el cas dels emmagatzematges monoavaluats so-

vint no s'explicita l'emmagatzematge perquè es pot concloure per exclusió: si no l'usem ni el comuniquem, és que l'estem emmagatzemant

Un únic MComp

- Independentment dels DC que tinguem, només hi ha un MComp
- Això significa que les conclusions que extraiem d'un DC han de ser compatibles amb la dels altres DC

Casos típics de PRE

Visibilitat d'activació

- Es tracta de comunicar dos ES: un crea l'enllaç dirigit, i l'altre l'usa
- L'usuari de l'enllaç dirigit (qui l'usa per accedir a un argument, o per tansmetre un missatge) necessita una PRE demanant l'existència de l'enllaç dirigit

`find()`

- Sovint es fa un `find()` i tot seguit s'usa el valor rebut
- Si el DC no comprova que aquest valor no sigui nul, llavors cal una PRE demanant l'èxit del `find(id)`

$$\exists a : A \text{ tal que } a.id = id$$

Contextualització del `find()`

- Si tots els objectes `:A` s'emmagatzemen en un mateix multiobjecte, n'hi h prou amb:

$$\exists a : A \text{ tal que } a.id = id$$

- Si un multiobjecte només emmagatzema alguns dels objectes `:A` llavors cal relativitzar la petició:

$$\exists a : A \in \text{multiobjecte tal que } a.id = id$$

Casos típics de POST

`add()`

- L'`add()` afegeix un objecte a un multiobjecte. Tot multiobjecte és un conjunt
- Cal assegurar, en forma de PRE si el DC no ho assegura prèviament, que l'objecte que es vol afegir no s'ha afegit anteriorment
- L'anàlisi es pot reduir a dos casos:
  - Objecte nou
    - Cada nou objecte (aquell obtingut amb un `new`) té un identificador diferent
    - Per tant no cal afegir cap PRE
  - Objecte pre-existent
    - Hem obtingut, amb un `find()` o similar, un objecte, que ara volem afegir en un multiobjecte
    - Cal una PRE per assegurar que l'objecte a afegir no s'ha afegit prèviament

5 MC per enginyeria inversa

Procediment

1. Supressió dels emmagatzemadors
   - Cal suprimir tot component tal que la seva única responsabilitat sigui l'emmagatzematge
     - L'emmagatzematge és propi del disseny, però no té cap mena de sentit en l'especificació
     - Sovint els components dedicats a l'emmagatzematge s'anomenen *magatzem*, *repositoris*,*compendi*, etc
   - Per suprimir-los cal curtircuitar les visibilitats d'atribut incidents amb les sortints (les dependències no ens interessen)
     - Suposem que `M` és un magatzem, i que tenim les visibilitats `A`$\rightarrow$`M` i `M`$\rightarrow$`B`*
     - Com a resultat de la supressió tindrem: `A`$\rightarrow$`B`*
2. Suprimim les dependències
   - Ens quedem només amb les visibilitats d'atribut
3. Convertim les visibilitats en associacions
   - Això implica trobar-ne les multiplicitats i dotar-les de semàntica
4. Suprimim les activacions locals al CU
   - Determinem quines són les associacions que provenen d'una visibilitat d'activació
   - Analitzem si l'activació té sentit que es mantingui fora del CU o no. En cas negatiu, suprimim l'associació
5. Analitzem les obligatorietats i optativitats

Detecció de multiplicitats i optativitats

Fonts d'informació

- Cada problema és diferent. En general però cal cercar la informació en:
  1. DC
  2. Problema
  3. Preferències generals

Anàlisi dels DC

- Cal analitzar la semàntica dels DC
  - Què fan
  - Quin és el seu propòsit
  - Quines precondicions pressuposen
    - Exemple. Si fem un `find()` i no comprovem si ens ha donat un resultat és perquè assumim que en tot moment hi ha un objecte amb el valor d'identificació donat

Anàlisi del problema

- Cal recórrer al nostre coneixeent del problema i analitzar si hi ha alguna PRE que ens afecti
  - Exemple. Una pregunta pot estar en molts controls
- Cal tornar a analitzar els DC assumint les PRE trobades

Preferències generals

- Hi ha casos on ni la lectura dels DC ni el nostre coneixement del problema ens donen la informació necessària
- Dit amb d'altres paraules, hi ha casos on, per exemple, res ens permet decidir entre la multiavaluació o la monoavaluació d'una associació
- En aquests casos recorrem a preferències generals
  - Per exemple, quan dues opcions siguin possibles, ens decantem per la menys marcada
    - Aquest criteri obliga a definir qui és més marcat que un altre