Brillien platform

Fazekas Imre
JUM

Motiváció

- Témakör:
  - telematika
  - elosztott, folyamat-vezérelt szolgáltatások
- Problémák:
  - gondolkodástól távol álló modellek
  - alacsony és/vagy redukált absztrakciós szint
  - erős aránytalanság: komplexitás / lefedett problémakör
  - bonyolult HA, fail over stb. megoldások

Powerobject modell

- OO paradigma matematikai modellje: halmazelmélet
- Cél: halmazelvűség továbbépítése
- Szemlélet: kommunikáció-orientáltság
- Új fogalmak:
  - Presence
  - Unit
  - Context
  - Flow

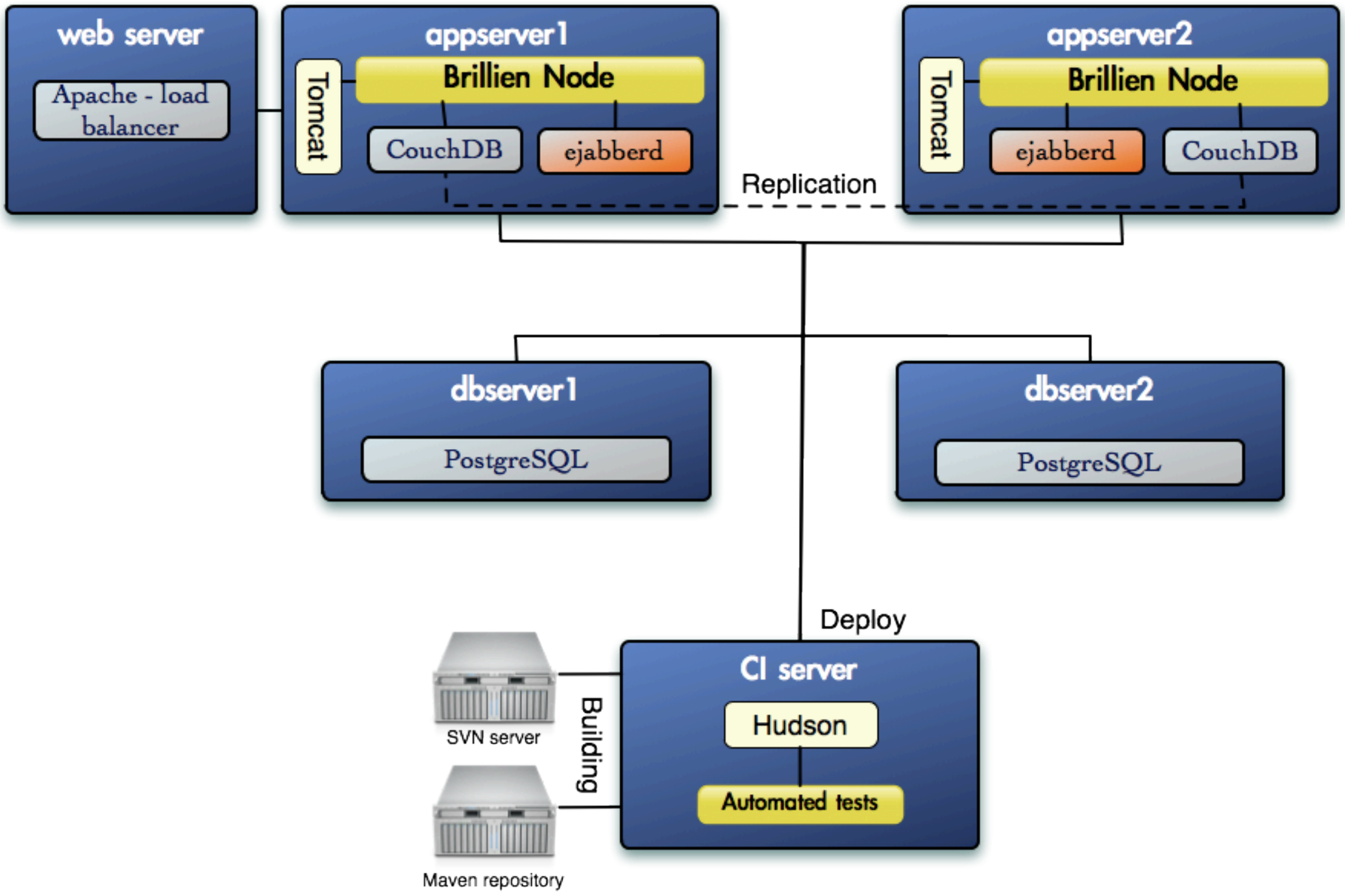
Brillien

- Powerobject implementáció (még részleges)
- Java-alapú alkalmazásszerver
- Üzleti logika rétegbeli megoldás:
    - integrációs eszközöket biztosít
    - nincs saját O/R vagy prezentációs réteg
- Jellemzők:
    - gyorsan tanulható, egyszerű modell
    - hatékony és gyors fejlesztés
    - könnyű integrálás és bevezetés

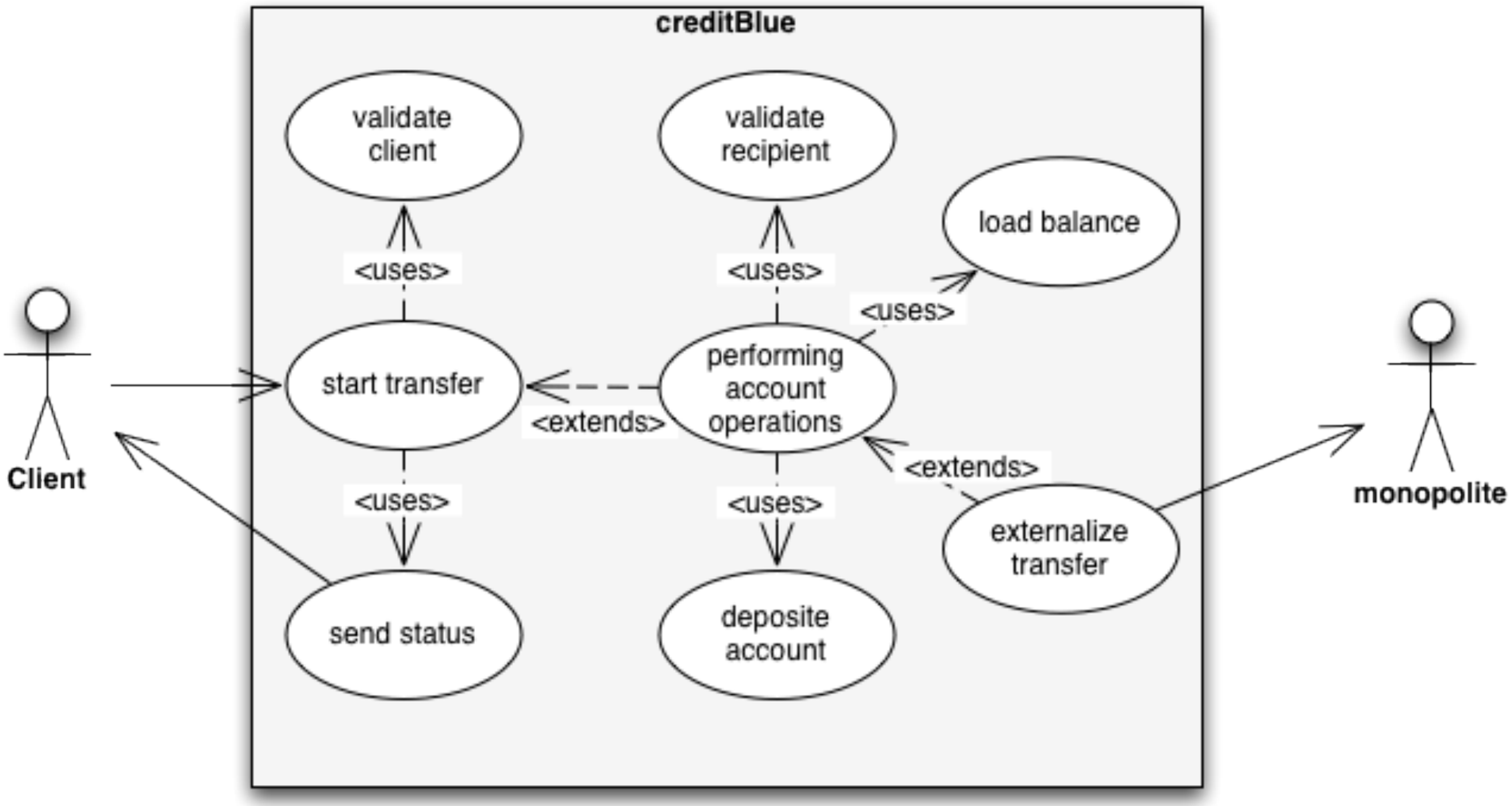
Brillien

- Kommunikációs réteg: XMPP
- Perzisztencia réteg: CouchDB
- Friss technológia
- Absztrakt gondolkodást igényel
- No contract
- Hiányzó funkciók:
  - modellező eszköz
  - web-alapú admin tool
  - IDE integráció

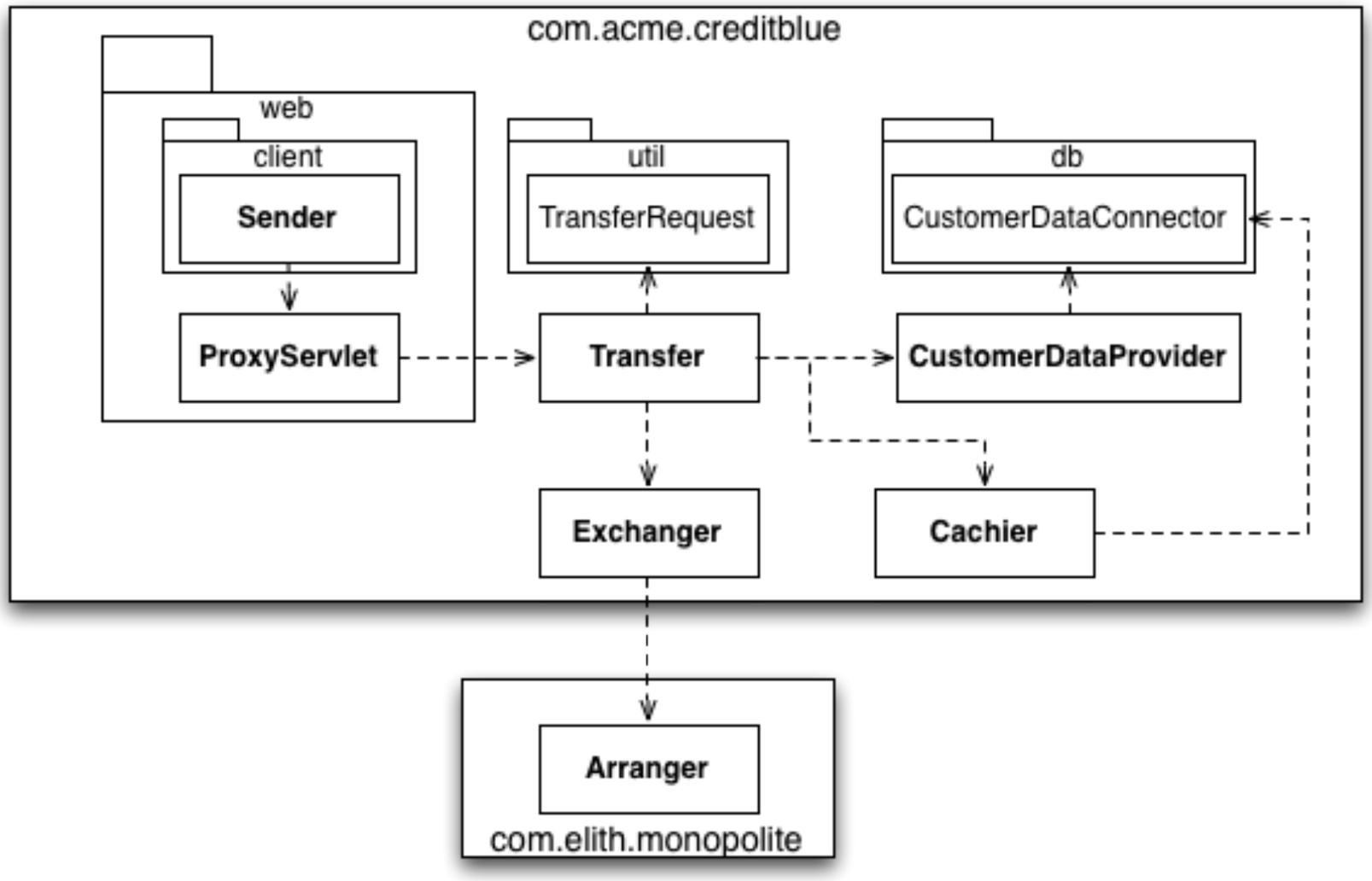
Brillien hálózati architektúra



Egy kisebb példa



Egy kisebb példa



Egy kisebb példa

- Minden Presence egy XMPP entitás
- JSON-alapú szerializáció
- Kommunikációs metódusok:
    - sendGet
    - sendSet
    - sendAGet
    - sendDelegatedGet
- “Sorrendi és név szerinti kötés”
- Példányosítási módok: Sparkle, Consonant, Resident

Egy kisebb példa

```
public class TransferRequest implements Validable {

    private String      senderName;

    private String      senderAccountNumber;

    private String      recipientName;

    private String      recipientAccountNumber;

    private long        amount;

    private String      currency;

   // set-get methods

}
```

Egy kisebb példa

```
@PresenceService( logLevel = PresenceService.INFO, messageExpiration = 2000 )

@Indispensable

@Sparkle

public class Transfer extends SuperFlow {

    @Activate( timeout = 8000 )

    public String startTransaction( @P(name="req") TransferRequest req )

       throws BrillienException {

           ...

    }
```

Egy kisebb példa

```
@PresenceService( logLevel = PresenceService.FINE )

@Consonant

public class Cachier extends SuperPresence {

    public String deposit( @P(name="owner") String owner,

        @P(name="accountNumber") String accountNumber, @P(name="amount") long amount,

        @P(name="currency") String currency ){

            // do something ...

            return "done";

    }

}
```

Tapasztalatok

- Nagy mértékben egyszerüsödő tervezési és fejlesztési folyamat
- Időbeli távolságok hatékony kezelése
- Gyors bevezetés és integráció
- Tanulás nehézkes a sík modellhez szokott személyeknek
- Nagyobb absztrakció, szabadság -> gondosabb tervezés
- Kód mentes a nehezékektől: csak üzleti logika
- Tervezési minták, ösvények kiforratlanok
- Kevés tutorial
- Unit tesztek: interfész teszt szükséges