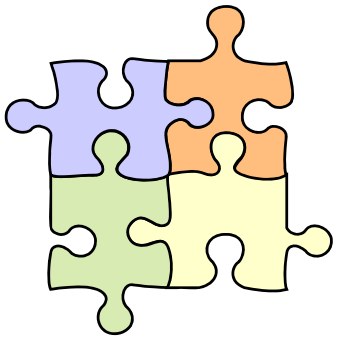
Luca Cabibbo

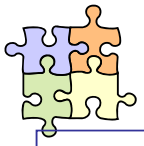


Architetture Software

Punto di vista della Concorrenza

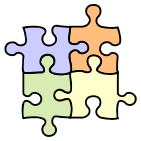
Dispensa AS 18

ottobre 2008



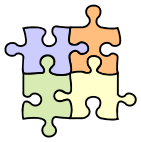
- Fonti

- [SSA] Chapter 18, The Concurrency Viewpoint



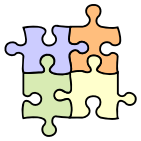
- Obiettivi e argomenti

- Obiettivi
  - descrivere il punto di vista della Concorrenza
- Argomenti
  - punto di vista della concorrenza
  - interessi
  - modelli
  - problemi e insidie



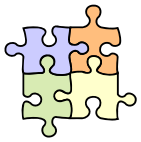
* Punto di vista della Concorrenza

- Il **punto di vista della Concorrenza** descrive la struttura della concorrenza del sistema, definendo le corrispondenze tra elementi funzionali e unità di concorrenza, per identificare in modo chiaro le parti del sistema che possono essere eseguite in modo concorrente, mostrando come ciò viene coordinato e controllato
  - interessi
    - struttura dei task; corrispondenze tra elementi funzionali e task; comunicazione interprocesso; gestione dello stato; sincronizzazione e integrità; startup e shutdown; fallimento dei task; rientranza
  - modelli
    - modelli della concorrenza; modelli a stati
  - applicabilità
    - sistemi che richiedono diversi processi/thread di esecuzione



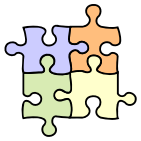
Vista della concorrenza

- La *vista della concorrenza*
  - descrive la struttura della concorrenza, ed i relativi vincoli, mostrando come i diversi processi/thread concorrenti comunicano e coordinano in modo effettivo la loro esecuzione
- Rilevanza
  - la concorrenza è importante soprattutto nei sistemi distribuiti e guidati dagli eventi
    - come gestire la concorrenza? come gestire la concorrenza con il middleware usato?
  - la concorrenza può essere importante per consentire prestazioni ed affidabilità
  - tuttavia, alcuni sistemi hanno poca concorrenza – in altri, è gestita da componenti già esistenti – ad es., un DBMS



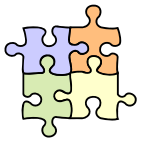
* Interessi

- Interessi affrontati dalla vista della concorrenza
  - struttura dei task
  - corrispondenze tra elementi funzionali e task
  - comunicazione interprocesso
  - gestione dello stato
  - sincronizzazione e integrità
  - startup e shutdown
  - fallimento dei task
  - rientranza



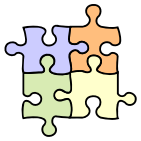
Preliminari

- Terminologia
  - *processo* – un processo del sistema operativo
  - *thread*
  - *task* – una generica unità di concorrenza – processo o thread



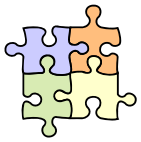
Struttura dei task

- *Struttura dei task*
  - l'aspetto più importante nella creazione di questa vista è stabilire la struttura dei task del sistema
    - quali task? come comunicano? come si coordinano?
  - va stabilita una strategia complessiva di gestione della concorrenza
    - per decidere come partizionare il carico
    - per decidere come distribuire le responsabilità



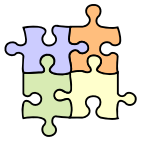
Corrispondenze tra elementi funzionali e task

- *Corrispondenze tra elementi funzionali e task*
  - il modo con cui gli elementi funzionali sono associati ai task può avere un impatto significativo su diverse qualità
    - prestazioni, efficienza, affidabilità, capacità di recupero, sicurezza, ...
  - ad esempio
    - elementi funzionali che devono cooperare strettamente potrebbero essere mappati su un singolo processo – prestazioni
    - elementi funzionali che devono essere isolati vanno mappati su processi diversi – sicurezza
    - possibile avere più task per un singolo elemento funzionale – prestazioni, affidabilità



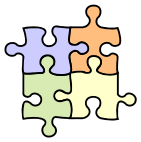
Comunicazione interprocesso

- *Comunicazione interprocesso*
  - la comunicazione e la condivisione di strutture di dati è facile ed efficiente nell'ambito di uno stesso processo
  - la comunicazione e la sincronizzazione tra processi in uno stesso calcolatore è più complessa e costosa
  - la comunicazione tra processi in calcolatori diversi è ancora più costosa
  - utile identificare i meccanismi di comunicazione interprocesso (e gli strumenti di middleware che li implementano) più adeguati al sistema in discussione
    - ad es., socket, RPC, RMI, messaging, memoria condivisa, pipe, code, ORB, ...
    - ciascuno ha i propri punti di forza e le proprie debolezze



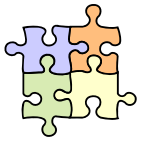
Gestione dello stato

- *Gestione dello stato*
  - per *stato* qui si intende lo “stato funzionale” in cui si può trovare un elemento funzionale
    - ad es., waiting-for-query, executing-query, ...
    - descritto solitamente da un automa a stati finiti, con gli stati validi, le transizioni valide tra stati, nonché le relazioni causa-effetto delle transazioni tra stati
  - più opportuno affrontare questo interesse nella vista della concorrenza
    - e non nella vista funzionale



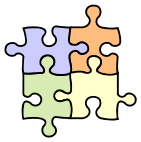
Sincronizzazione ed integrità

- *Sincronizzazione ed integrità*
  - l’esecuzione concorrente di operazioni non deve portare alla corruzione delle informazioni gestite dal sistema
    - anomalie possibili – perdita di aggiornamento, lettura sporca, letture inconsistenti, aggiornamento fantasma, inserimento fantasma, ...
  - si devono perseguire alcune qualità desiderabili
    - safety (non succede niente di male), liveness (se possibile, succede qualcosa di buono), prestazioni (succede qualcosa di buono)
  - diversi meccanismi per la gestione della concorrenza
    - da quelli di basso livello dei linguaggi di programmazione
    - a quelli di livello più alto offerti da strumenti di middleware dedicati – ad es., un TP monitor, un application server



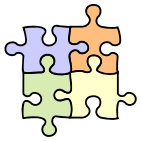
Startup e shutdown

- *Startup e shutdown*
  - un insieme complesso di task può richiedere
    - l'attivazione dei task in un ordine specifico
    - la disattivazione dei task in un ordine specifico – ad es., per evitare perdita dai dati
    - di dover capire come riattivare un certo task – ad es., riattivando prima altri task, in un certo ordine



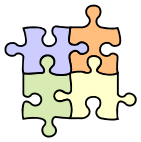
Fallimento dei task

- *Fallimento dei task*
  - è possibile che un task non sia disponibile quando è richiesto da un altro task
    - ad es., perché fallito, o perché fallita la comunicazione
  - come garantire che il fallimento (anche temporaneo) di un task non provochi il fallimento dell'intero sistema?



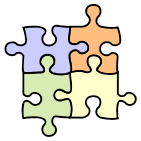
Rientranza

- *Rientranza*
  - un elemento software è *rientrante* se è in grado di operare correttamente se usato in modo concorrente da più task
  - può essere importante analizzare la rientranza a livello architetturale
    - ad es., per ogni elemento che può essere acceduto in modo concorrente e che deve gestire uno stato (delle informazioni), bisogna capire se lo stato (o una sua parte) va associato ai singoli accessi, alle sessioni individuali, oppure all'intero elemento (quindi va condiviso tra i diversi utenti)



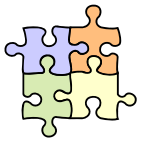
* Modelli

- Tipi di modelli per la vista della concorrenza
  - modelli della concorrenza
  - modelli a stati
    - ad es., modello di macchina a stati di UML, reti di Petri, ...



- Modelli della concorrenza

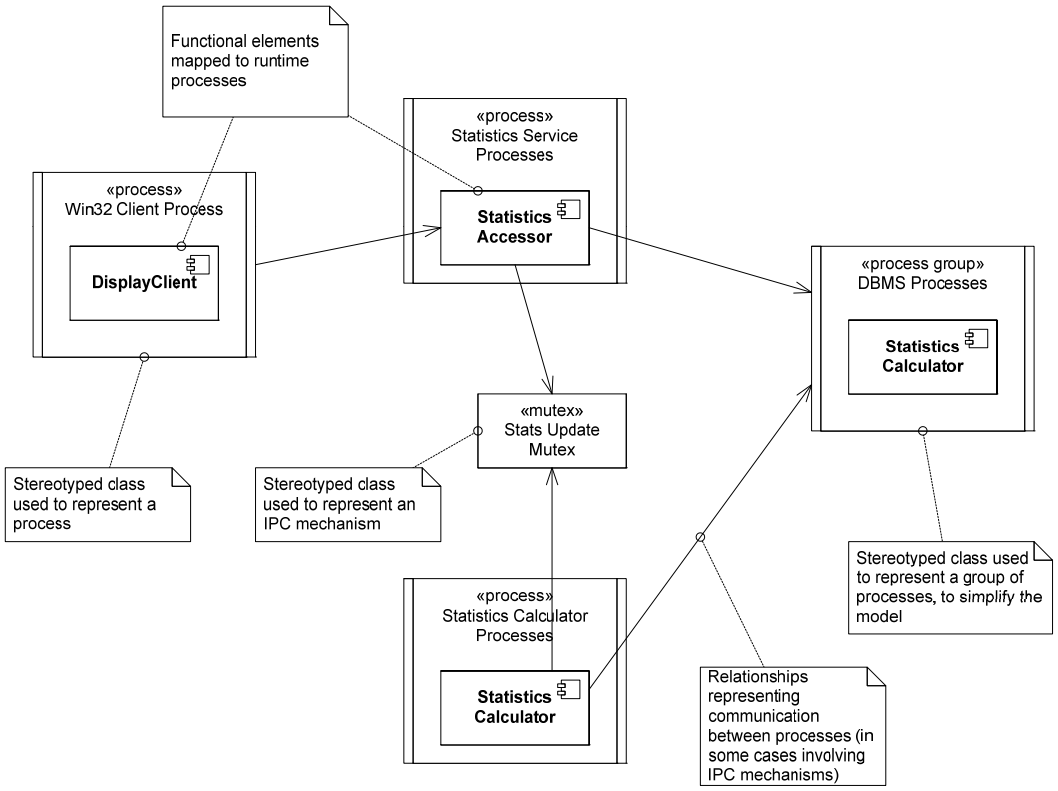
- I modelli per descrivere la struttura della concorrenza normalmente comprendono i seguenti elementi
  - *processi* – intesi come processi del sistema operativo
  - *gruppi di processi*
    - se possono essere collettivamente considerati una singola entità a livello di sistema
    - ad es., un'istanza di un DBMS
  - *thread*
    - talvolta possono essere ignorati a livello architetturale
  - *comunicazione interprocesso*
    - meccanismi di chiamata di procedure – ad es., RPC ed RMI
    - meccanismi di coordinamento dell'esecuzione
    - meccanismi di condivisione dei dati



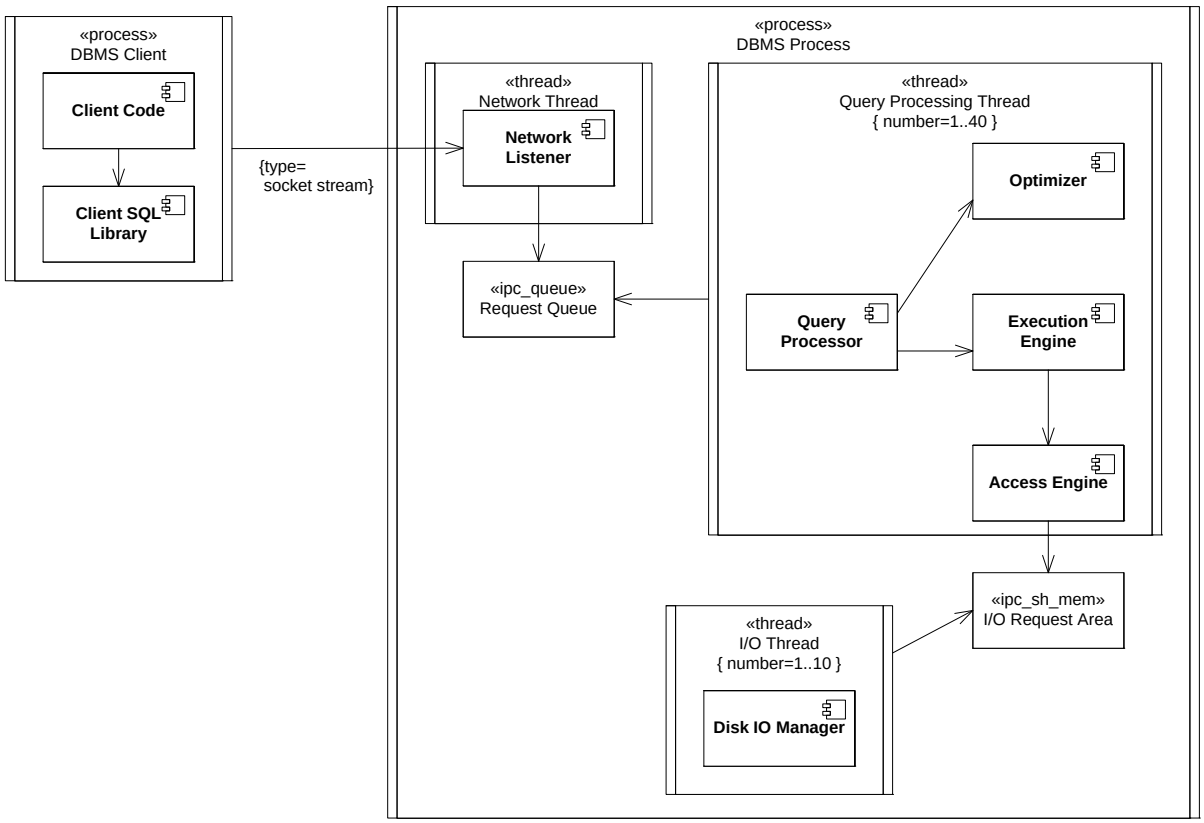
Notazioni

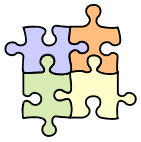
- Possibili diverse notazioni per i modelli della concorrenza
  - UML
    - non un tipo di diagramma o elemento esplicito
    - possibile usare classi e oggetti semplici oppure *attivi* – con stereotipi – che contengono *componenti* – che rappresentano elementi funzionali
  - notazioni formali
    - ad es., CSP (Communicating Sequential Processes), CCS (Calculus for Communicating Systems)
    - spesso notazioni astratte e matematiche – utili, ad es., per l'analisi dei deadlock
    - poco utili per la comunicazione con le parti interessate
  - notazioni informali
    - attenzione a "definirne" la semantica

Esempio

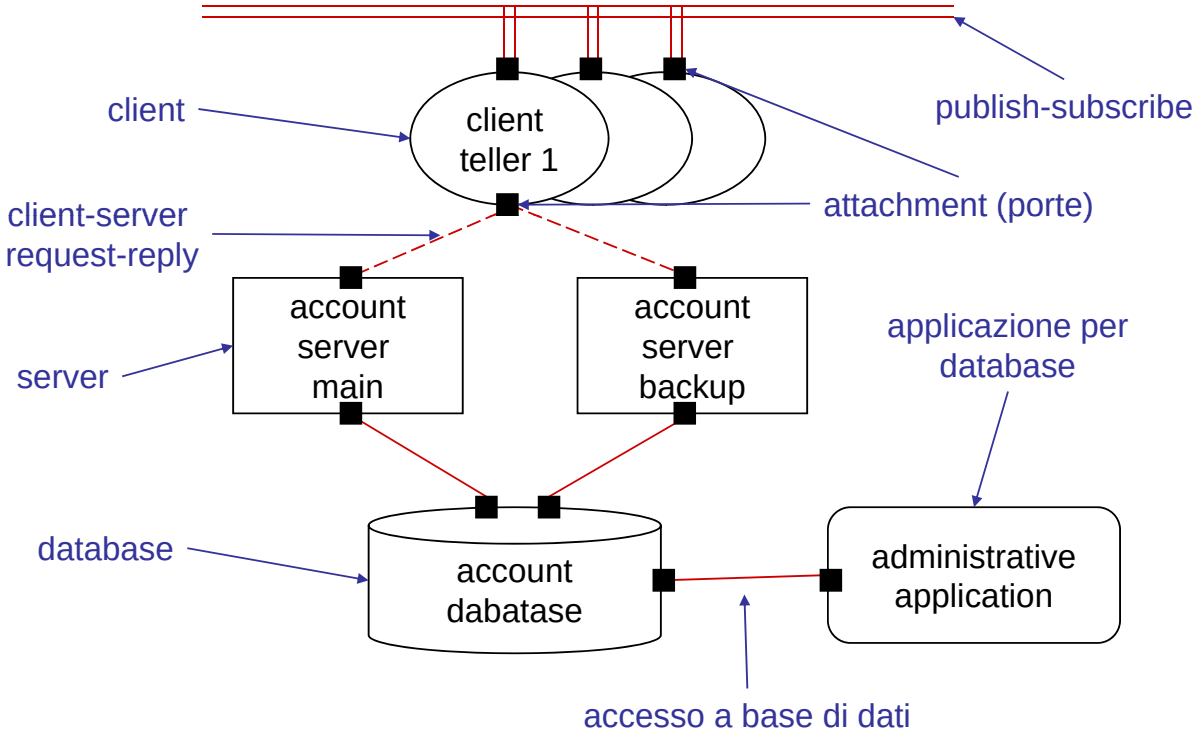


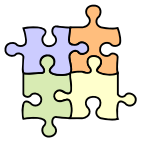
Esempio





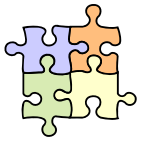
Esempio – notazione informale





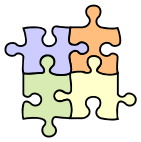
Attività (1)

- Definisci le corrispondenze tra elementi funzionali e task
  - nei casi più semplici, una corrispondenza 1:1
  - nei casi più complessi, ci sono corrispondenze N:M
    - elemento partizionato tra processi
    - elementi in esecuzione in più processi
  - la concorrenza va introdotta solo dove necessario
    - la concorrenza introduce un overhead (di complessità, quantomeno) per la sua gestione
    - questo va pagato solo per ottenere altre qualità



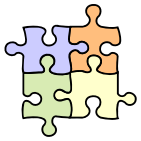
Attività (2)

- Determina il progetto dei thread
  - solo se è architetturalmente significativo
    - ad es., per avere un approccio uniforme in tutte le parti del sistema
- Considera la condivisione di risorse
  - risorse condivise vanno protette dalla corruzione – ad es., mediante meccanismi di locking
  - questo aspetto è di interesse anche per la vista delle informazioni, e va affrontato come una singola attività



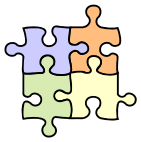
Attività (3)

- Assegna priorità a thread e processi
- Analizza i deadlock
- Analizza la contention (accesso concorrente a risorse condivise)
  - se eccessiva, le prestazioni possono degradare in modo significativo
    - garantisci che la struttura di concorrenza scelta consenta di gestire il carico atteso



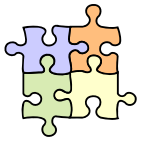
- Modelli a stati

- I modelli per descrivere lo stato degli elementi runtime normalmente comprendono i seguenti elementi
  - *stato*
    - una condizione identificabile e stabile in cui può trovarsi un elemento – ad es., in attesa di una risposta
  - *transizione*
    - un possibile cambiamento di stato, causato da un evento
  - *evento*
    - la notifica che è avvenuto qualcosa di interessante
  - *azione*
    - un'elaborazione atomica (non interrompibile) che può essere associata ad una transizione

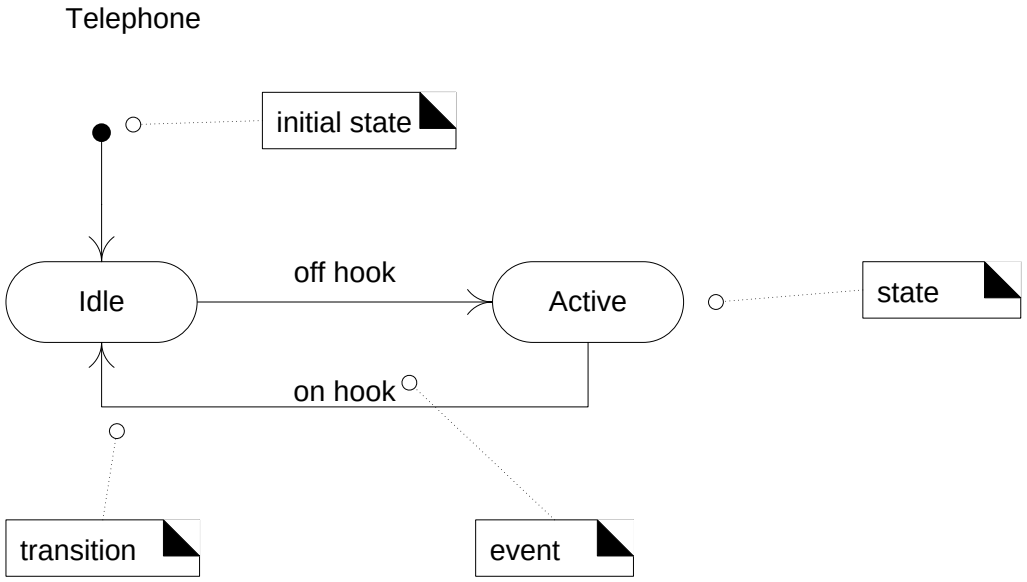


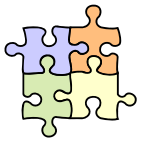
Notazioni

- Possibili diverse notazioni
  - automi
  - reti di Petri
  - diagrammi di macchina a stati di UML

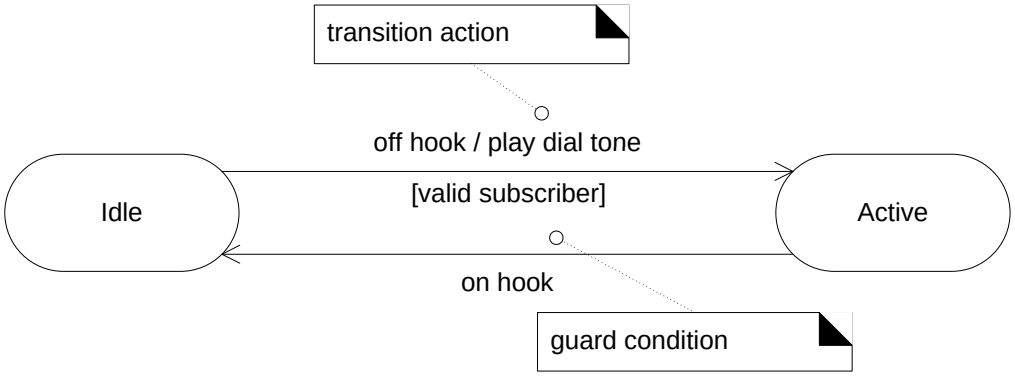


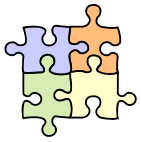
Esempio



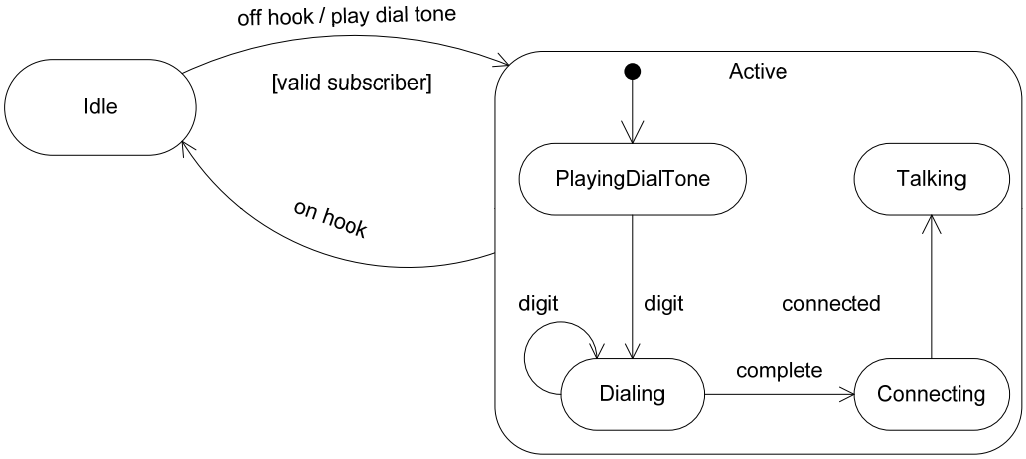


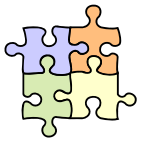
Azioni legate a transizioni e guardie





Stati annidati





* Problemi e insidie

- Modellare la concorrenza sbagliata
  - concentrati su ciò che è architetturalmente significativo
- Complessità eccessiva
  - una concorrenza troppo complessa può essere difficile da progettare, gestire e analizzare
  - ogni aspetto introdotto deve essere giustificato opportunamente
- Problemi da analizzare ed evitare
  - contention di risorse – hot spot
  - deadlock
  - condizioni di race