Azzolini Riccardo 2019-12-03

Altri diagrammi UML

1 Timing diagram

I **timing diagram** permettono la rappresentazione esplicita del tempo. Essi sono utili per mostrare:

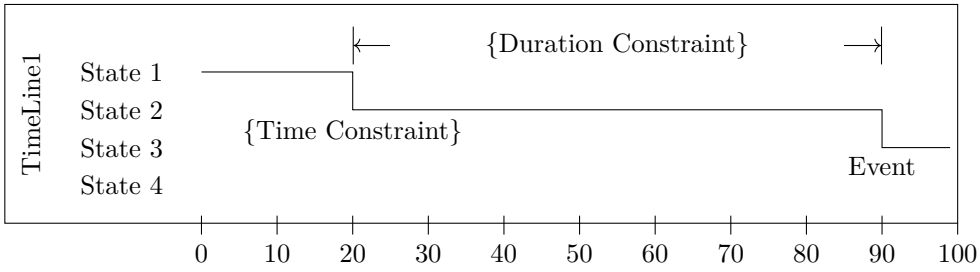
- come lo stato/valore di un elemento cambia nel tempo;
- l'interazione tra eventi temporizzati;
- i vincoli di tempo e di durata tra eventi temporizzati.

Questi diagrammi possono quindi essere usati per esprimere, ad esempio, i vincoli non funzionali relativi alle prestazioni.

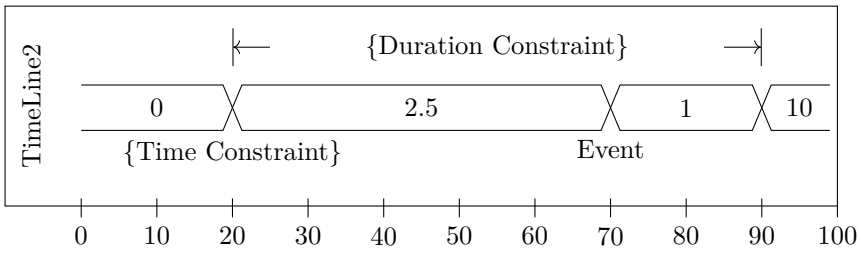
1.1 Lifeline

Una lifeline si disegna in modo diverso a seconda del tipo di stato dell'entità rappresentata:

- valori discreti:



- valori continui:



In corrispondenza di una transizione (cambiamento di stato) è possibile indicare

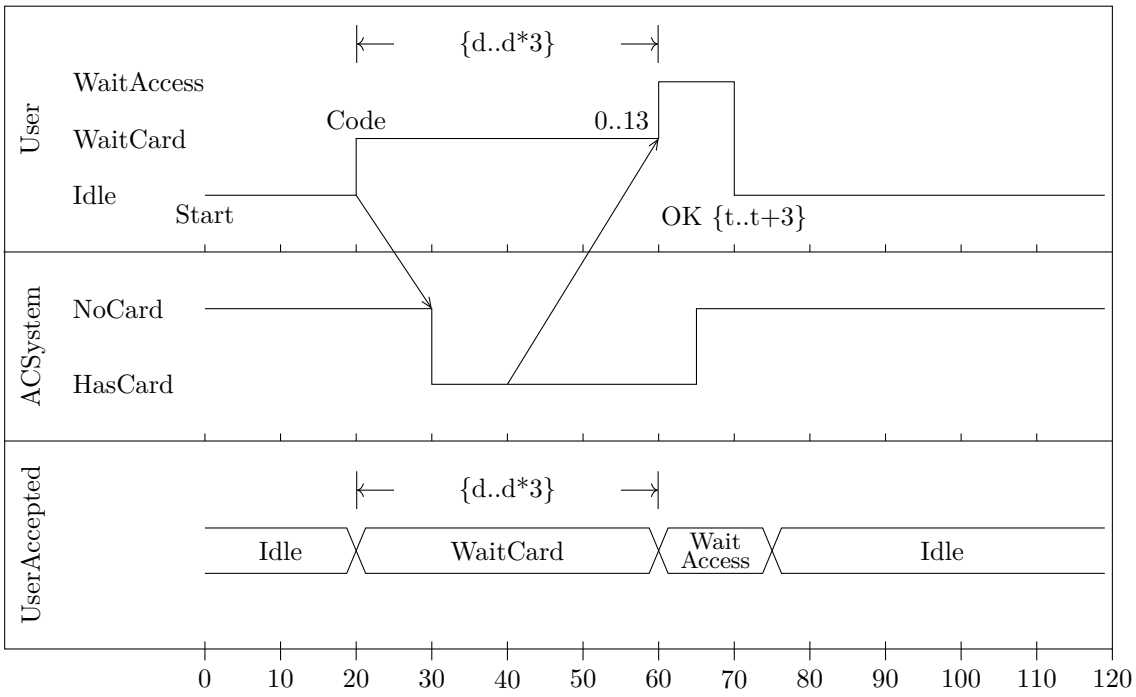
- eventi
- vincoli di tempo
- vincoli di durata

che provocano tale transizione.

1.2 Lifeline multiple

Un diagramma può contenere anche due o più lifeline, che:

- devono avere tutte lo stesso asse x;
- possono scambiarsi dei messaggi (indicati tramite delle frecce), e questi possono provocare delle transizioni.



2 Component diagram

Un **componente** rappresenta una parte modulare di un sistema:

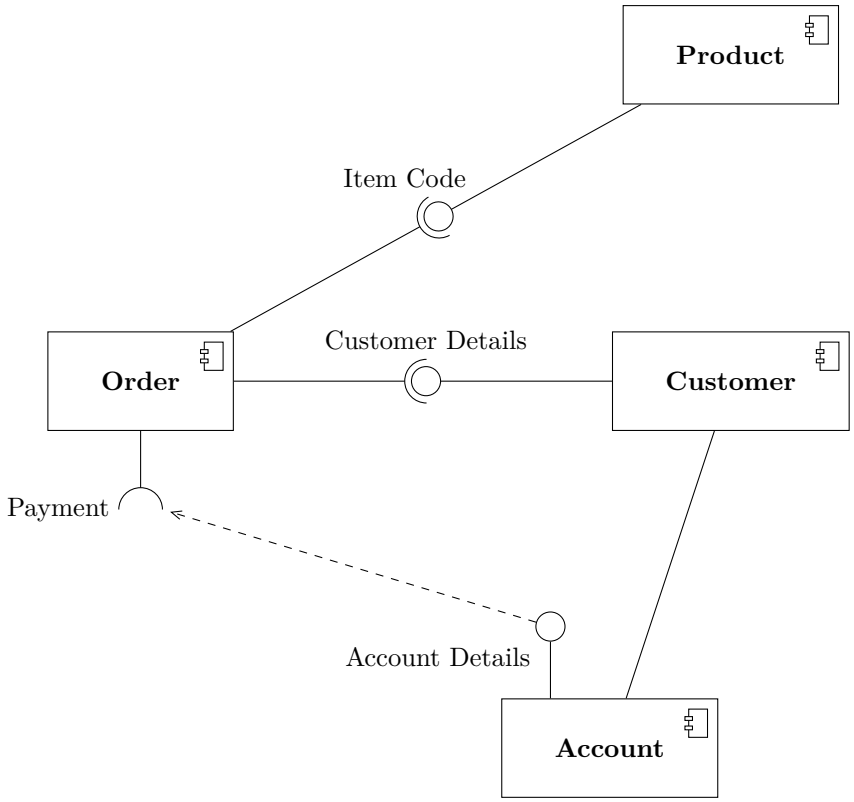
- incapsula i propri contenuti;

- il suo comportamento è definito in termini di interfacce fornite e richieste;
- di solito, è implementato da una o più classi/oggetti, ma può comprendere anche documentazione, ecc.;
- può avere più implementazioni, che sono intercambiabili: una qualsiasi di esse può essere immessa nell'ambiente di esecuzione.

Le funzionalità di un sistema si possono costruire assemblando componenti, cioè collegando le interfacce fornite e richieste.

Un **component diagram** descrive i componenti che costituiscono concretamente il sistema finale (mentre i class diagram, ecc., sono diagrammi concettuali).

2.1 Esempio

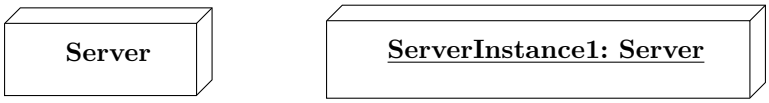


3 Deployment diagram

I **deployment diagram** specificano l'architettura del sistema, descrivendo l'assegnamento dei semilavorati software (*artifact*) ai nodi hardware.

3.1 Nodi

I **nodi** rappresentano dispositivi hardware o ambienti di esecuzione software.



Essi possono essere descritti come annidati, e possono essere connessi da canali di comunicazione, che consentono di modellare sistemi a rete.

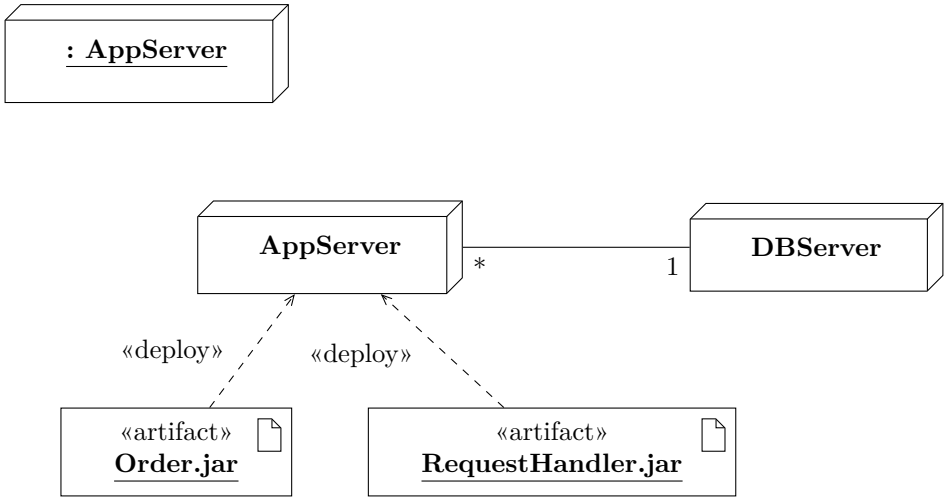
3.2 Artifact

Un artifact è la specifica di un'informazione "fisica" usata o prodotta da un processo di sviluppo, o nell'installazione e uso di un sistema.

Alcuni esempi di tipi di artifact sono:

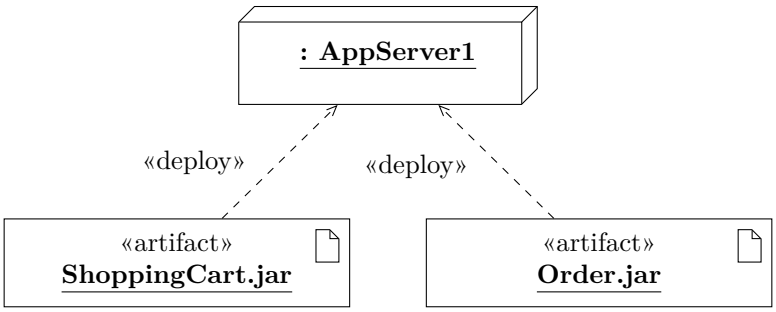
- file di modelli;
- file sorgenti;
- script;
- eseguibili binari;
- tabelle di un database;
- documenti tecnici;
- messaggi di email;
- ecc.

3.3 Esempio

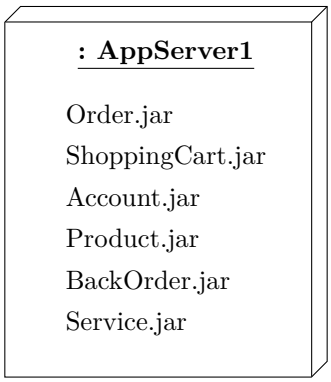


3.4 Notazione alternativa

Invece di rappresentarli separatamente,



gli artifact assegnati a un nodo possono essere elencati all'interno del nodo stesso:



3.5 Esempio di rete