Ingegneria del Software (6 crediti) a.a. 2011-12
Prova Scritta del 6 febbraio 2012 – Bozza di soluzione

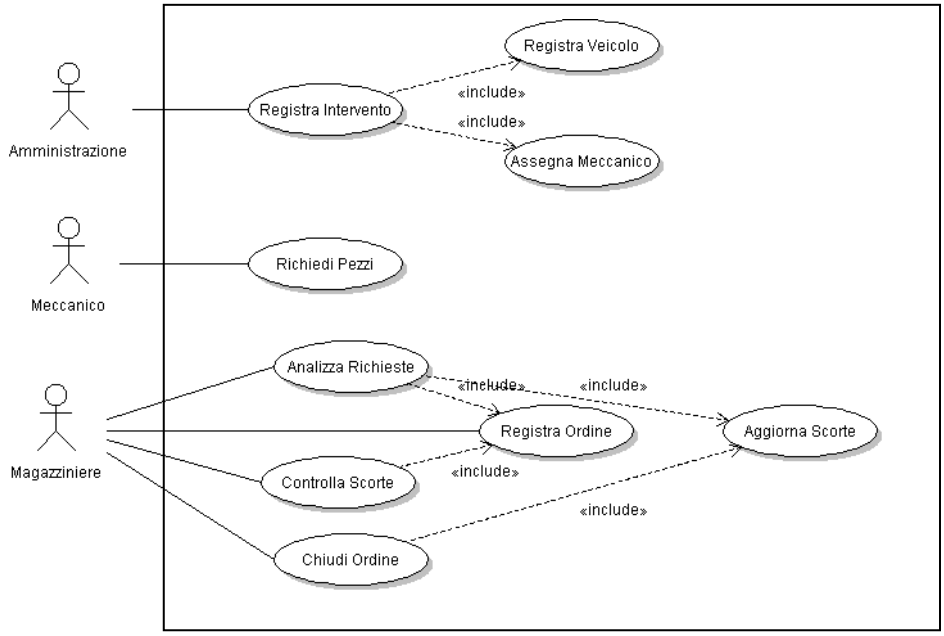
Esercizio 1

NOTA: Si chiedeva di modellare il sistema da realizzare ed i suoi casi d'uso, non come le cose vengono gestite nel dominio applicativo prima dell'introduzione del sistema sw (cioe' il meccanico chiede al magazziniere, il magazziniere –attore secondario!- gli da' i pezzi, …)

a) Identificare gli attori del sistema descritto, distinguendo in attori primari e secondari.

Attori primari: Meccanico, Magazziniere, Amministrazione

b) Identificare i casi d'uso del sistema e produrre un diagramma dei casi d'uso.



c) Specificare il caso d'uso "Richiesta pezzi", che prevede l'inserimento da parte del meccanico responsabile di un intervento dell'elenco dei pezzi di ricambio relativi alla realizzazione di tale intervento.

Use Case: Richiesta pezzi
Level: User-Goal
Intention in Context: Il maccanico richiede dei pezzi di ricambio per effettuare la riparazione
Primary Actor: Meccanico
Main Success Scenario:
1. Il Sistema chiede al Meccanico di scegliere un intervento tra quelli a lui attribuiti dall'amministrazione
2. Il Meccanico sceglie un intervento (il quale è associato al veicolo)
3. Il Sistema presenta una lista di pezzi di ricambio presenti in officina dividendoli per categoria
4. Il Meccanico sceglie una o più categorie e i pezzi che gli interessano

Lo use case finisce con successo.
Extension:
2a. Il Meccanico esce dalla richiesta pezzi
    2a.1 Il Sistema visualizza il menu principale e aspetta
4a. Il Meccanico esce dalla richiesta pezzi
    4a.1 Il Sistema visualizza il menu principale e aspetta

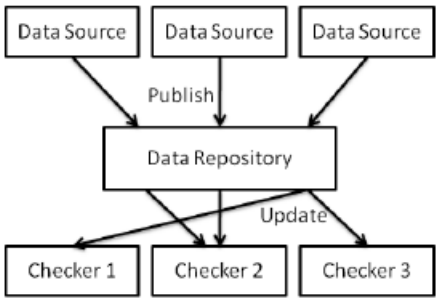
Esercizio 2

Monitoraggio di un aeroplano

NOTA: Lo stile architetturale e' uno stile di controllo. Nella soluzione proposta si utilizza uno stile a eventi publish-subscribe, con l'idea che i sensori inviino stream di dati che sono poi monitorati via software. In questo modo si riescono a ottenere reazioni in quasi real-time, come richiesto. Erano accettabili anche soluzioni Interrupt driven ma in questo caso andava chiarito bene che cosa inviavano i sensori, che cosa veniva monitorato via hardware e che cosa via software, come venivano generate interrupt, … o anche con controllo centralizzato (real-time system control).

a) Stile architetturale: Event-based

b) Architettura del sistema



Il sistema consiste in un data repository, che memorizza i valori correnti di tutti i dati dei sensori. I diversi controllori possono registrarsi (subscribe) chiedendo di essere avvisati in caso di cambiamenti in diversi tipi di dati. Quando le sorgenti dati generano nuovi dati, il data repository informa i controllori registrati che sono disponibili nuovi dati. I controllori sono moduli individuali che possono effettuare i loro controlli in parallelo mano a mano che nuovi dati diventano disponibili.

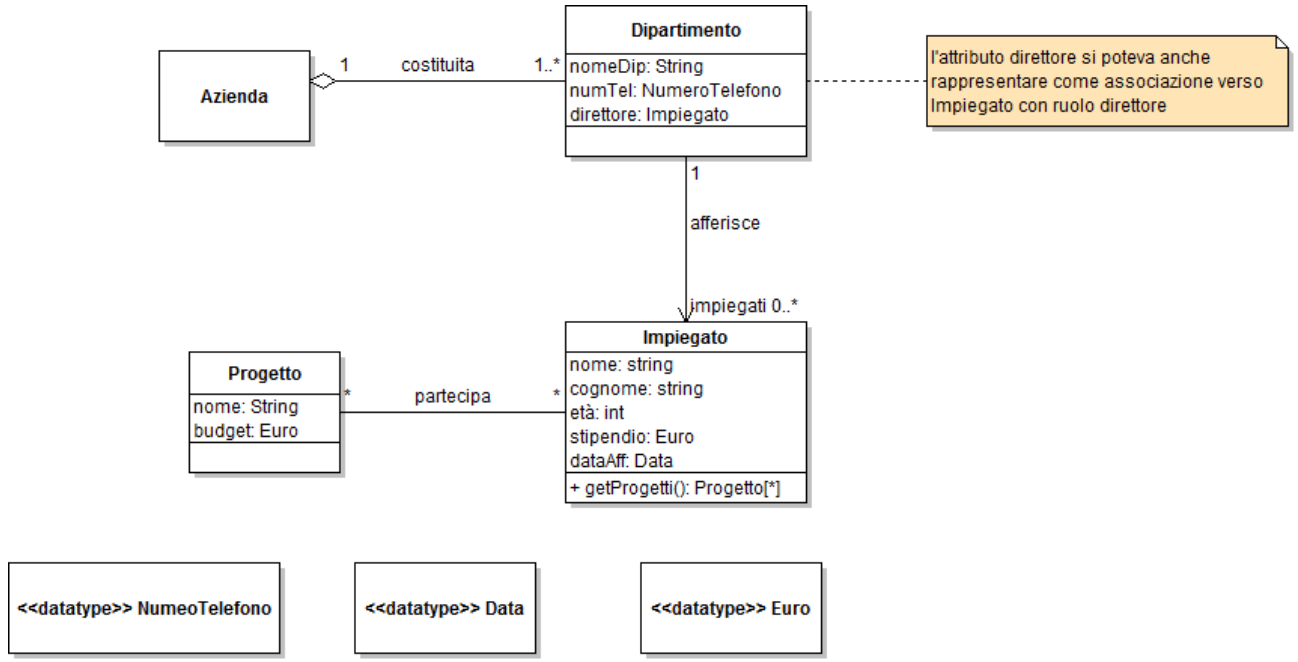
c) Principali punti di forza (vantaggi) e di debolezza (svantaggi) dell'uso di tale architettura:

+ tempo di risposta: i controlli vengono effettuati non appena i nuovi dati sono disponibili
+ efficienza: i controlli possono essere effettuati non in parallelo
+ estendibilità: nuovi controlli possono essere aggiunti senza bisogno di effettuare modifiche nel sistema

- integrità dei controlli: i controlli sono fatti indipendentemente uno dall'altro, quindi potrebbero causare azioni "in contraddizione/in conflitto" come risultato del controllo
- affidabilità: il data repository è single point of failure.

Esercizio 3

a)



b)

```
public class Dipartimento {
        private String nome;
        private NumeroTelefono numTel;
        private LinkedList<Impiegato> impiegati;
        private Impiegato direttore;
}
```

La data di afferenza che è vista a livello di modellazione come un classe associativa può essere implementata come attributo nella classe Impiegato

c)

```
public List<Progetto> elencaProgetti() {
        List<Progetto> ris = new List<Progetto>()
        foreach (Impiegato i in impiegati)
           foreach (Progetto p in i.getProgetti())
               if(! ris.contains(p)) {
                     ris.add(p)
                     p.printNome()
               }
}
```

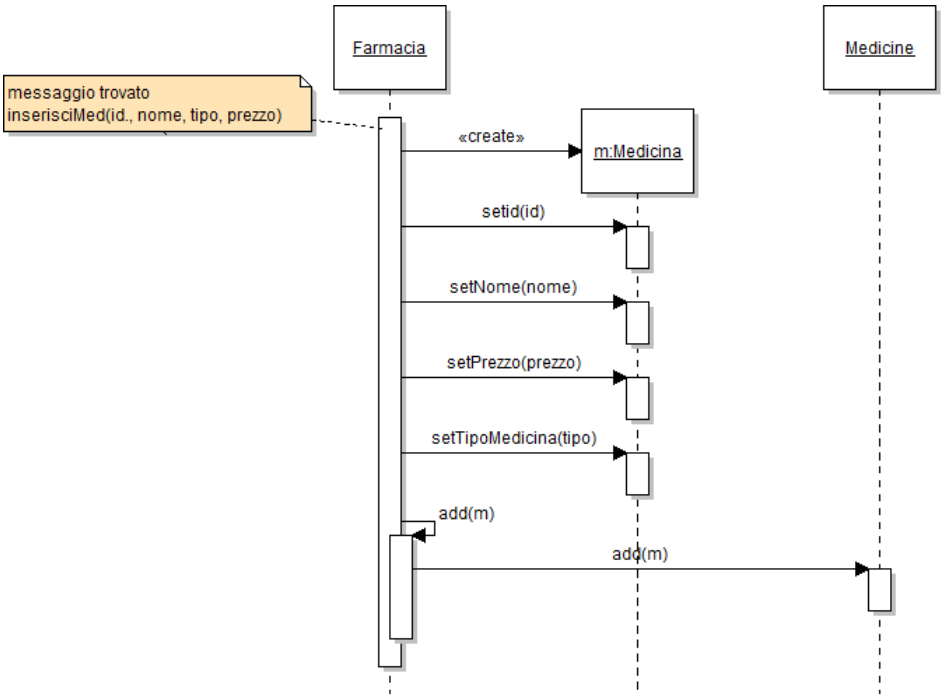
d)

AZIENDA(idA, …)
DIPARTIMENTO(idD,nomeDip,numTel,idA^{AZIENDA},direttore^{IMPIEGATO})
IMPIEGATO(idI,nome,cognome,eta,stipendio,dataAff,dip^{DIPARTIMENTO})
PROGETTO(idP,nome,budget)
PARTECIPA(idP^{PPROGETTO},idI^{IMPIEGATO})

e)

```
Context Impiegato
Inv: (self.stipendio<1000) implies (self.progetti-->size()<=2)
```

Esercizio 4



c) JUnit 3.x

```
import junit.framework.TestCase;
public class testContaMedicine extends TestCase {
	Farmacia f;

	public void setUp() {
		f = new Farmacia();
	}

	public void testContaVuoto() {
		assertEquals("0 0 0", f.contaMedicinePerTipo());
	}

	public void testContaA() {
		f.inserisciMedicina(1, "aspirina", TipoMedicina.A, 3);
		assertEquals("1 0 0", f.contaMedicinePerTipo());
	}

	public void testContaB() {
		f.inserisciMedicina(1, "aspirina", TipoMedicina.B, 3);
		assertEquals("0 1 0", f.contaMedicinePerTipo());
	}

	public void testContaC() {
		f.inserisciMedicina(1, "aspirina", TipoMedicina.C, 3);
		assertEquals("0 0 1", f.contaMedicinePerTipo());
	}

	public void testConta2A() {
		f.inserisciMedicina(1, "aspirina", TipoMedicina.A, 3);
		f.inserisciMedicina(2, "aspirina", TipoMedicina.A, 3);
		f.inserisciMedicina(3, "malox", TipoMedicina.B, 3);
		assertEquals("2 1 0", f.contaMedicinePerTipo());
	}

	public void testConta2AstessoID() {
		f.inserisciMedicina(1, "aspirina", TipoMedicina.A, 3);
		f.inserisciMedicina(1, "aspirina C", TipoMedicina.C, 3);
		f.inserisciMedicina(3, "malox", TipoMedicina.B, 3);
		assertEquals("1 1 0", f.contaMedicinePerTipo());
	}}
```