UNIVERSITA' DEGLI STUDI DI PALERMO

FACOLTA' DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

A.A. 2002-2003

Sistema multiagente per la gestione di un'azienda di biciclette

Tesina di Ingegneria del Software di:

Caico Roberto

Gargagliano Giuseppe

Termine Francesco

Indice ridotto dei contenuti

Indice dei contenuti

PROJECT MANAGEMENT

Sistema BikesAgent

Redatto da:	Caico Roberto Gargagliano Giuseppe Termine Francesco	_____	_____
Emesso da:	Caico Roberto Gargagliano Giuseppe Termine Francesco	_____	_____
Approvato da:	Caico Roberto Gargagliano Giuseppe Termine Francesco	_____	_____

Tipo documento:	SPMP *Commessa*	.01 *Num. Doc.*	.01 *Vers.*
Prima emissione:	12/04/2004		
Distribuzione:	Interna		
Data di stampa:	18/04/04 18.25		

Path del file: *C:\documenti\tesina*

La validità di questa informazione è garantita al momento della stampa del documento.

Introduzione

Il sistema BikeAgents ha come scopo la gestione del processo produttivo di un'azienda che produce due differenti modelli di biciclette, un modello da corsa ed uno da trekking

Il documento ha come obiettivo quello di identificare con chiarezza le attività che caratterizzano la commessa al fine di avere un quadro chiaro sullo sviluppo della stessa e inoltre dà la possibilità di ottimizzare i tempi di produzione. Si avrà dunque una iniziale schedulazione delle attività che caratterizzeranno il processo di realizzazione della commessa. Si passerà successivamente a una più precisa e puntuale schematizzazione delle attività incastrandole al fine di poterne ottimizzare i tempi e facilitarne lo sviluppo stesso. Infatti una chiara e corretta schedulazione delle attività dà la possibilità non solo di ottimizzare i tempi ma di avere una corretta schematizzazione temporale delle attività.

Schedulazione del Progetto

Schedulazione del progetto		
Data	**Fase di progetto**	**Attività**
21/07/2003→04/08/2003	Raccolta requisiti utente per la progettazione della Commessa	
05/08/2003→03/09/2003	Attività di studio di: 1. Microsoft Project 2. Studio Swing - Java 2D – AWT 3. Studio Jdbc – Odbc Driver 4. Studio Sun One Studio 5 5. Studio Microsoft Access 6. Studio Sql Language 7. Studio Jade 8. Studio XML Language 9. Studio RDF 10. Studio Rational Rose 11. Specifiche FIPA 12. Studio Passi 13. Studio PTK 14. Studio Agent Factory	
03/09/2003		Discussione con il team di progetto per eventuali chiarimenti e per discussione di esempi e materiale studiato e raccolto.
04/09/2003→12/09/2003	Analisi Progettazione e Struttura dati: 1. Scelta Struttura Dati 2. Costruzione Tabelle e relazioni 3. Disegno Schema Concettuale - ERD	
12/09/2003		Presentazione del progetto al cliente
15/09/2003→20/10/2003	Progettazione software divisa nelle seguenti fasi: • System Requirement Model • Agent Society Model	

	• Agent Implementation Model • Progettazione DataBase	
22/09/2003		Incontro col committente
08/10/2003		Incontro col committente
16/10/2003		Riunione con il team di progetto
21/10/2003→22/12/2003	Implementazione	
21/10/2003		Presentazione prima versione al cliente
23/12/2003→16/01/2004	Fase di testing suddivisa in: • Test e Correzioni	
19/01/2004→19/03/2004	Sospensione delle Attività	
22/03/2004→09/04/2004	Stampa RDF e consegna	

Evoluzione del piano di sviluppo del progetto

Per poter pianificare correttamente ed effettuare in modo semplice cambiamenti nella pianificazione, per la gestione ed allocazione delle risorse necessarie alla realizzazione del prodotto e per la gestione delle previsioni si è utilizzato l'applicativo Microsoft Project 2003 Professional.

Il progetto utilizza PASSI, una metodologia di progettazione orientata agli agenti ed UML per lo sviluppo del software.

Il prodotto finale e le sue versioni intermedie sono state testate su PC che utilizzano sistema operativo Windows XP Home Edition e Professional Edition.

Materiale di riferimento

Per la realizzazione del sistema sono stati utilizzati i seguenti materiali di riferimento:

- Bruegge-Dutoit :Object-Oriented Software Engineering: Conquering Complex and Changing System (Corso di Ingegneria del Software)
- Tutto il materiale di esempio trovato sul sito di Ingegneria del Software del professore M. Cossentino. Tale materiale può essere recuperato collegandosi al sito internet http://www.csai.unipa.it/cossentino/se02_03/ e sfruttando i collegamenti ivi presenti
- Documentazione JADE 3.0b1 scaricabile dal sito http://jade.cselt.it/
- Deitel & Deitel: Java Fondamenti di programmazione (Manuale di java)
- Deitel & Deitel: Java Tecniche avanzate di programmazione (Manuale di java)
- Documentazione del J2SDK1.4.0
- Guida in linea di Rational Rose Enterprise Edition
- Guida in linea di Microsoft Project Professional 2003
- Guida in linea di Microsoft Access 2003
- Guida in linea di Microsoft Visio Professional 2003

Organizzazione del progetto

Schema del processo

Il progetto è iniziato il 21 Luglio 2003 ed è terminato il 09 Aprile 2004 con al sospensione dal 19 Gennaio 2004 al 19 Aprile 2004. Presentiamo le principali attività svolte durante il periodo di lavoro che terminano (in genere) con la creazione di versioni intermedie del prodotto. Tali versioni intermedie (eccetto il prodotto finale stesso che si reputa essere completo e funzionante) servono per mostrare il sistema al cliente. Tali attività sono di seguito riportate.

- Revisione di progetto dopo la raccolta dei requisiti utenti per la stesura e la gestione del sistema BikeAgent: 3 settembre 2003;
- presentazione del progetto di massima e sua approvazione da parte del cliente: 12 settembre 2003;
- Incontro col committente dopo stesura diagramma Agent Identification: 22 settembre 2003;
- Incontro col committente dopo stesura diagrammi Domain Ontology e Communication Ontology: 8 ottobre 2003;
- rielaborazione, correzione e discussione con riunione interna del progetto: 16 ottobre 2003;
- riassunto e presentazione risultati della fase di progettazione al committente: 21 ottobre 2003;
- presentazione e dimostrazione prima versione del software al cliente: 12 gennaio 2004;
- consegna del prodotto: 13 aprile 2004.

Riportiamo di seguito la descrizione delle varie attività e fasi di lavoro che si sono affrontate per poter realizzare il sistema software progettato.

Attività di studio

Questa fase è stata dedicata allo studio di fattibiltà del progetto e porta ad una prima descrizione formale.

Si è provveduto poi al consolidamento delle conoscenze del linguaggio Java, dei tools Microsoft Project e Access e all'apprendimento dell'uso del tool Rational Rose, della piattaforma Jade e del linguaggio SQL.

Questa prima fase scaturisce nella stesura di questo documento.

Presentazione al cliente

Presentazione del progetto al cliente e approvazione delle funzionalità del sistema.

Progettazione software

In questa attività sono state svolte le varie fasi di progettazione previste da PASSI:

- System Requirements Model: descrive i requisiti del sistema, le funzionalità degli agenti, i ruoli giocati da ogni agente nel portare a termine i propri compiti e i comportamenti;
- Agent Society Model: include la descrizione dell'ontologia del sistema e delle comunicazioni tra gli agenti, dei ruoli e dei protocolli utilizzati;
- Agent Implementation Model: include la descrizione della struttura e dei comportamenti di ogni singolo agente e della società degli agenti;

Inoltre è stato effettuato il progetto concettuale e logico del Database utilizzato.

Implementazione

Durante questa fase è stata effettuata la codifica delle interfacce e dei singoli agenti con l'ausilio del tool "Agent Factory".

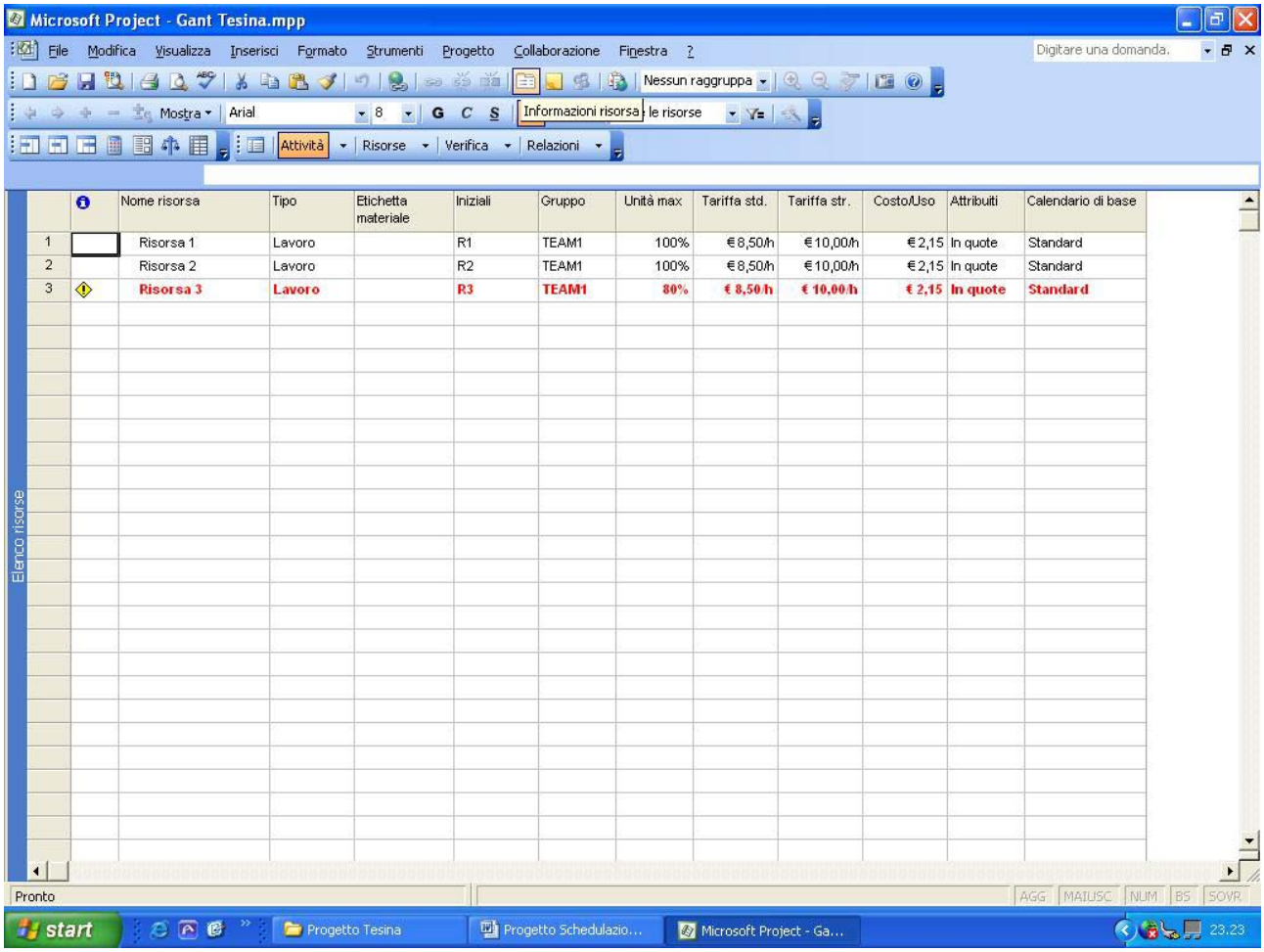
Testing

L'attività di Testing risulta molto importante al fine di poter identificare il maggior numero di errori che possono essere stati generati in fase di sviluppo o le eventuali discrepanze del programma che si possono presentare.

Diagrammi di pianificazione del progetto

Risorse del progetto

Mostriamo adesso le risorse che vengono utilizzate nel nostro progetto e gli attributi più importanti di esse.

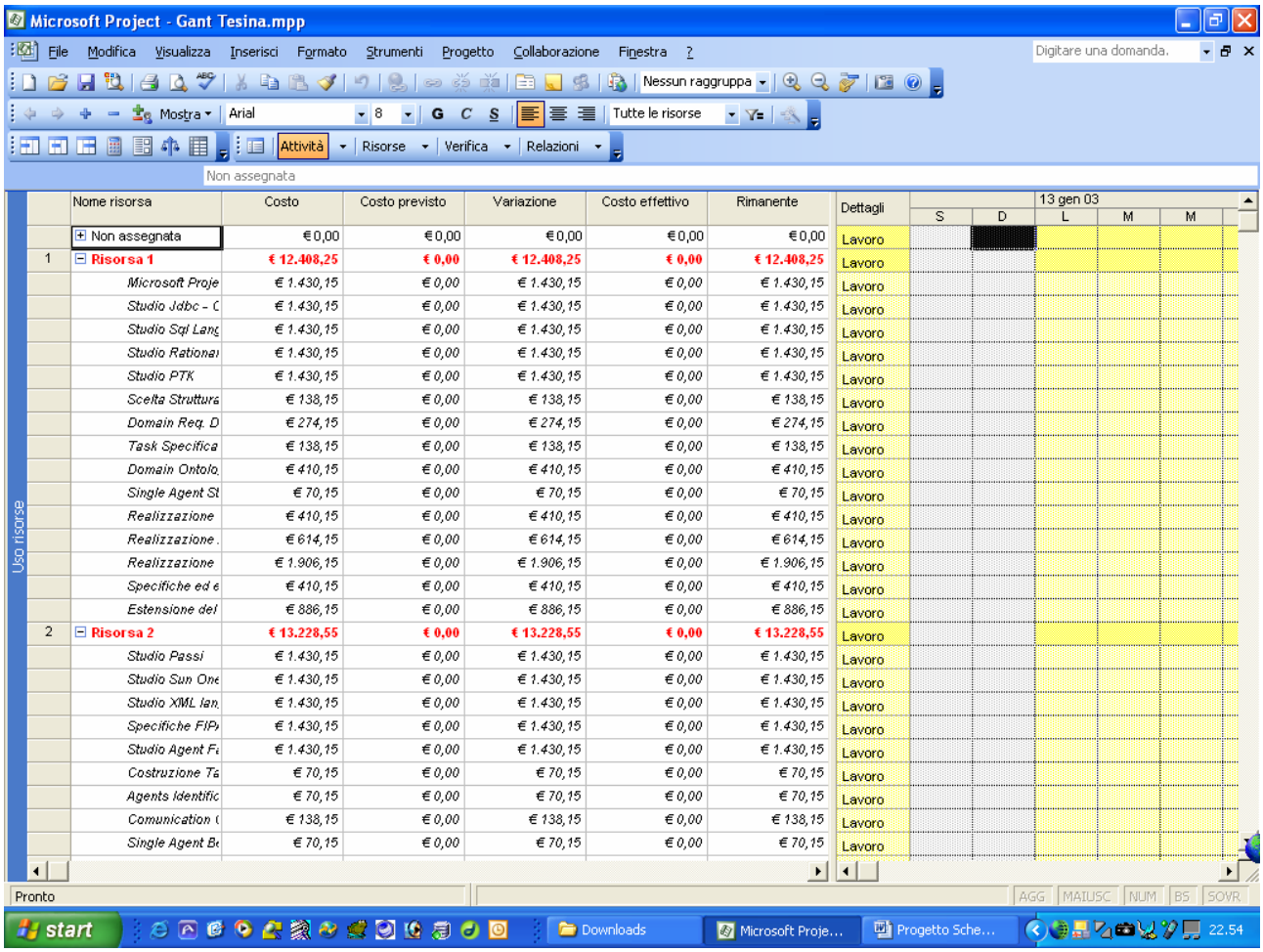


	Nome risorsa	Tipo	Etichetta materiale	Iniziali	Gruppo	Unità max	Tariffa std.	Tariffa str.	Costo/Uso	Attribuiti	Calendario di base
1	Risorsa 1	Lavoro		R1	TEAM1	100%	€8,50/h	€10,00/h	€2,15	In quote	Standard
2	Risorsa 2	Lavoro		R2	TEAM1	100%	€8,50/h	€10,00/h	€2,15	In quote	Standard
3	**Risorsa 3**	**Lavoro**		**R3**	**TEAM1**	**80%**	**€ 8,50/h**	**€ 10,00/h**	**€ 2,15**	**In quote**	**Standard**

Come si può vedere le risorse associata al progetto sono 3 che per convenzione abbiamo chiamato Risorsa1, Risorsa2 e Risorsa3. Nei progetti generalmente a tutte le risorse vengono associate diverse attività. In questa commessa tutte le risorse saranno interessate a tutto il progetto al fine di avere una conoscenza omogenea dell'applicazione e del processo produttivo seguito.

Nella nostra commessa si è ipotizzato un costo per ora ordinaria di 8,50€, nel caso di straordinario le ore saranno pagate a 10,00 €. Come si può evincere dal diagramma sopra riportato. Una delle 3 risorse, come può accadere nella realtà, può essere assegnata alla commessa ma non essere produttivo al 100% per svariate motivazioni. Infatti la risorsa 3 nel nostro diagramma potrà essere impegnata all' 80%. L'applicativo automaticamente come si può vedere mostra questa incongruenza tramite la riga rossa che identifica un problema nella gestione delle risorse.

Riepilogo Costi Produzione



	Nome risorsa	Costo	Costo previsto	Variazione	Costo effettivo	Rimanente	Dettagli
	Non assegnata	€ 0,00	€ 0,00	€ 0,00	€ 0,00	€ 0,00	Lavoro
1	**Risorsa 1**	**€ 12.408,25**	**€ 0,00**	**€ 12.408,25**	**€ 0,00**	**€ 12.408,25**	Lavoro
	Microsoft Proje	€ 1.430,15	€ 0,00	€ 1.430,15	€ 0,00	€ 1.430,15	Lavoro
	Studio Jdbc - C	€ 1.430,15	€ 0,00	€ 1.430,15	€ 0,00	€ 1.430,15	Lavoro
	Studio Sql Lang	€ 1.430,15	€ 0,00	€ 1.430,15	€ 0,00	€ 1.430,15	Lavoro
	Studio Rational	€ 1.430,15	€ 0,00	€ 1.430,15	€ 0,00	€ 1.430,15	Lavoro
	Studio PTK	€ 1.430,15	€ 0,00	€ 1.430,15	€ 0,00	€ 1.430,15	Lavoro
	Scelta Struttura	€ 138,15	€ 0,00	€ 138,15	€ 0,00	€ 138,15	Lavoro
	Domain Req. D	€ 274,15	€ 0,00	€ 274,15	€ 0,00	€ 274,15	Lavoro
	Task Specifica	€ 138,15	€ 0,00	€ 138,15	€ 0,00	€ 138,15	Lavoro
	Domain Ontolo	€ 410,15	€ 0,00	€ 410,15	€ 0,00	€ 410,15	Lavoro
	Single Agent St	€ 70,15	€ 0,00	€ 70,15	€ 0,00	€ 70,15	Lavoro
	Realizzazione	€ 410,15	€ 0,00	€ 410,15	€ 0,00	€ 410,15	Lavoro
	Realizzazione	€ 614,15	€ 0,00	€ 614,15	€ 0,00	€ 614,15	Lavoro
	Realizzazione	€ 1.906,15	€ 0,00	€ 1.906,15	€ 0,00	€ 1.906,15	Lavoro
	Specifiche ed e	€ 410,15	€ 0,00	€ 410,15	€ 0,00	€ 410,15	Lavoro
	Estensione del	€ 886,15	€ 0,00	€ 886,15	€ 0,00	€ 886,15	Lavoro
2	**Risorsa 2**	**€ 13.228,55**	**€ 0,00**	**€ 13.228,55**	**€ 0,00**	**€ 13.228,55**	Lavoro
	Studio Passi	€ 1.430,15	€ 0,00	€ 1.430,15	€ 0,00	€ 1.430,15	Lavoro
	Studio Sun One	€ 1.430,15	€ 0,00	€ 1.430,15	€ 0,00	€ 1.430,15	Lavoro
	Studio XML lan	€ 1.430,15	€ 0,00	€ 1.430,15	€ 0,00	€ 1.430,15	Lavoro
	Specifiche FIPA	€ 1.430,15	€ 0,00	€ 1.430,15	€ 0,00	€ 1.430,15	Lavoro
	Studio Agent Fa	€ 1.430,15	€ 0,00	€ 1.430,15	€ 0,00	€ 1.430,15	Lavoro
	Costruzione Ta	€ 70,15	€ 0,00	€ 70,15	€ 0,00	€ 70,15	Lavoro
	Agents Identific	€ 70,15	€ 0,00	€ 70,15	€ 0,00	€ 70,15	Lavoro
	Comunication	€ 138,15	€ 0,00	€ 138,15	€ 0,00	€ 138,15	Lavoro
	Single Agent Be	€ 70,15	€ 0,00	€ 70,15	€ 0,00	€ 70,15	Lavoro

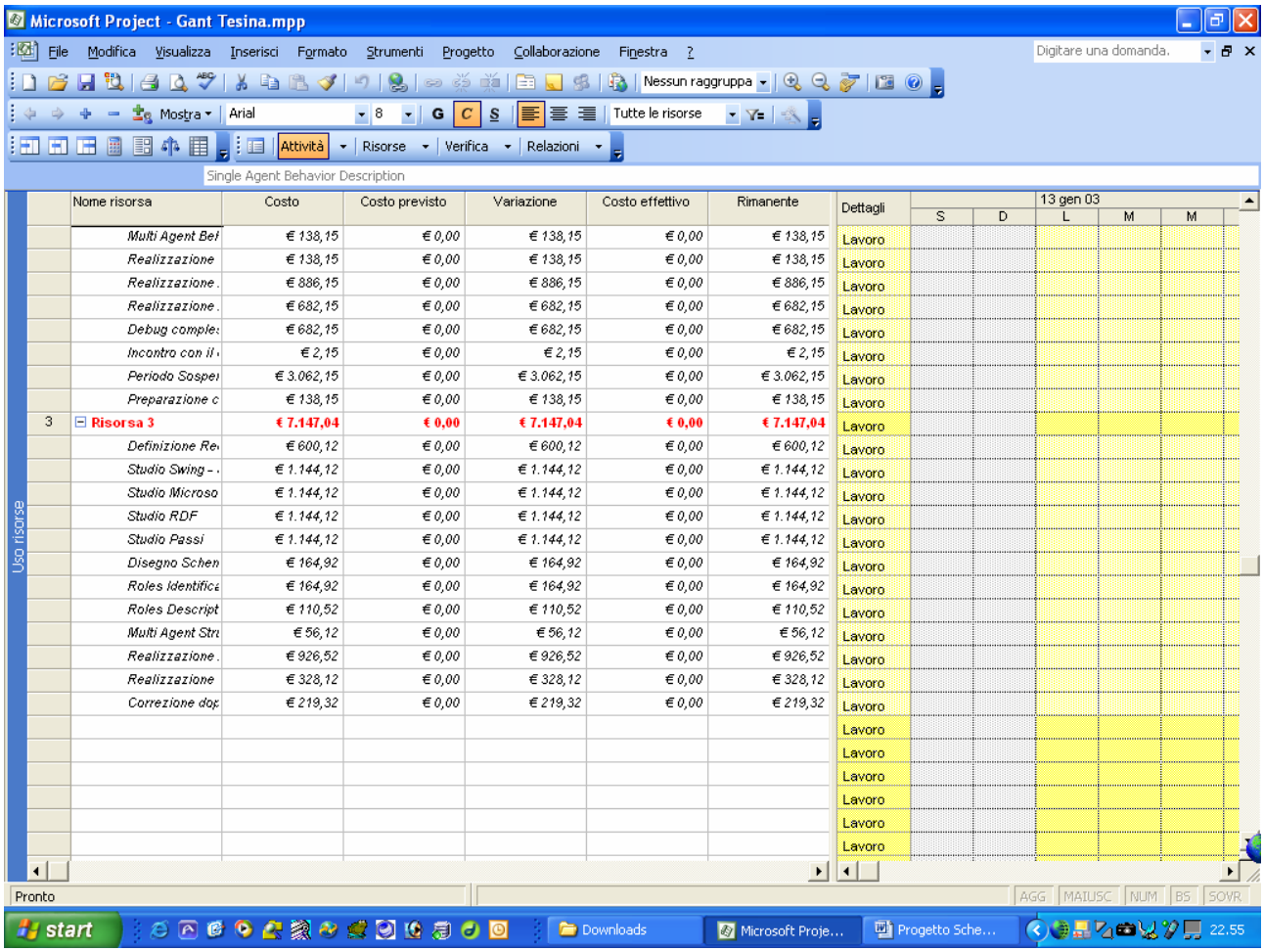
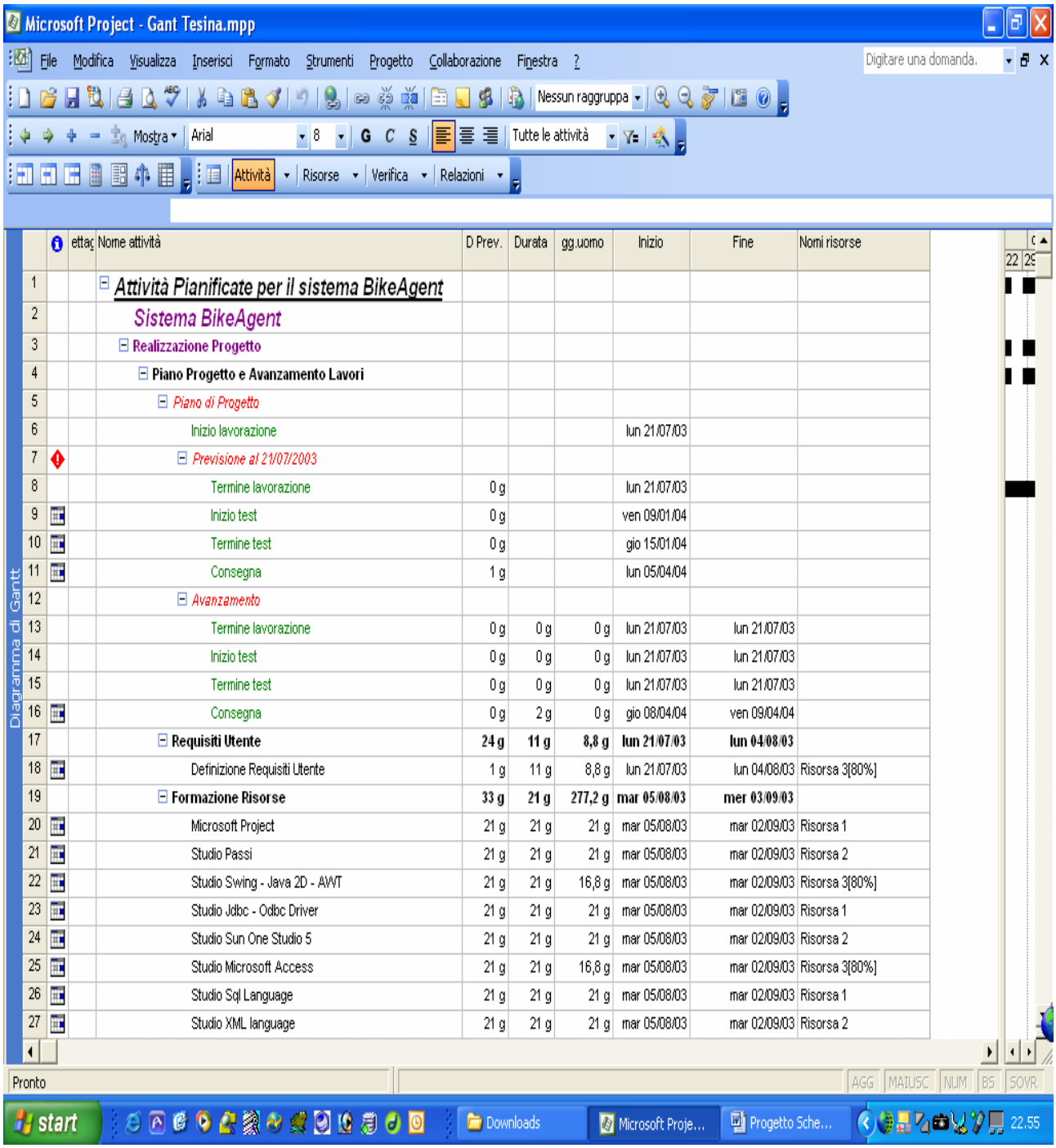
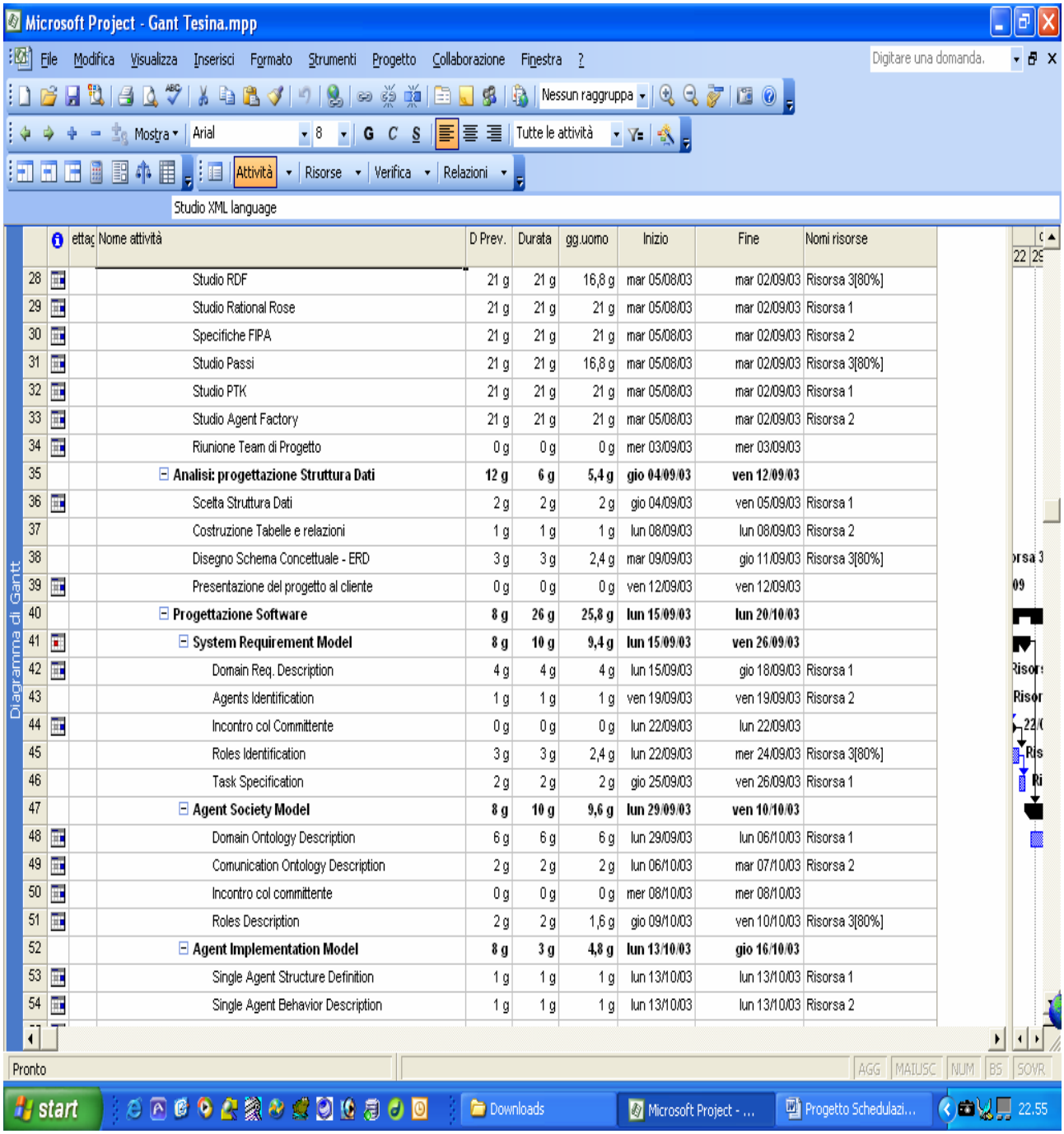

Microsoft Project - Gant Tesina.mpp
File Modifica Visualizza Inserisci Formato Strumenti Progetto Collaborazione Finestra ?
Digitare una domanda.
Nessun raggruppa
Mostra Arial 8 Tutte le risorse
Attività Risorse Verifica Relazioni
Single Agent Behavior Description
Nome risorsa Costo Costo previsto Variazione Costo effettivo Rimanente Dettagli
13 gen 03
S D L M M
Risorsa 3 € 7.147,04 € 0,00 € 7.147,04 € 0,00 € 7.147,04
Uso risorse
Pronto
AGG MAIUSC NUM BS SOVR
start
Downloads
Microsoft Proje...
Progetto Sche...
22.55

Diagramma di Gantt

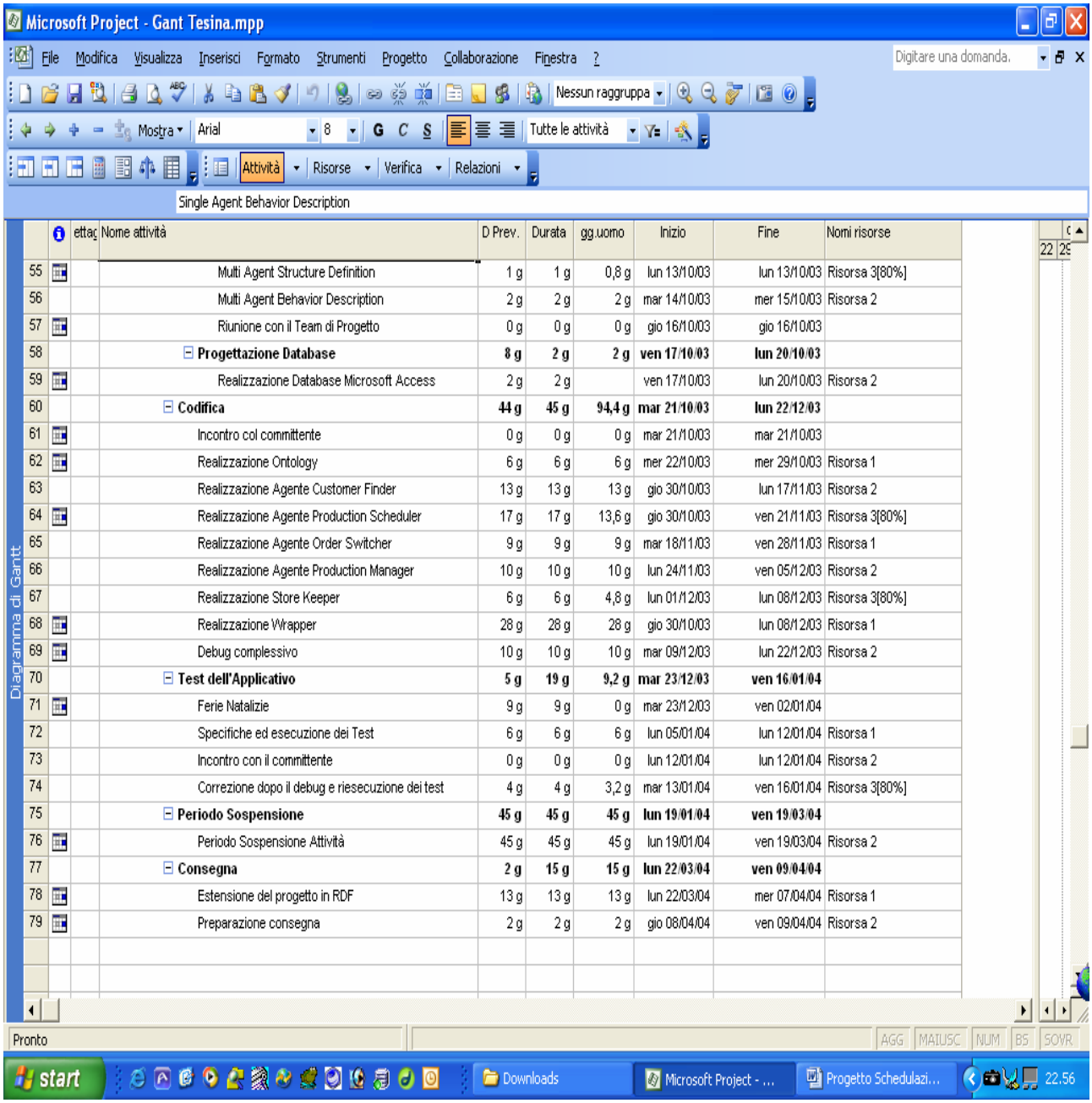
Presentiamo adesso il diagramma di Gantt con l'elenco delle risorse impiegate che mostra l'evoluzione temporale del progetto e delle attività affrontate per costruire il sistema.



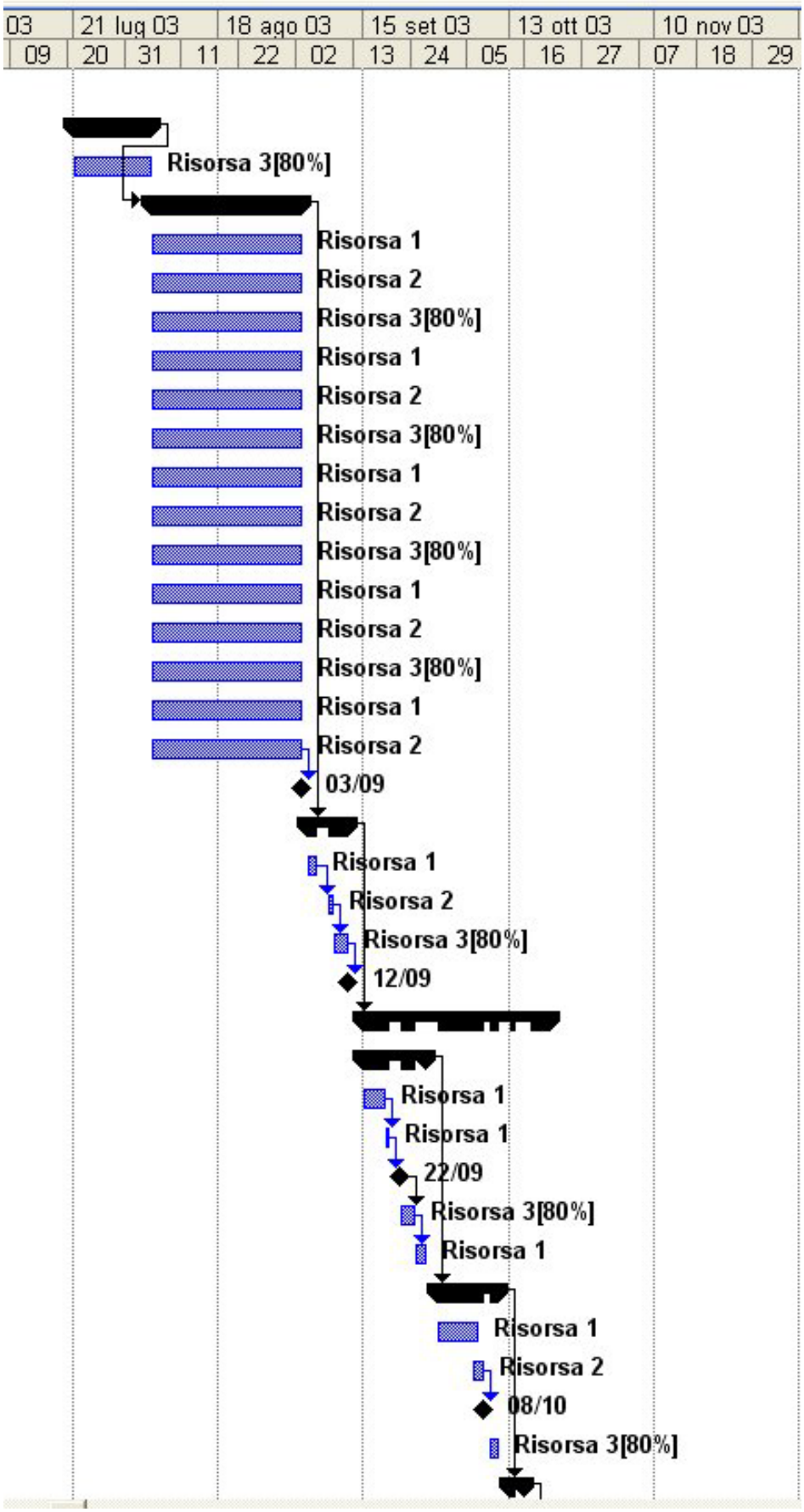
	ettaç	Nome attività	D Prev.	Durata	gg.uomo	Inizio	Fine	Nomi risorse
1		*Attività Pianificate per il sistema BikeAgent*						
2		*Sistema BikeAgent*						
3		Realizzazione Progetto						
4		Piano Progetto e Avanzamento Lavori						
5		*Piano di Progetto*						
6		Inizio lavorazione				lun 21/07/03		
7		*Previsione al 21/07/2003*						
8		Termine lavorazione	0 g			lun 21/07/03		
9		Inizio test	0 g			ven 09/01/04		
10		Termine test	0 g			gio 15/01/04		
11		Consegna	1 g			lun 05/04/04		
12		*Avanzamento*						
13		Termine lavorazione	0 g	0 g	0 g	lun 21/07/03	lun 21/07/03	
14		Inizio test	0 g	0 g	0 g	lun 21/07/03	lun 21/07/03	
15		Termine test	0 g	0 g	0 g	lun 21/07/03	lun 21/07/03	
16		Consegna	0 g	2 g	0 g	gio 08/04/04	ven 09/04/04	
17		**Requisiti Utente**	**24 g**	**11 g**	**8,8 g**	**lun 21/07/03**	**lun 04/08/03**	
18		Definizione Requisiti Utente	1 g	11 g	8,8 g	lun 21/07/03	lun 04/08/03	Risorsa 3[80%]
19		**Formazione Risorse**	**33 g**	**21 g**	**277,2 g**	**mar 05/08/03**	**mer 03/09/03**	
20		Microsoft Project	21 g	21 g	21 g	mar 05/08/03	mar 02/09/03	Risorsa 1
21		Studio Passi	21 g	21 g	21 g	mar 05/08/03	mar 02/09/03	Risorsa 2
22		Studio Swing - Java 2D - AWT	21 g	21 g	16,8 g	mar 05/08/03	mar 02/09/03	Risorsa 3[80%]
23		Studio Jdbc - Odbc Driver	21 g	21 g	21 g	mar 05/08/03	mar 02/09/03	Risorsa 1
24		Studio Sun One Studio 5	21 g	21 g	21 g	mar 05/08/03	mar 02/09/03	Risorsa 2
25		Studio Microsoft Access	21 g	21 g	16,8 g	mar 05/08/03	mar 02/09/03	Risorsa 3[80%]
26		Studio Sql Language	21 g	21 g	21 g	mar 05/08/03	mar 02/09/03	Risorsa 1
27		Studio XML language	21 g	21 g	21 g	mar 05/08/03	mar 02/09/03	Risorsa 2

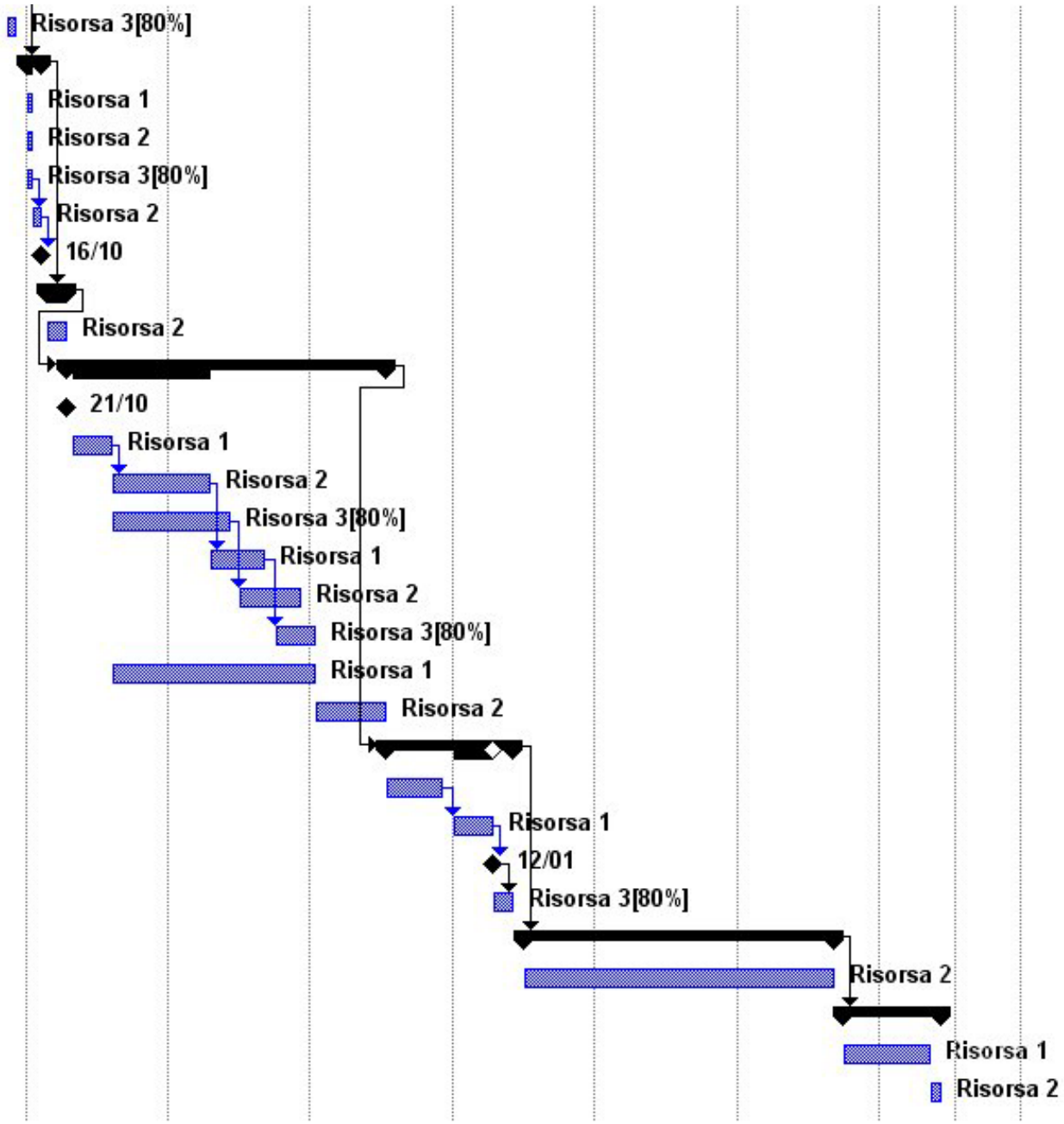


	ettag	Nome attività	D Prev.	Durata	gg.uomo	Inizio	Fine	Nomi risorse
28		Studio RDF	21 g	21 g	16,8 g	mar 05/08/03	mar 02/09/03	Risorsa 3[80%]
29		Studio Rational Rose	21 g	21 g	21 g	mar 05/08/03	mar 02/09/03	Risorsa 1
30		Specifiche FIPA	21 g	21 g	21 g	mar 05/08/03	mar 02/09/03	Risorsa 2
31		Studio Passi	21 g	21 g	16,8 g	mar 05/08/03	mar 02/09/03	Risorsa 3[80%]
32		Studio PTK	21 g	21 g	21 g	mar 05/08/03	mar 02/09/03	Risorsa 1
33		Studio Agent Factory	21 g	21 g	21 g	mar 05/08/03	mar 02/09/03	Risorsa 2
34		Riunione Team di Progetto	0 g	0 g	0 g	mer 03/09/03	mer 03/09/03	
35		**Analisi: progettazione Struttura Dati**	**12 g**	**6 g**	**5,4 g**	**gio 04/09/03**	**ven 12/09/03**	
36		Scelta Struttura Dati	2 g	2 g	2 g	gio 04/09/03	ven 05/09/03	Risorsa 1
37		Costruzione Tabelle e relazioni	1 g	1 g	1 g	lun 08/09/03	lun 08/09/03	Risorsa 2
38		Disegno Schema Concettuale - ERD	3 g	3 g	2,4 g	mar 09/09/03	gio 11/09/03	Risorsa 3[80%]
39		Presentazione del progetto al cliente	0 g	0 g	0 g	ven 12/09/03	ven 12/09/03	
40		**Progettazione Software**	**8 g**	**26 g**	**25,8 g**	**lun 15/09/03**	**lun 20/10/03**	
41		**System Requirement Model**	**8 g**	**10 g**	**9,4 g**	**lun 15/09/03**	**ven 26/09/03**	
42		Domain Req. Description	4 g	4 g	4 g	lun 15/09/03	gio 18/09/03	Risorsa 1
43		Agents Identification	1 g	1 g	1 g	ven 19/09/03	ven 19/09/03	Risorsa 2
44		Incontro col Committente	0 g	0 g	0 g	lun 22/09/03	lun 22/09/03	
45		Roles Identification	3 g	3 g	2,4 g	lun 22/09/03	mer 24/09/03	Risorsa 3[80%]
46		Task Specification	2 g	2 g	2 g	gio 25/09/03	ven 26/09/03	Risorsa 1
47		**Agent Society Model**	**8 g**	**10 g**	**9,6 g**	**lun 29/09/03**	**ven 10/10/03**	
48		Domain Ontology Description	6 g	6 g	6 g	lun 29/09/03	lun 06/10/03	Risorsa 1
49		Comunication Ontology Description	2 g	2 g	2 g	lun 06/10/03	mar 07/10/03	Risorsa 2
50		Incontro col committente	0 g	0 g	0 g	mer 08/10/03	mer 08/10/03	
51		Roles Description	2 g	2 g	1,6 g	gio 09/10/03	ven 10/10/03	Risorsa 3[80%]
52		**Agent Implementation Model**	**8 g**	**3 g**	**4,8 g**	**lun 13/10/03**	**gio 16/10/03**	
53		Single Agent Structure Definition	1 g	1 g	1 g	lun 13/10/03	lun 13/10/03	Risorsa 1
54		Single Agent Behavior Description	1 g	1 g	1 g	lun 13/10/03	lun 13/10/03	Risorsa 2



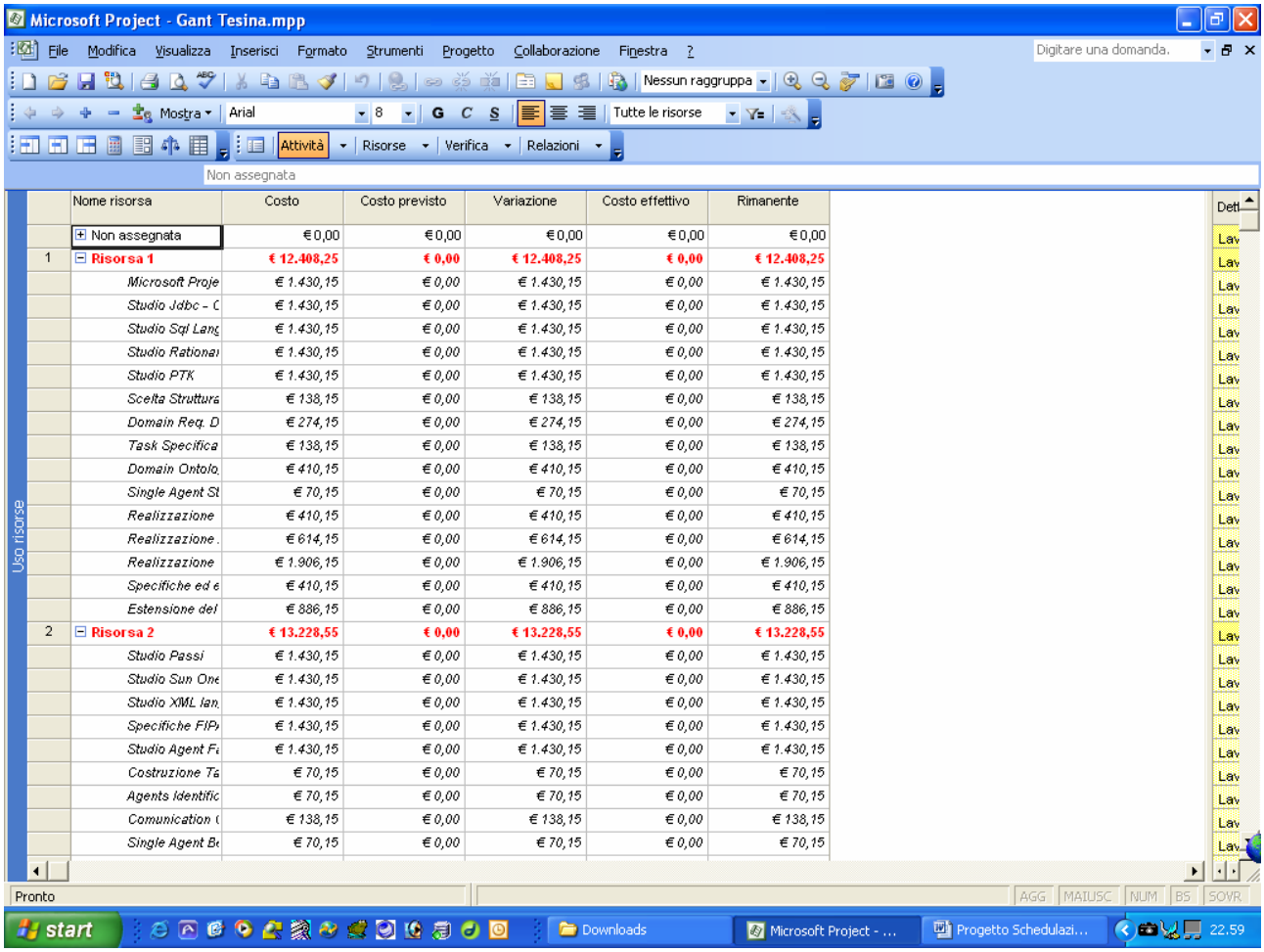
		Nome attività	D Prev.	Durata	gg.uomo	Inizio	Fine	Nomi risorse
55		Multi Agent Structure Definition	1 g	1 g	0,8 g	lun 13/10/03	lun 13/10/03	Risorsa 3[80%]
56		Multi Agent Behavior Description	2 g	2 g	2 g	mar 14/10/03	mer 15/10/03	Risorsa 2
57		Riunione con il Team di Progetto	0 g	0 g	0 g	gio 16/10/03	gio 16/10/03	
58		**Progettazione Database**	**8 g**	**2 g**	**2 g**	**ven 17/10/03**	**lun 20/10/03**	
59		Realizzazione Database Microsoft Access	2 g	2 g		ven 17/10/03	lun 20/10/03	Risorsa 2
60		**Codifica**	**44 g**	**45 g**	**94,4 g**	**mar 21/10/03**	**lun 22/12/03**	
61		Incontro col committente	0 g	0 g	0 g	mar 21/10/03	mar 21/10/03	
62		Realizzazione Ontology	6 g	6 g	6 g	mer 22/10/03	mer 29/10/03	Risorsa 1
63		Realizzazione Agente Customer Finder	13 g	13 g	13 g	gio 30/10/03	lun 17/11/03	Risorsa 2
64		Realizzazione Agente Production Scheduler	17 g	17 g	13,6 g	gio 30/10/03	ven 21/11/03	Risorsa 3[80%]
65		Realizzazione Agente Order Switcher	9 g	9 g	9 g	mar 18/11/03	ven 28/11/03	Risorsa 1
66		Realizzazione Agente Production Manager	10 g	10 g	10 g	lun 24/11/03	ven 05/12/03	Risorsa 2
67		Realizzazione Store Keeper	6 g	6 g	4,8 g	lun 01/12/03	lun 08/12/03	Risorsa 3[80%]
68		Realizzazione Wrapper	28 g	28 g	28 g	gio 30/10/03	lun 08/12/03	Risorsa 1
69		Debug complessivo	10 g	10 g	10 g	mar 09/12/03	lun 22/12/03	Risorsa 2
70		**Test dell'Applicativo**	**5 g**	**19 g**	**9,2 g**	**mar 23/12/03**	**ven 16/01/04**	
71		Ferie Natalizie	9 g	9 g	0 g	mar 23/12/03	ven 02/01/04	
72		Specifiche ed esecuzione dei Test	6 g	6 g	6 g	lun 05/01/04	lun 12/01/04	Risorsa 1
73		Incontro con il committente	0 g	0 g	0 g	lun 12/01/04	lun 12/01/04	Risorsa 2
74		Correzione dopo il debug e riesecuzione dei test	4 g	4 g	3,2 g	mar 13/01/04	ven 16/01/04	Risorsa 3[80%]
75		**Periodo Sospensione**	**45 g**	**45 g**	**45 g**	**lun 19/01/04**	**ven 19/03/04**	
76		Periodo Sospensione Attività	45 g	45 g	45 g	lun 19/01/04	ven 19/03/04	Risorsa 2
77		**Consegna**	**2 g**	**15 g**	**15 g**	**lun 22/03/04**	**ven 09/04/04**	
78		Estensione del progetto in RDF	13 g	13 g	13 g	lun 22/03/04	mer 07/04/04	Risorsa 1
79		Preparazione consegna	2 g	2 g	2 g	gio 08/04/04	ven 09/04/04	Risorsa 2

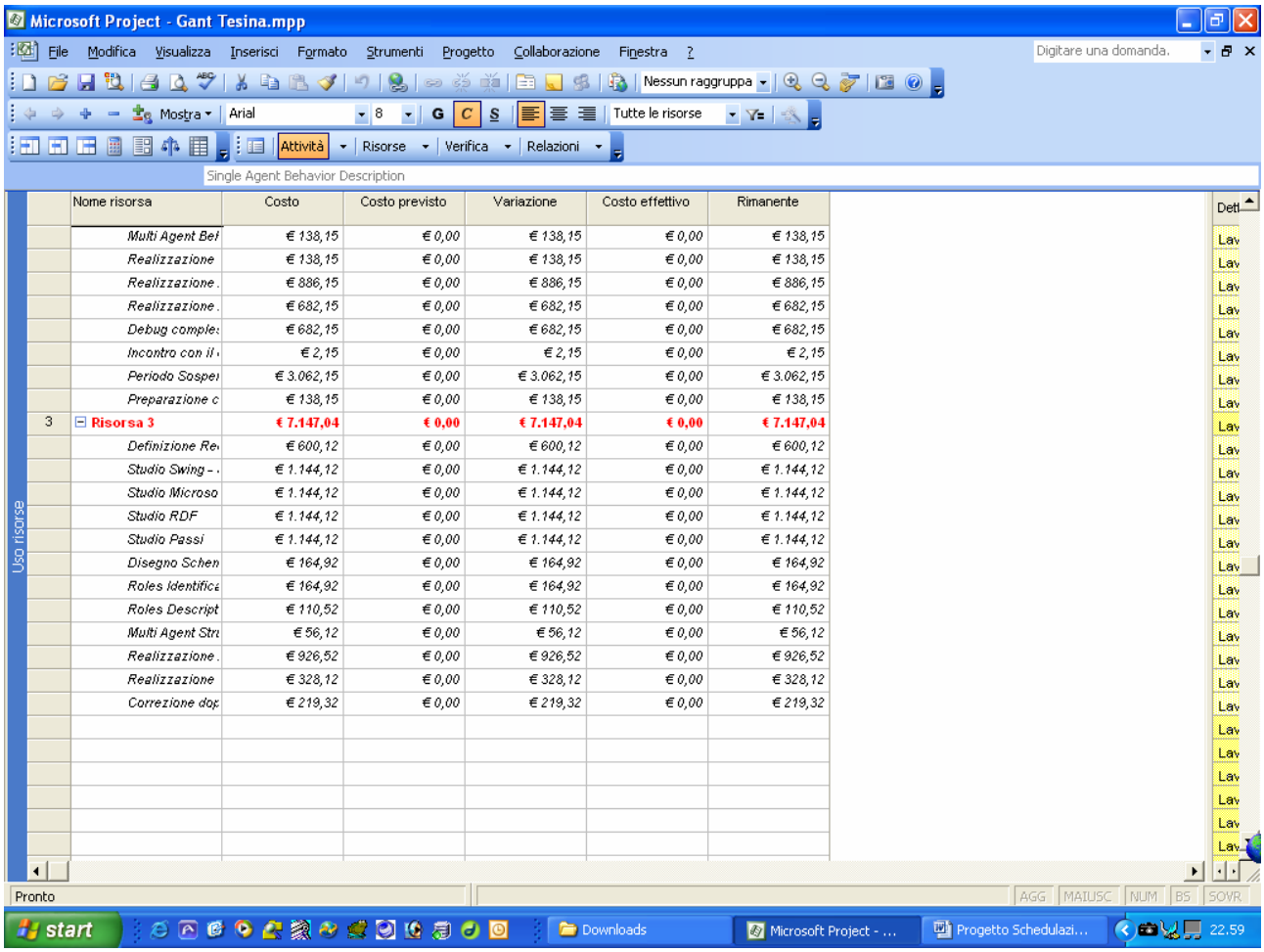

03
21 lug 03
18 ago 03
15 set 03
13 ott 03
10 nov 03
09 20 31 11 22 02 13 24 05 16 27 07 18 29
Risorsa 3[80%]
Risorsa 1
Risorsa 2
Risorsa 3[80%]
Risorsa 1
Risorsa 2
Risorsa 3[80%]
Risorsa 1
Risorsa 2
Risorsa 3[80%]
Risorsa 1
Risorsa 2
Risorsa 3[80%]
Risorsa 1
Risorsa 2
03/09
Risorsa 1
Risorsa 2
Risorsa 3[80%]
12/09
Risorsa 1
Risorsa 1
22/09
Risorsa 3[80%]
Risorsa 1
Risorsa 1
Risorsa 2
08/10
Risorsa 3[80%]


Risorsa 3[80%]
Risorsa 1
Risorsa 2
Risorsa 3[80%]
Risorsa 2
16/10
Risorsa 2
21/10
Risorsa 1
Risorsa 2
Risorsa 3[80%]
Risorsa 1
Risorsa 2
Risorsa 3[80%]
Risorsa 1
Risorsa 2
Risorsa 1
12/01
Risorsa 3[80%]
Risorsa 2
Risorsa 1
Risorsa 2

Costi del progetto

Inseriamo adesso un grafico dal project che tiene conto dei costi totali sostenuti nelle varie fasi di sviluppo del prodotto.

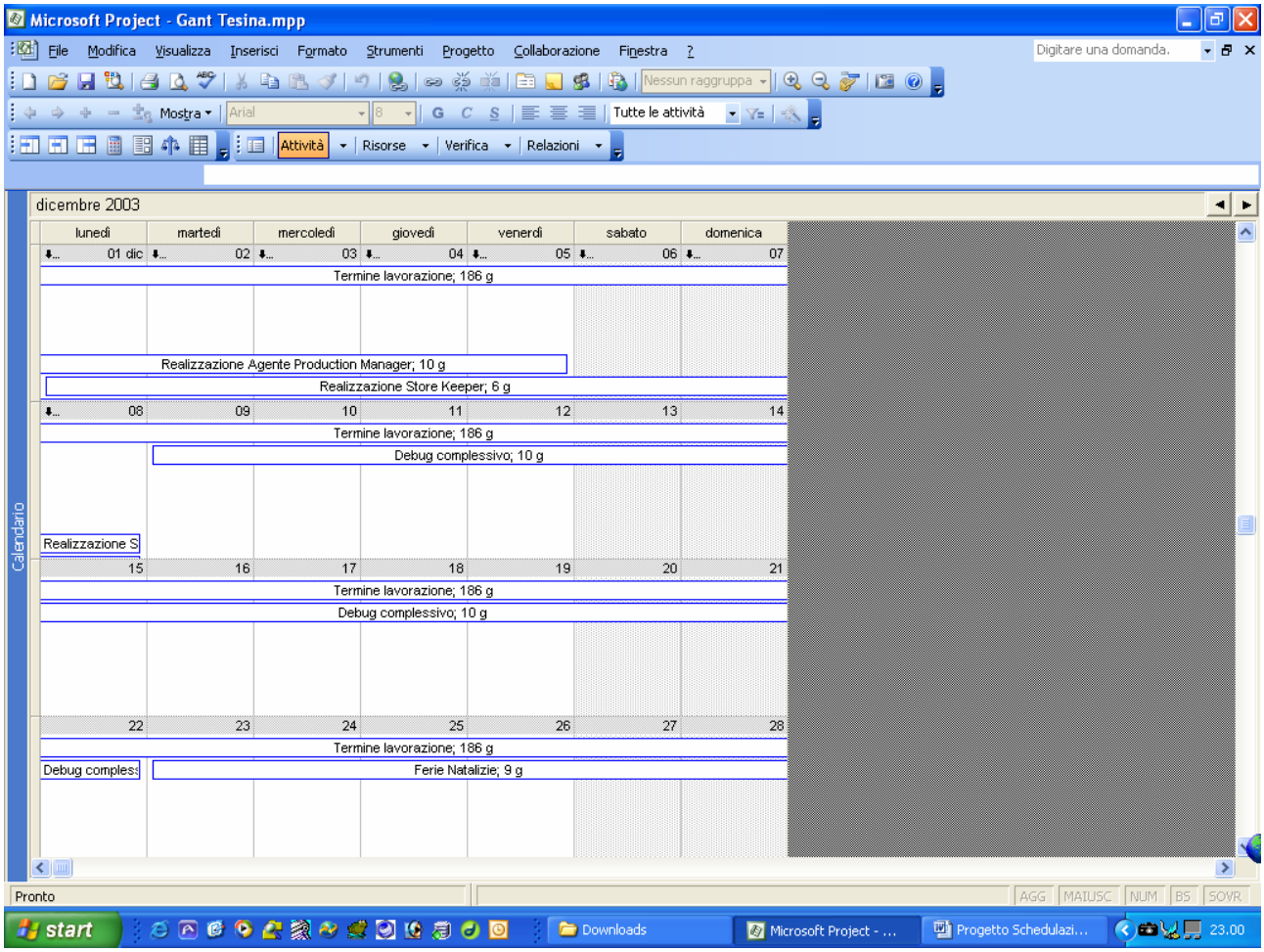



Microsoft Project - Gant Tesina.mpp
File Modifica Visualizza Inserisci Formato Strumenti Progetto Collaborazione Finestra ?
Digitare una domanda.
Nessun raggruppa
Mostra Arial 8 G C S Tutte le risorse
Attività Risorse Verifica Relazioni
Single Agent Behavior Description
Nome risorsa Costo Costo previsto Variazione Costo effettivo Rimanente
Multi Agent Bel € 138,15 € 0,00 € 138,15 € 0,00 € 138,15
Realizzazione € 138,15 € 0,00 € 138,15 € 0,00 € 138,15
Realizzazione. € 886,15 € 0,00 € 886,15 € 0,00 € 886,15
Realizzazione. € 682,15 € 0,00 € 682,15 € 0,00 € 682,15
Debug comple: € 682,15 € 0,00 € 682,15 € 0,00 € 682,15
Incontro con il € 2,15 € 0,00 € 2,15 € 0,00 € 2,15
Periodo Sospe € 3.062,15 € 0,00 € 3.062,15 € 0,00 € 3.062,15
Preparazione c € 138,15 € 0,00 € 138,15 € 0,00 € 138,15
3 Risorsa 3 € 7.147,04 € 0,00 € 7.147,04 € 0,00 € 7.147,04
Definizione Re € 600,12 € 0,00 € 600,12 € 0,00 € 600,12
Studio Swing - € 1.144,12 € 0,00 € 1.144,12 € 0,00 € 1.144,12
Studio Microso € 1.144,12 € 0,00 € 1.144,12 € 0,00 € 1.144,12
Studio RDF € 1.144,12 € 0,00 € 1.144,12 € 0,00 € 1.144,12
Studio Passi € 1.144,12 € 0,00 € 1.144,12 € 0,00 € 1.144,12
Disegno Schen € 164,92 € 0,00 € 164,92 € 0,00 € 164,92
Roles Identifica € 164,92 € 0,00 € 164,92 € 0,00 € 164,92
Roles Descript € 110,52 € 0,00 € 110,52 € 0,00 € 110,52
Multi Agent Stru € 56,12 € 0,00 € 56,12 € 0,00 € 56,12
Realizzazione. € 926,52 € 0,00 € 926,52 € 0,00 € 926,52
Realizzazione € 328,12 € 0,00 € 328,12 € 0,00 € 328,12
Correzione dop € 219,32 € 0,00 € 219,32 € 0,00 € 219,32
Uso risorse
Pronto
AGG MAIUSC NUM BS SOVR
start
Downloads
Microsoft Project - ...
Progetto Schedulazi...
22.59

Calendario delle Attività del progetto

Mostriamo infine una visione dettagliata del calendario delle attività :

Si riporta il mese di dicembre a titolo di esempio:



ANALISI DEI REQUISITI

Sistema BikesAgent

Redatto da:	Caico Roberto Gargagliano Giuseppe Termine Francesco	_____	_____
Emesso da:	Caico Roberto Gargagliano Giuseppe Termine Francesco	_____	_____
Approvato da:	Caico Roberto Gargagliano Giuseppe Termine Francesco	_____	_____

Tipo documento:	RAD	.01	.01
	Commessa	*Num. Doc.*	*Vers.*

Prima emissione: 12/04/2004

Distribuzione: Interna

Data di stampa: 18/04/04 18.25

Path del file: *C:\documenti\tesina*

La validità di questa informazione è garantita al momento della stampa del documento.

Obiettivi generali

Obiettivo di questo progetto è quello di automatizzare ed informatizzare il lavoro d'impresa di una fabbrica di biciclette. Tale obiettivo verrà perseguito mediante una sistema informativo che non violi il fine aziendale né la filosofia produttiva dell'impresa. Il sistema accelererà lo scambio di informazioni all'interno dell'azienda mediante un sistema informatico efficiente ed efficace, ottimizzerà il lavoro delle singole figure professionali coinvolte nella catena produttiva e si avvantaggerà delle nuove tecnologie informatiche.

Sistema corrente

Nel progetto non è prevista la reingegnerizzazione di alcun tipo di un sistema informatico preesistente. Il lavoro, lungo la catena produttiva, è stato sempre svolto mediante supporti cartacei e sistemi di comunicazioni semplici quali fax, telefono, e posta. Non è pensabile, quindi, di integrare nel sistema proposto alcun meccanismo della precedente gestione aziendale, proponendo invece un sistema che ricalchi la precedente struttura ma che in realtà è pensato, progettato e realizzato ex-novo.

Sistema proposto

Introduzione

Il sistema proposto è pensato per essere usato dalle figure professionali dell'azienda. Quindi avrà le stesse funzionalità che queste hanno sempre utilizzato, migliorate però dalla gestione coerente e automatica delle informazioni. Garantirà la gestione dei clienti, l'inserimento, la modifica e la cancellazione degli ordini. Sarà supportata la scomposizione degli ordini in lotti e lo smistamento di questi ai relativi stabilimenti. Mediante un sistema grafico efficace e interattivo sarà semplificata l'attività di schedulazione dei lotti; verrà altresì supportata l'operazione di inventario delle bici prodotte e garantita la gestione di approvvigionamento delle materie prime. Tutto ciò tramite interfacce semplici e intuitive che semplificheranno le operazioni degli utenti addetti ai vari stabilimenti. Tutto è progettato in modo tale da richiedere agli utenti del sistema un training minimo, diminuendo così i costi di formazione del personale.

Requisiti funzionali

Le funzionalità dell'intero sistema vengono suddivise per una più facile lettura nella gestione dei vari settori della catena produttiva. E' ininfluente ai fini del progetto concepire come del tutto analoghe le funzionalità dei due stabilimenti in cui si producono i due modelli diversi di biciclette

- Gestione settore amministrativo
- Gestione settore produttivo
- Gestione stabilimento
- Gestione magazzino

Gestione settore amministrativo

Figura professionale: Addetto all'ufficio clienti.

Il sistema permetterà in fase amministrativa di:

- Raccogliere gli ordini della clientela
- Gli ordini verranno caratterizzati dall'inserimento, assieme alle quantità per ogni modello, anche delle date di consegna
- Gestione (modifica/cancellazione) dei dati degli ordini e dei clienti inseriti nel database

Gestione settore produttivo

Figura professionale: Addetto alla produzione, responsabile settore produzione.

Il sistema permetterà in fase produttiva di:

- Supportare l'operatore nella creazione dei lotti
- Verranno creati automaticamente dei lotti di produzione caratterizzati da un numero di pezzi dello stesso modello compreso tra 50 e 150 e da data di consegna vicina
- L'utente potrà quindi modificare i lotti creati creando eventualmente lotti anomali
- Smistare i lotti creati ai rispettivi stabilimenti

Gestione stabilimento

Figura professionale: Responsabile di stabilimento, Operaio addetto alla produzione

Il sistema permetterà in ognuno degli stabilimenti di:

- Ricevere i lotti di produzione e provvedere alla loro schedulazione

- Fornire un sistema grafico interattivo di supporto per la schedulazione
- Memorizzare, sovrascrivendo la vecchia, la schedulazione completata
- Confermare i lotti totalmente prodotti
- Fare l'inventario delle biciclette prodotte memorizzando per ognuna di esse
    - Il numero di telaio automaticamente creato dal software
    - La data di produzione
    - Il numero di lotto di produzione
    - Il cliente a cui si riferisce l'ordine
- Visualizzare e stampare le etichette da apporre nei telai

Gestione magazzino

Figura professionale: Responsabile di magazzino.

Il sistema permetterà in fase di gestione del magazzino di:

- Creare automaticamente la lista degli approvvigionamenti delle materie prime in base al numero delle biciclette prodotte dall'ultimo approvvigionamento
- Visualizzare e stampare la lista degli approvvigionamenti
- Creare/visualizzare/modificare le distinte di produzione delle biciclette

Requisiti non funzionali

Interfacce utente e fattori umani

Si suppone che le figure professionali che utilizzeranno il sistema non hanno alcun tipo di background informatico. Per questo motivo le interfacce sono state realizzate in modo semplice, ricalcando le procedure che erano precedentemente svolte da ciascuna figura professionale.

Ogni utente avrà la sua interfaccia dedicata e progettata per adempiere allo scopo del suo lavoro in modo efficace ed efficiente. Verranno utilizzati controlli per il corretto inserimento dei dati, come ad esempio: il formato delle date; correttezza dei numeri inseriti.

Considerazioni Hardware

Il sistema proposto non necessita di particolare hardware. E' necessario per ciascun agente una computer desktop di fascia media (pentium 4 2GHz, 256Mb Ram, monitor, tastiera e mouse).
Inoltre si suppone che tutti i pc dell'azienda siano collegati tramite rete Lan.
Inoltre è necessario:

- Una stampante laser per StoreKeeper
- una stampante di targhe metalliche per il ProductionManager

Si consiglia inoltre l'utilizzo di un portatile per l'addetto ufficio clienti, in modo da poter svolgere il suo compito oltre che nell'ufficio amministrazione anche in postazioni remote presso i clienti della ditta.

Caratteristiche delle performance

Il sistema garantirà una veloce interazione con l'utente evitando tempi morti e informandolo sullo stato dell'operazione corrente. Infatti al sistema non è richiesto un grosso costo computazionale e il tempo impiegato dal sistema per una risposta è di ordini di grandezza inferiori rispetto a quello speso dall'utente per il completamento di un'operazione.

Gestione degli errori e tolleranza ai guasti

Il sistema è stato progettato per essere Fault-Tolerance. Il primo filtro sugli errori avviene a livello di interfaccia con controlli sulla corretta immissione dei dati. Un secondo filtro avviene a livello di comunicazione di agenti, si informa sempre l'utente di un errore di comunicazione e sul successo o insuccesso di un'operazione. L'ultimo filtro è a livello di database, un errore di coerenza nei dati nel database viene rilevato e comunicato all'utente interessato.
Inoltre tutti gli agenti sono indipendenti gli uni dagli altri garantendo anche funzionalità parziali in caso di guasto tecnico o di personale mancante.
Unica eccezione fa il wrapper che è vitale per la gestione delle informazioni.
Sono inoltre ereditate tutte le garanzie di comunicazione e connessione dei sistemi basati su JADE.

Sicurezza

Non è stato implementato alcun tipo di sistema di sicurezza, si suppone che tutti gli utenti coinvolti nell'utilizzo del sistema siano fidati e abbiano fini non opposti a quelli dell'azienda. Si rimanda quindi all'amministratore del sistema la gestione della sicurezza mediante i comuni sistemi garantiti dai sistemi operativi.

Pseudorequisiti

Il sistema verrà sviluppato in linguaggio java, dato che questo è il linguaggio utilizzato dalla piattaforma ad agenti JADE.

In fase di sviluppo verrà utilizzato come sistema operativo il Microsoft Windows XP Professional e un database Microsoft Access perché sono di facile reperibilità sul mercato e di facile amministrazione.

Modelli del sistema

Identificazione degli attori

- **Addetto all'ufficio clienti** Figura professionale concepita alla stregua del "commesso viaggiatore" cioè colui che, da sede fissa in stabilimento, o tramite portatile in collegamento remoto, va alla ricerca di nuovi clienti e nuovi ordinativi; per tali motivi è lecito considerare tanti agenti Customer Finder quanti addetti alla ricerca di nuovi clienti.

  Compiti principali sono:

  - Inserire i dati di nuovi ordini
  - Inserire i dati di nuovi clienti
  - Visualizzare lo stato d ordini già inseriti
  - Modificare, quando possibile, dati degli ordini precedentemente inseriti
  - Visualizzare e modificare i dati di clienti già memorizzati

- **Addetto alla produzione** Figura professionale che si occupa di ricevere periodicamente (giornalmente, settimanalmente, o all'avvenuto inserimento di nuovi ordini da produrre), i dati dei nuovi ordini inseriti e di eseguirne la lottizzazione secondo specifiche standard assegnate (numero max di pezzi per lotto pari a 150 unità, ordini accorpabili per modello e data di consegna vicina).

  Compiti principali sono:

  - Leggere i nuovi ordini inseriti
  - Usare il software di supporto alla creazione dei lotti
  - Accorpare i lotti seguendo i criteri sopra elencati
  - Confermare i lotti creati e inviarli agli stabilimenti

- **Responsabile settore produzione** Figura professionale che si occupa della creazione dei lotti anomali, lotti cioè che non rientrano nelle specifiche di progetto ma che per questioni di gestione d'azienda è necessario produrre egualmente; verosimilmente nell'organigramma dell'azienda tale figura potrebbe svolgere contemporaneamente la mansione di addetto alla produzione e quindi creare interamente i lotti da produrre.
  Compiti principali sono:
  - Creare i lotti anomali
  - Confermare i lotti creati e inviarli agli stabilimenti

- **Responsabile di stabilimento** Figura professionale che si occupa di ricevere periodicamente (giornalmente, settimanalmente, o all'avvenuto inserimento di nuovi lotti da produrre), i dati dei nuovi lotti inseriti e di schedularne la produzione tenendo presente la massima capacità produttiva standard dello stabilimento di 50 biciclette al giorno.
  Compiti principali sono:
  - Leggere i nuovi lotti inseriti
  - Usare il software grafico di supporto alla creazione della schedulazione
  - Schedulare i lotti
  - Confermare la schedulazione ed inviarla allo stabilimento

- **Operaio addetto alla produzione** Figura professionale che si occupa della produzione dei lotti schedulati e della conferma dell'avvenuta produzione di un lotto.
  Compiti principali sono:
  - Leggere la schedulazione dei lotti da produrre
  - Confermare la produzione dei lotti
  - Usare il software di supporto per la produzione delle etichette
  - Visualizzare le etichette
  - Stampare le etichette e affiggerle nel telaio delle biciclette prodotte

- **Responsabile di magazzino** Figura professionale che si occupa della gestione del magazzino e quindi dell'approvvigionamento dei materiali per la produzione delle biciclette e della modifica delle distinte di produzione delle biciclette .

  Compiti principali sono:

  - Leggere, modificare e salvare le distinte di produzione
  - Usare il software di supporto alla creazione della lista dei materiali da approvvigionare
  - Stampare la lista ed inviarla ai fornitori

- **Database** Chiaramente non un essere umano ma visto come un sistema esterno che interagisce con l'intero sistema; in esso arrivano i dati da memorizzare da ogni settore della catena di produzione che provvede a immagazzinare mantenendo la coerenza e la coesione dei dati memorizzati.

Identificazione degli scenari

Scenario Name Inserimento e lottizzazione degli ordini di un nuovo cliente e visualizzazione della gestione dei messaggi del processo produttivo tra diversi stabilimenti

Participating actor istances	Pippo : Addetto ufficio clienti Ciccio : Addetto alla produzione Carlo : Responsabile della produzione Database
Entry condition	1. Pippo e Ciccio lanciano il ognuno il proprio agente; il wrapper (agente del database) si considera sempre attivo
Flow of Events	2. Pippo introduce i dati di un nuovo cliente e li salva nel database 3. Si sposta quindi nel pannello di immissione degli ordini e contratta col committente sul numero di biciclette da produrre, sul modello e sulla data di consegna accertandosi, dalla visualizzazione degli ordini già inseriti, se è possibile evadere tale ordinativo 4. Ciccio nel frattempo lottizza i nuovi ordini che automaticamente all'avvio del programma ha visualizzato e invia i nuovi lotti allo stabilimento 5. Pippo conclude l'operazione di contrattazione col committente ed invia l'ordine al database 6. Ciccio vede nella sua interfaccia che il bottone "Leggi Ordini" ha cambiato colore ed è divenuto rosso, segno che nuovi ordini sono stati inseriti 7. Clicca quindi sul corrispondente bottone e riceve i dati sui nuovi ordini dal database e provvede alla loro lottizzazione 8. Ciccio si accorge che nella creazione dei lotti, secondo le specifiche, 30 pezzi sarebbero restati fuori dalla produzione 9. Decide quindi di contattare Carlo ed esporgli il problema 10. Carlo stabilisce quindi di accorpare quei 30 pezzi in esubero ad un lotto completo, creando quindi un lotto anomalo di 180 pezzi 11. Carlo, o Ciccio, confermano i lotti creati e li inviano al database

Scenario Name Inserimento e lottizzazione degli ordini di un cliente già memorizzato nel Database

Participating actor istances	Pippo : Addetto ufficio clienti Database
Entry condition	1. Pippo lancia il proprio agente; il wrapper (agente del database) si considera sempre attivo
Flow of Events	2. Si sposta nel pannello di immissione degli ordini e seleziona nella casella del committente, tra tutti i committenti memorizzati, la persona con la quale sta trattando per l'ordinazione. 3. Inserisce quindi i dati dell'ordine e conferma l'inserimento cliccando sul bottone apposito della sua interfaccia 4. Attende qualche secondo e riceve il messaggio di conferma di avvenuta memorizzazione dell'ordine nel database

Scenario Name Modifica dei dati dei committenti

Participating actor istances	Pippo : Addetto ufficio clienti Database
Entry condition	1. Pippo lancia il proprio agente; il wrapper (agente del database) si considera sempre attivo
Flow of Events	2. Si sposta nel pannello di visualizzazione dei dati dei clienti salvati nel Database 3. Clicca sull'apposito bottone della sua interfaccia per leggere tali dati. 4. Visualizza i dati ed apporta le modifiche ai campi; N.B. non potrà modificare il nome della ditta relativa al committente in quanto esso lo caratterizza univocamente 5. Al termine dell'operazione clicca sul bottone per salvare tali modifiche nel Database 6. Attende qualche secondo e riceve il messaggio di conferma di avvenuta memorizzazione dell'ordine nel database

Scenario Name Modifica dei dati degli ordini

Participating actor istances	Pippo : Addetto ufficio clienti Database
Entry condition	1. Pippo lancia il proprio agente; il wrapper (agente del database) si considera sempre attivo
Flow of Events	2. Si sposta nel pannello di visualizzazione degli ordini salvati nel Database 3. Clicca sull'apposito bottone della sua interfaccia per leggere tali dati. 4. Visualizza i dati ed apporta le modifiche ai campi; N.B. non sarà possibile modificare gli ordini che sono già stati lottizzati in quanto hanno già cominciato il loro processo produttivo 5. Al termine dell'operazione clicca sul bottone per salvare tali modifiche nel Database 6. Attende qualche secondo e riceve il messaggio di conferma di avvenuta memorizzazione dell'ordine nel database

Scenario Name Cancellazione dei committenti

Participating actor istances	Pippo : Addetto ufficio clienti Database
Entry condition	1. Pippo lancia il proprio agente; il wrapper (agente del database) si considera sempre attivo
Flow of Events	2. Si sposta nel pannello di visualizzazione dei dati dei clienti salvati nel Database 3. Clicca sull'apposito bottone della sua interfaccia per leggere tali dati. 4. Visualizza i dati e seleziona i committenti che devono essere cancellati 5. Al termine dell'operazione clicca sul bottone per cancellare i clienti dal Database 6. Attende qualche secondo e riceve il messaggio di conferma di avvenuta memorizzazione dell'ordine nel database

Scenario Name	Cancellazione degli ordini
Participating actor istances	Pippo : Addetto ufficio clienti Database
Entry condition	1. Pippo lancia il proprio agente; il wrapper (agente del database) si considera sempre attivo
Flow of Events	2. Si sposta nel pannello di visualizzazione degli ordini salvati nel Database 3. Clicca sull'apposito bottone della sua interfaccia per leggere tali dati. 4. Visualizza i dati e seleziona gli ordini che devono essere selezionati (sempre che non siano già stati lottizzati) 5. Al termine dell'operazione clicca sul bottone per cancellare tali ordini dal Database 6. Attende qualche secondo e riceve il messaggio di conferma di avvenuta memorizzazione dell'ordine nel database

Scenario Name	Lottizzazione degli ordini
Participating actor istances	Ciccio : Addetto alla produzione Carlo : Responsabile della produzione Database
Entry condition	1. Ciccio lancia il proprio agente; il wrapper (agente del database) si considera sempre attivo
Flow of Events	2. Nella sua interfaccia visualizza due pannelli con i relativi ordini da schedulare ed inviare rispettivamente agli stabilimenti di competenza (bici da corsa o da trekking) 3. Si sposta in uno dei due pannelli e, supportato dal software e dall'interfaccia grafica, crea i lotti 4. Nel caso di indecisione nella creazione di lotti che non rientrano nelle specifiche di produzione contatta Carlo 5. Carlo supervisiona il lavoro di Ciccio e decide di accorpare determinati lotti e di creare lotti anomali 6. Carlo, o Ciccio, confermano i lotti creati e li inviano al database 7. Attendono qualche secondo e ricevono il messaggio di conferma di avvenuta memorizzazione dell'ordine nel database 8. Si spostano quindi nell'altro pannello e ripetono l'operazione per gli ordini da inviare all'altro stabilimento 9. All'invio dell'ordine ricevono un messaggio di corretto inserimento di entrambi i lotti nel Database per entrambi gli stabilimenti

	10. Completate le operazioni chiudono il programma o lo lasciano in stand-by in attesa di nuovi ordini da lottizzare

Scenario Name	Ricezione e schedulazione dei lotti e visualizzazione della gestione dei messaggi del processo produttivo tra diversi stabilimenti
Participating actor istances	Bobby : Responsabile di stabilimento Ciccio : Addetto alla produzione Database
Entry condition	1. Bobby e Ciccio lanciano ognuno il proprio agente
Flow of Events	2. Bobby visualizza nel suo terminale un elenco dei lotti da schedulare e i lotti gia schedulati in precedenza 3. Procede dunque, mediante supporto grafico, alla schedulazione dei nuovi lotti 4. Ciccio nel frattempo lottizza i nuovi ordini che automaticamente all'avvio del programma ha visualizzato e invia i nuovi lotti allo stabilimento 5. Bobby vede nella sua interfaccia che il bottone "Carica Dati" ha cambiato colore ed è divenuto rosso, segno che nuovi lotti sono stati inseriti 6. Clicca quindi sul corrispondente bottone e provvede allo scaricamento dei dati dei nuovi lotti da schedulare 7. Termina la schedulazione dei lotti e la salva nel database

Scenario Name	Schedulazione con supporto grafico
Participating actor istances	Bobby : Responsabile di stabilimento Database
Entry condition	1. Bobby lancia il proprio agente
Flow of Events	2. Visualizza nel suo terminale un elenco dei lotti da schedulare e i lotti gia schedulati in precedenza 3. Nella sua interfaccia visualizza i dati della schedulazione sia a mezzo testo che con un supporto grafico in cui si evince molto intuitivamente l'andamento temporale della produzione dei lotti 4. Inserisce le date dei nuovi lotti da schedulare o manualmente settando la data, o interagendo col sistema grafico di

	supporto, dal quale evince, tralaltro, la data massima di inizio produzione di ogni singolo lotto N.B. è pure possibile modificare la data di produzione di lotti già schedulati ma non ancora prodotti 5. Al termine della schedulazione clicca sul corrispondente bottone della sua interfaccia e invia i dati al database 6. Attende qualche secondo e riceve il messaggio di conferma di avvenuta memorizzazione dei dati nel database

Scenario Name Produzione e registrazione dei pezzi prodotti

Participating actor istances	Claudio : Operaio addetto alla produzione Database
Entry condition	1. Claudio lancia il proprio agente
Flow of Events	2. Claudio visualizza nel suo terminale un elenco dei lotti già schedulati, letti automaticamente all'avvio dell'agente dal database 3. Tra questi seleziona quelli dei quali il suo stabilimento ha ultimato la produzione e sono pronti per essere consegnati 4. Clicca sull'apposito bottone e conferma l'avvenuta produzione di tali lotti che varrà salvata nel Database 5. Attende qualche secondo e riceve il messaggio di conferma di avvenuta memorizzazione dei dati nel database

Scenario Name Anteprima e stampa delle etichette

Participating actor istances	Claudio : Operaio addetto alla produzione Database
Entry condition	1. Claudio lancia il proprio agente
Flow of Events	2. Claudio visualizza nel suo terminale un elenco dei lotti già schedulati, letti automaticamente all'avvio dell'agente dal database, e va a vidimare la check box corrispondente per confermare l'avvenuta produzione di tali lotti 3. Il software a questo punto provvede alla generazione automatica delle etichette relative ai lotti selezionati da Claudio 4. Claudio visualizza una lista con tutte le etichette da apporre sulle biciclette

	5. Cliccando sugli appositi bottoni può visualizzare un anteprima di stampa di tutte le etichette, ed avviare la stampa 6. Oppure può selezionare singolarmente le etichette create e visualizzarle in dimensioni di stampa reali sul video 7. Una volta stampate tutte le etichette Claudio avrà cura di andarle ad attaccare nei telai delle biciclette

Scenario Name Creazione lista approvvigionamenti

Participating actor istances	Fulvio : Responsabile di magazzino Database
Entry condition	1. Fulvio lancia il proprio agente
Flow of Events	2. Si sposta nel pannello di creazione della lista degli approvvigionamenti 3. Clicca sul bottone "Crea Lista" e visualizza la lista con le quantità da approvvigionare calcolate dal software automaticamente in base al numero di biciclette prodotte 4. Inserisce al lato di ogni componente nell'apposita cella la data di consegna 5. Stampa la distinta e la invia tramite fax ai fornitori

Scenario Name Creazione lista approvvigionamenti e visualizzazione della gestione dei messaggi del processo produttivo tra diversi stabilimenti

Participating actor istances	Fulvio : Responsabile di magazzino Claudio : Operaio addetto alla produzione Database
Entry condition	1. Fulvio e Claudio lanciano ognuno il proprio agente
Flow of Events	2. Fulvio visualizza nel suo terminale due differenti tabelle con le distinte di produzione (componenti e quantità) dei due modelli di biciclette 3. Si accerta della correttezza di tali distinte e si sposta nel pannello relativo alla visualizzazione della lista di approvvigionamento delle materie prime 4. Clicca sul bottone "Crea Lista" e visualizza la lista con le quantità da approvvigionare calcolate dal software automaticamente in base al numero di biciclette prodotte 5. Claudio nel frattempo conferma la produzione di un nuovo

	lotto 6. Fulvio vede nella sua interfaccia che il bottone "Crea Lista" ha cambiato colore ed è divenuto rosso, segno che nuovi lotti sono stati prodotti 7. Clicca quindi nuovamente sul bottone e visualizza la nuova distinta con le nuove quantità da approvvigionare 8. Stampa la distinta e la invia tramite fax ai fornitori

Scenario Name	Creazione lista approvvigionamenti con modifiche alle distinte
Participating actor istances	Fulvio : Responsabile di magazzino Claudio : Operaio addetto alla produzione Database
Entry condition	1. Fulvio e Claudio lanciano ognuno il proprio agente
Flow of Events	2. Fulvio visualizza nel suo terminale due differenti tabelle con le distinte di produzione (componenti e quantità) dei due modelli di biciclette 3. Si accerta della correttezza di tali distinte e si sposta nel pannello relativo alla visualizzazione della lista di approvvigionamento delle materie prime 4. Clicca sul bottone "Crea Lista" e visualizza la lista con le quantità da approvvigionare calcolate dal software automaticamente in base al numero di biciclette prodotte 5. Claudio nel frattempo comunica telefonicamente a Fulvio che negli ultimi lotti prodotti per il modello "bici da corsa"sono stati inseriti dei contachilometri analogici al posto di quelli digitali in numero di 2 su ogni bicicletta 6. Fulvio seleziona allora il pannello con la distinta di produzione del modello "bici da corsa" e modifica tale distinta con l'inserimento dei due contachilometri analogici 7. Si riporta quindi nel pannello di visualizzazione della lista degli approvvigionamenti e clicca sul bottone "Crea lista" 8. La lista viene automaticamente creata dal software con i nuovi dati delle distinte e Fulvio può stamparla ed inviarla ai fornitori

Scenario Name Problemi di caduta della rete

Participating actor istances	Pippo : Addetto ufficio clienti (Valido per qualsiasi altro attore per qualsiasi operazione da qualsiasi terminale)
Entry condition	Database
Flow of Events	1. Pippo lancia il proprio agente 2. Contatta il fornitore e svolge un'operazione (per esempio compila il form di inserimento dei dati di un nuovo ordine) 3. Clicca sul bottone "Salva" e visualizza nella barra di trasmissione della sua interfaccia il messaggio: "Salvataggio dati non riuscito" 4. Riprova a cliccare nuovamente sul bottone e stavolta visualizza il messaggio: "Salvataggio dati avvenuto con successo" 5. Conclusa positivamente l'operazione ritorna alle proprie occupazioni

PROGETTO PASSI

Sistema BikesAgent

Redatto da:	Caico Roberto Gargagliano Giuseppe Termine Francesco	_____	_____
Emesso da:	Caico Roberto Gargagliano Giuseppe Termine Francesco	_____	_____
Approvato da:	Caico Roberto Gargagliano Giuseppe Termine Francesco	_____	_____

Tipo documento:	PP1	.01	.01
	Commessa	*Num. Doc.*	*Vers.*

Prima emissione: 10/04/2004

Distribuzione: Interna

Data di stampa: 18/04/04 18.25

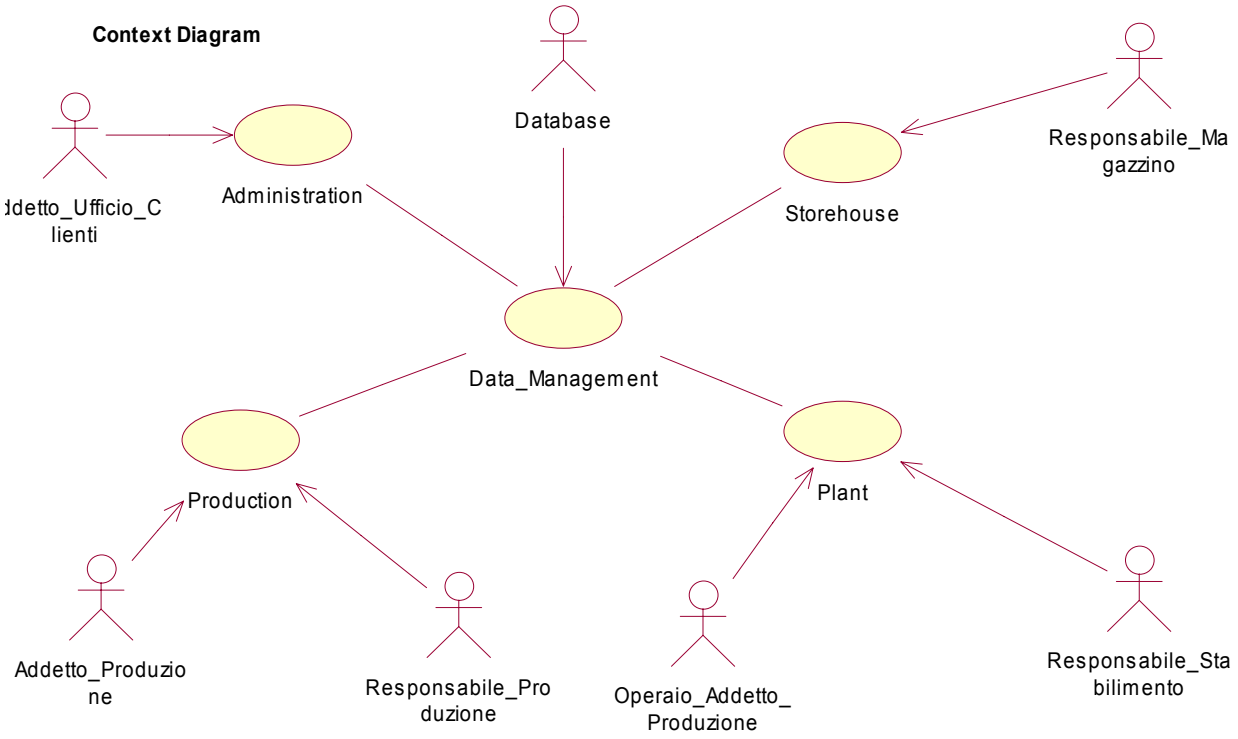
Path del file: *C:\documenti\tesina*

La validità di questa informazione è garantita al momento della stampa del documento.

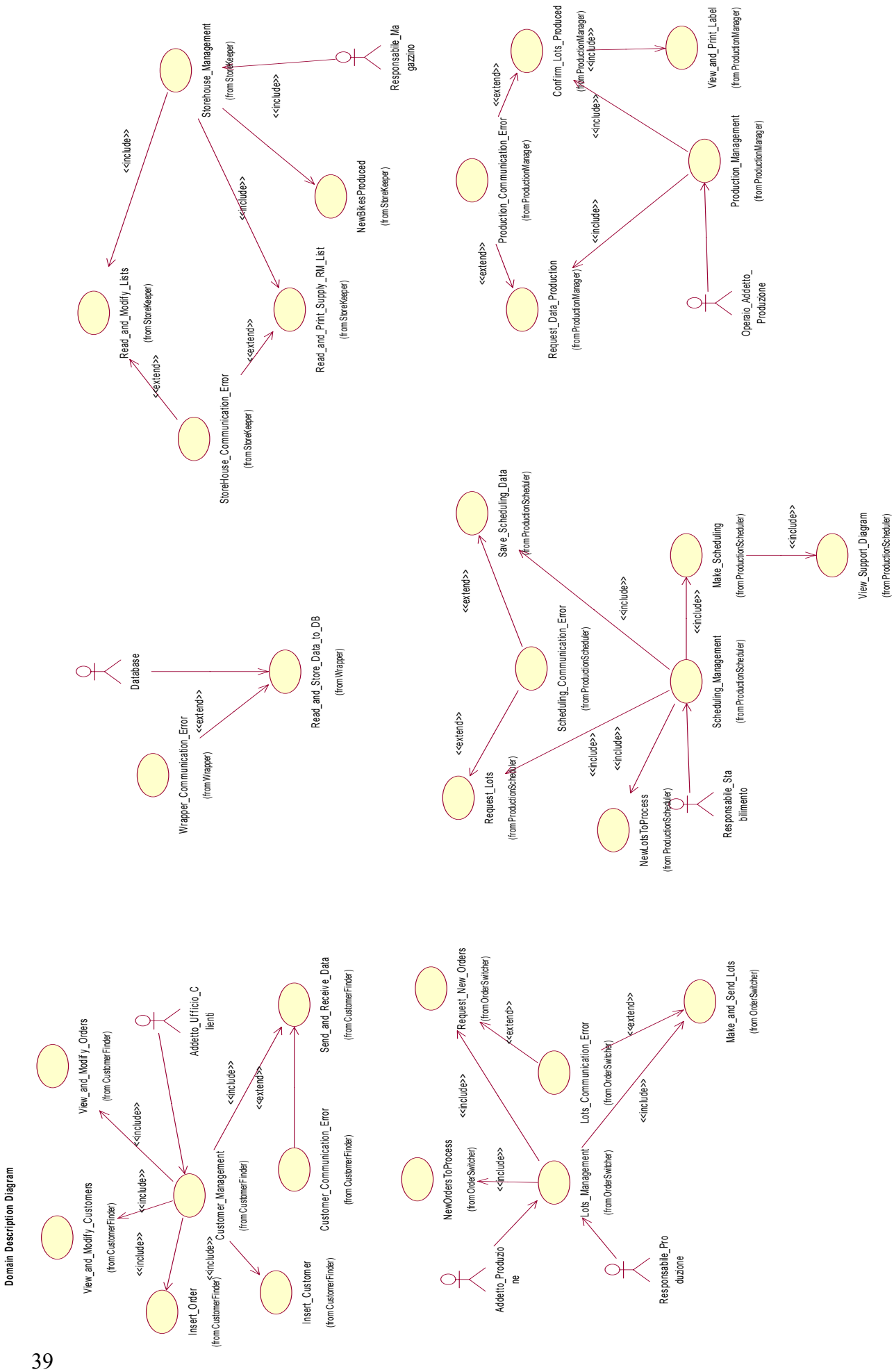
01-Domain Description phase

Descrizione funzionale del sistema a mezzo dei convenzionali diagrammi dei casi d'uso. In questa fase, che scaturisce naturalmente dall'analisi dei requisiti e dall'interazione sistemista-committente, vengono mostrati in due differenti diagrammi le funzionalità generali del sistema e le relazioni generali tra i vari casi d'uso.

Il primo dei due diagrammi mostra con un alto livello di astrazione la suddivisione delle specifiche del progetto e l'interazione degli attori precedentemente individuati con essi; .



Una volta data questa visione generale del progetto vengono scomposti ognuno dei "macro" casi d'uso in una serie di casi d'uso più specifici che mostrano con un maggiore dettaglio la funzionalità che il software permetterà di fare ad ogni attore col quale andrà ad interagire.


Domain Description Diagram
View_and_Modify_Orders
(from CustomerFinder)
View_and_Modify_Customers
(from CustomerFinder)
Addetto_Ufficio_Clienti
Insert_Order
(from CustomerFinder)
Customer_Management
(from CustomerFinder)
Send_and_Receive_Data
(from CustomerFinder)
Customer_Communication_Error
(from CustomerFinder)
Insert_Customer
(from CustomerFinder)
<<include>>
<<extend>>
Request_New_Orders
(from OrderSwitcher)
NewOrdersToProcess
(from OrderSwitcher)
Addetto_Produzione
Lots_Management
(from OrderSwitcher)
Lots_Communication_Error
(from OrderSwitcher)
Make_and_Send_Lots
(from OrderSwitcher)
Responsable_Produzione
Database
Wrapper_Communication_Error
(from Wrapper)
Read_and_Store_Data_to_DB
(from Wrapper)
Read_and_Modify_Lists
(from StoreKeeper)
Storehouse_Management
(from StoreKeeper)
Responsable_Magazzino
StoreHouse_Communication_Error
(from StoreKeeper)
Read_and_Print_Supply_RM_List
(from StoreKeeper)
NewBikesProduced
(from StoreKeeper)
Save_Scheduling_Data
(from ProductionScheduler)
Request_Lots
(from ProductionScheduler)
Scheduling_Communication_Error
(from ProductionScheduler)
NewLotsToProcess
(from ProductionScheduler)
Responsable_Stabilimento
Scheduling_Management
(from ProductionScheduler)
Make_Scheduling
(from ProductionScheduler)
View_Support_Diagram
(from ProductionScheduler)
Production_Communication_Error
(from ProductionManager)
Request_Data_Production
(from ProductionManager)
Confirm_Lots_Produced
(from ProductionManager)
Production_Management
(from ProductionManager)
View_and_Print_Label
(from ProductionManager)
Operaio_Addetto_Produzione

Nella tabella di sotto viene data una spiegazione generale sulla funzionalità di ogni singolo caso d'uso; per la spiegazione completa di ognuno di essi si viene rimandati al paragrafo successivo relativo alla descrizione degli agenti.

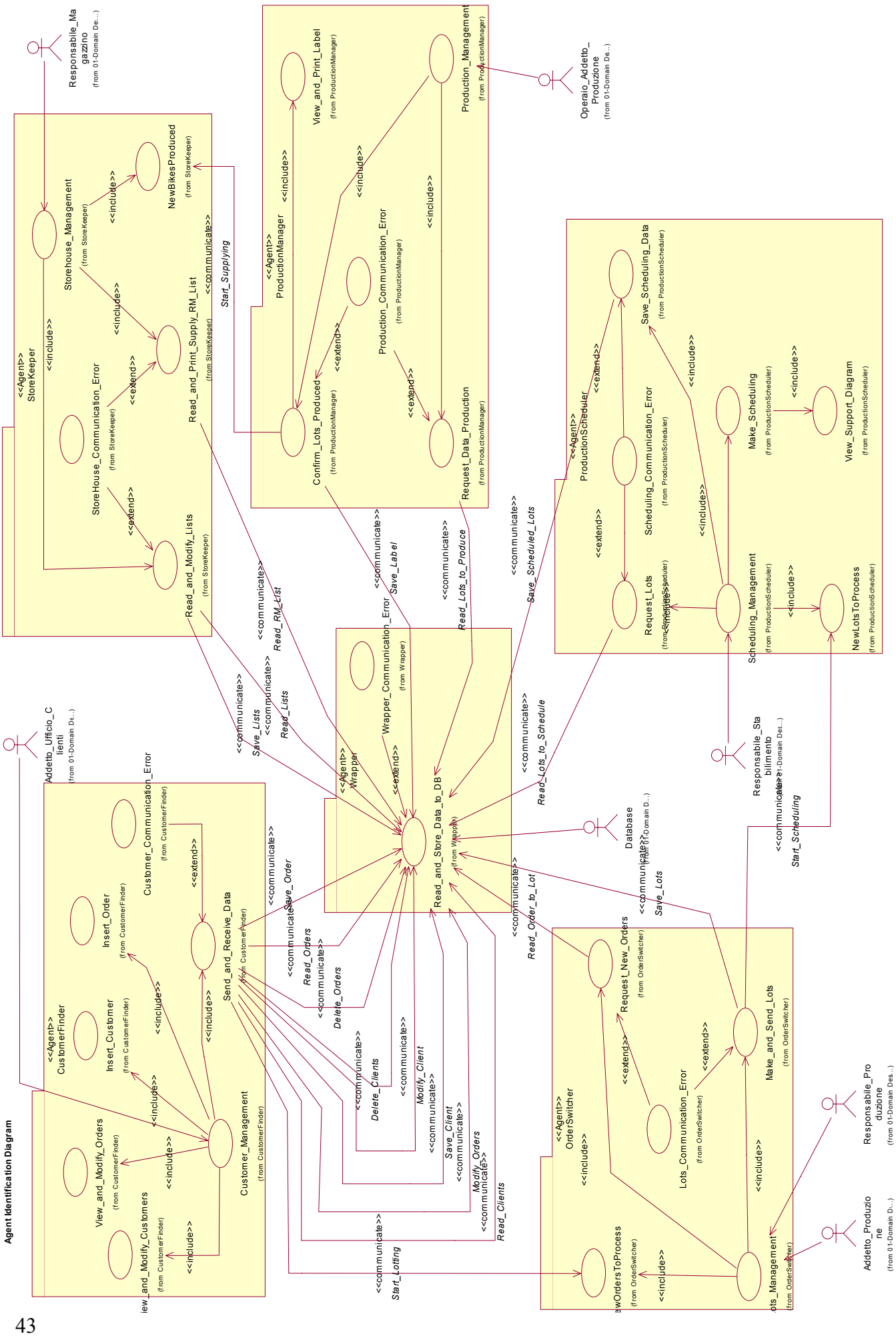
Use Case	Description
Make_Scheduling	L'operatore sceglie le date di inizio produzione dei vari lotti visualizzando ed eventualmente modificando le date di schedulazione degli altri lotti non ancora prodotti
Customer_Management	Gestisce l'inserimento e la modifica dei dati dei committenti e degli ordini facendo interagire l'operatore con una interfaccia grafica dalla quale poter svolgere tutte le operazioni a lui richieste
View_Support_Diagram	Viene visualizzato un diagramma di supporto alla schedulazione
Save_Scheduling_Data	Salva la schedulazione dei lotti nel Database.
NewLotsToProcess	Riceve dall'addetto alla produzione Order Switcher l'informazione che ci sono nuovi lotti da schedulare.
View_and_Modify_Customers	Visualizza i dati di tutti i clienti memorizzati nel Database e ne permette la modifica e/o la cancellazione
Production_Management	Gestisce l'inventario e la registrazione dei pezzi prodotti facendo interagire l'operatore con una interfaccia grafica dalla quale poter svolgere tutte le operazioni a lui richieste.
View_and_Modify_Orders	Visualizza i dati di tutti gli ordini memorizzati nel Database e ne permette la modifica e/o la cancellazione
Customer_Communication_Error	Gestisce gli errori di comunicazione dei dati
Request_Data_Production –	Legge dal Database i lotti già schedulati pronti per essere prodotti..
Send_and_Receive_Data_ Production_Communication_Error	Gestisce il salvataggio e il recupero dei dati degli ordini e dei clienti. Gestisce gli errori di comunicazione.
Insert_Customer	Gestisce la compilazione di un form
Confirm_Lots_Produced	Conferma l'avvenuta produzione dei lotti.
View_and_Print_Label	Visualizza le etichette relative alle biciclette prodotte e ne permette la stampa.
Storehouse_Management	Gestisce la creazione/modifica delle distinte e della lista degli approvvigionamenti facendo interagire l'operatore con una interfaccia grafica dalla quale poter svolgere tutte le operazioni a lui richieste.
Insert_Order	Gestisce la compilazione di un form nell'interfaccia di inserimento dei dati dell'ordine
Lots_Management	Gestisce la creazione e lo smistamento dei lotti facendo interagire l'operatore con una interfaccia grafica dalla quale poter svolgere tutte le operazioni a lui richieste
Read_and_Print_Supply_RM_List	Legge dal Database i dati sul numero delle biciclette prodotte e crea automaticamente la lista degli approvvigionamenti che può essere visualizzata, impaginata e stampata.
Request_New_Orders	Legge dal Database i nuovi ordini inseriti.
Make_and_Send_Lots	Crea i lotti e li invia per la memorizzazione al Database.
Read_and_Modify_Lists	Legge le distinte di produzione dal Database e ne permette la modifica
Lots_Communication_Error StoreHouse_Communication_Error	Gestisce gli errori di comunicazione dei dati dei lotti e dei nuovi ordini da scaricare dal Database
NewBikesProduced	Riceve dal Production Manager l'informazione che sono state prodotte nuove biciclette.
NewOrdersToProcess	Notifica all'utente l'inserimento nel Database di nuovi ordini da processare.
Read_and_Store_Data_to_DB	Gestisce tutte le comunicazioni in entrata ed in uscita per la memorizzazione o l'invio di dati dal Database.
Scheduling_Management	Gestisce la schedulazione dei lotti. facendo interagire l'operatore con una interfaccia grafica dalla quale poter svolgere tutte le operazioni a lui richieste
Wrapper_Communication_Error	Gestisce gli errori di comunicazione e di esecuzione delle query.
Request_Lots	Legge dal Database tutti i lotti inseriti, schedulati e non,che devono essere ancora prodotti..
Scheduling_Communication_Error	Gestisce gli errori di comunicazione dei dati della schedulazione effettuata.

02-Agent Identification phase

Una volta suddivise le varie funzionalità del sistema in diversi casi d'uso in questa fase vengono individuati una serie di package che incorporino logicamente più casi d'uso affini alla elaborazione di un servizio. I nomi di ognuno di questi package individueranno univocamente gli agenti che interagiranno nel sistema e,di riflesso, quali attori interagiranno con quali agenti. Nel diagramma è inoltre possibile individuare delle relazioni tra alcuni casi d'uso con stereotipo *<<comunicate>>;* queste indicano le interazioni sociali tra i vari agenti della società. Da queste interazioni si nota subito come l'agente Wrapper comunichi con tutti e svolga contemporaneamente più comunicazioni differenti con lo stesso agente; di riflesso tutti gli altri agenti comunicheranno col Wrapper ma alcuni di essi comunicheranno con altri agenti per una gestione più rapida ed efficiente della catena produttiva di produzione.
Nella tabella sottostante viene data una presentazione degli agenti individuati che verranno esaminati nei paragrafi successivi più dettagliatamente

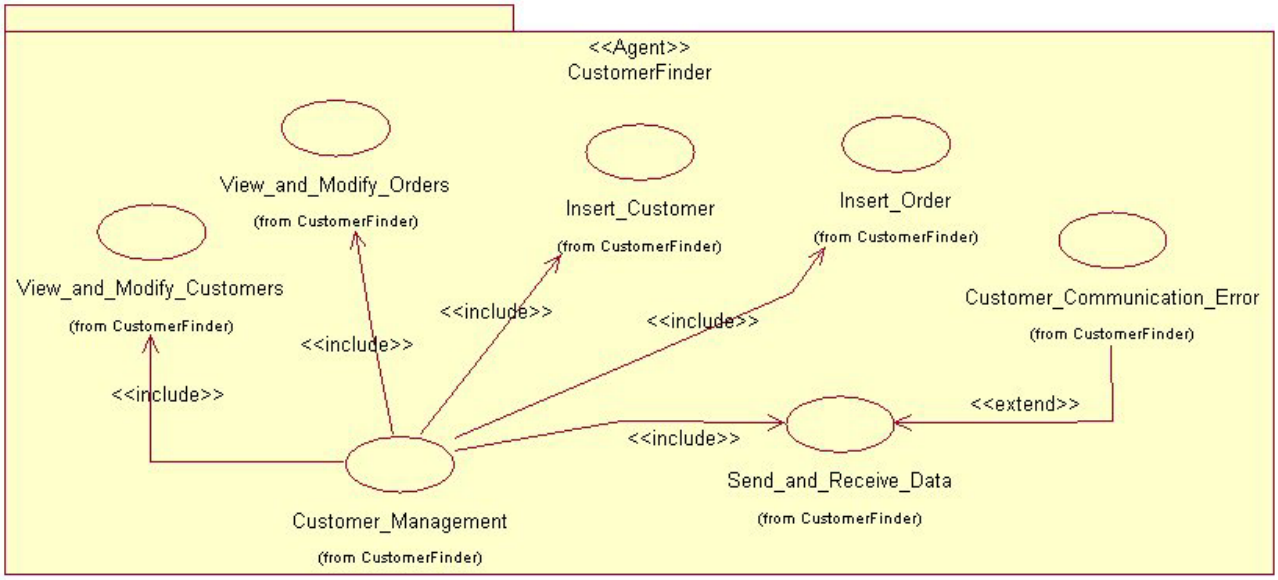
Name Agent	Description
CustomerFinder	Agente che gestisce il settore amministrativo permettendo all'operatore l'inserimento degli ordini e dei dati dei nuovi committenti, e potendo andare a leggere e modificare tali dati (di committenti ed ordini) in qualsiasi momento. Esso comunica col Wrapper per tutte le azioni di memorizzazione e lettura dei dati dal Database, e con l'OrderSwitcher per indicargli che ha nuovo lavoro da svolgere.
OrderSwitcher	Agente che si occupa nella fase amministrativa di leggere dal Database i nuovi ordini inseriti e di dare un supporto all'operatore per la creazione dei lotti di produzione. Esso comunica col Wrapper per tutte le azioni di memorizzazione e lettura dei dati dal Database, con l'OrderSwitcher da cui riceve l'informazione che c'è nuovo lavoro da svolgere e con il ProductionScheduler a cui invia un messaggio di informazione che ha nuovi lotti da schedulare.
ProductionScheduler	Agente che si occupa nella fase produttiva della schedulazione dei lotti da produrre nello stabilimento; viene offerto un sistema di supporto grafico avanzato all'operatore per una maggiore semplicità nella fase di schedulazione. Esso comunica col Wrapper per tutte le azioni di memorizzazione e lettura dei dati dal Database, e con l'OrderSwitcher da cui riceve l'informazione che c'è nuovo lavoro da svolgere.
ProductionManager	Agente che si occupa nella fase produttiva di fornire un supporto grafico all'operatore il quale confermerà ii lotti effettivamente prodotti nello stabilimento e potrà visualizzare e stampare le etichette da applicare ai telai delle biciclette pronte per la consegna. Esso comunica col Wrapper per tutte le azioni di memorizzazione e lettura dei dati dal Database, e con lo StoreKeeper a cui invia un messaggio con

	l'informazione che nuove bici sono state prodotte e dunque occorre approvvigionare i componenti e le materie prime di produzione.
StoreKeeper	Agente che si occupa della gestione del magazzino fornendo all'operatore un supporto grafico che gli permetta di apportare modifiche alle distinte di produzione delle biciclette e di creare e stampare la lista degli approvvigionamenti.
Wrapper	Agente che interagisce direttamente col server dal quale preleva i dati richiestigli dagli altri agenti e salva i nuovi dati inviatigli.


Agent Identification Diagram
Responsabile_Magazzino
(from 01-Domain De...)
Operaio_Addetto_Produzione
(from 01-Domain De..)
Addetto_Ufficio_Clienti
(from 01-Domain De..)
Database
(from 01-Domain D..)
Responsabile_Stabilimento
(from 01-Domain Des..)
Addetto_Produzione
(from 01-Domain D...)
Responsabile_Produzione
(from 01-Domain Des..)
<<Agent>>
StoreKeeper
Storehouse_Management
(from StoreKeeper)
NewBikesProduced
(from StoreKeeper)
StoreHouse_Communication_Error
(from StoreKeeper)
Read_and_Print_Supply_RM_List
(from StoreKeeper)
Read_and_Modify_Lists
(from StoreKeeper)
<<Agent>>
ProductionManager
View_and_Print_Label
(from ProductionManager)
Production_Management
(from ProductionManager)
Production_Communication_Error
(from ProductionManager)
Confirm_Lots_Produced
(from ProductionManager)
Request_Data_Production
(from ProductionManager)
<<Agent>>
ProductionScheduler
Save_Scheduling_Data
(from ProductionScheduler)
Scheduling_Communication_Error
(from ProductionScheduler)
Request_Lots
(from ProductionScheduler)
Make_Scheduling
(from ProductionScheduler)
View_Support_Diagram
(from ProductionScheduler)
Scheduling_Management
(from ProductionScheduler)
NewLotsToProcess
(from ProductionScheduler)
<<Agent>>
Wrapper
Wrapper_Communication_Error
(from Wrapper)
Read_and_Store_Data_to_DB
(from Wrapper)
<<Agent>>
CustomerFinder
Customer_Communication_Error
(from CustomerFinder)
Insert_Order
(from CustomerFinder)
Insert_Customer
(from CustomerFinder)
Send_and_Receive_Data
(from CustomerFinder)
View_and_Modify_Orders
(from CustomerFinder)
Customer_Management
(from CustomerFinder)
View_and_Modify_Customers
(from CustomerFinder)
<<Agent>>
OrderSwitcher
Request_New_Orders
(from OrderSwitcher)
Lots_Communication_Error
(from OrderSwitcher)
Make_and_Send_Lots
(from OrderSwitcher)
NewOrdersToProcess
(from OrderSwitcher)
Lots_Management
(from OrderSwitcher)
<<include>>
<<extend>>
<<communicate>>
Start_Supplying
Read_RM_List
Save_Label
Read_Lots_to_Produce
Save_Lists
Read_Lists
Save_Order
Read_Orders
Delete_Orders
Delete_Clients
Modify_Client
Save_Client
Modify_Orders
Read_Clients
Start_Lotting
Save_Scheduled_Lots
Read_Lots_to_Schedule
Read_Order_to_Lot
Save_Lots
Start_Scheduling

Agent: CustomerFinder

L'agente che si occupa dell'inserimento dei dati degli ordini e dei clienti.



Descrizione dei casi d'uso

Customer Management
Gestisce l'inserimento e la modifica dei dati dei committenti e degli ordini facendo interagire l'operatore con una interfaccia grafica dalla quale poter svolgere tutte le operazioni a lui richieste.

Participating actor	Iniziato dall'Addetto ufficio clienti Comunica col Database
Entry condition	1. L'addetto all'ufficio clienti lancia il programma
Flow of Events	2. Il programma si avvia con un'interfaccia e nel frattempo richiede al Database i dati sui clienti memorizzati 3. L'addetto all'ufficio clienti decide quale dei 4 compiti a lui richiesti svolgere (inserire ordini, visualizzare/modificare/cancellare ordini precedentemente inseriti, inserire clienti, visualizzare/modificare/cancellare clienti precedentemente inseriti) e si sposta nel pannello dell'interfaccia a tale compito relativo 4. Clicca sull'icona e lancia l'attività da svolgere
Exit condition	5. Termina le sue operazioni e chiude il programma
Special requirements	6. La connessione di rete con il Database deve essere attiva e funzionante

Insert Order
Gestisce la compilazione di un form nell'interfaccia di inserimento dei dati dell'ordine

Participating actor	Iniziato dall'Addetto ufficio clienti Comunica col Database
Entry condition	1. L'addetto all'ufficio clienti seleziona il pannello di
Flow of Events	inserimento degli ordini 2. Compila il form di inserimento con i dati richiesti
Exit condition	3. Clicca sul tasto per salvare l'ordine nel Database

Insert Customers
Gestisce la compilazione di un form nell'interfaccia di inserimento dei dati del committente

Participating actor	Iniziato dall'Addetto ufficio clienti Comunica col Database
Entry condition	1. L'addetto all'ufficio clienti seleziona il pannello di
Flow of Events	inserimento dei dati del cliente 2. Compila il form di inserimento con i dati richiesti
Exit condition	3. Clicca sul tasto per memorizzare il cliente nel Database

View_and_Modify_Customers
Visualizza i dati di tutti i clienti memorizzati nel Database e ne permette la modifica e/o la cancellazione.

Participating actor	Iniziato dall'Addetto ufficio clienti Comunica col Database
Entry condition	1. L'addetto all'ufficio clienti seleziona il pannello di
Flow of Events	visualizzazione e modifica dei dati dei clienti 2. Riscrive i campi dei clienti da modificare
Exit condition	3. Clicca sul tasto per salvare le modifiche
Flow of Events	2. Oppure seleziona i clienti da cancellare interamente
Exit condition	3. Clicca sul tasto per cancellare i clienti

View_and_Modify_Orders
Visualizza i dati di tutti gli ordini memorizzati nel Database e ne permette la modifica e/o la cancellazione.

Participating actor	Iniziato dall'Addetto ufficio clienti Comunica col Database
Entry condition	1. L'addetto all'ufficio clienti seleziona il pannello di
Flow of Events	visualizzazione e modifica dei dati degli ordini 2. Riscrive i campi degli ordini da modificare
Exit condition	3. Clicca sul tasto per salvare le modifiche
Flow of Events	2. Oppure seleziona gli ordini da cancellare interamente
Exit condition	3. Clicca sul tasto per cancellare gli ordini

Send_and_Receive_Data
Gestisce il salvataggio e il recupero dei dati memorizzati nel Database.

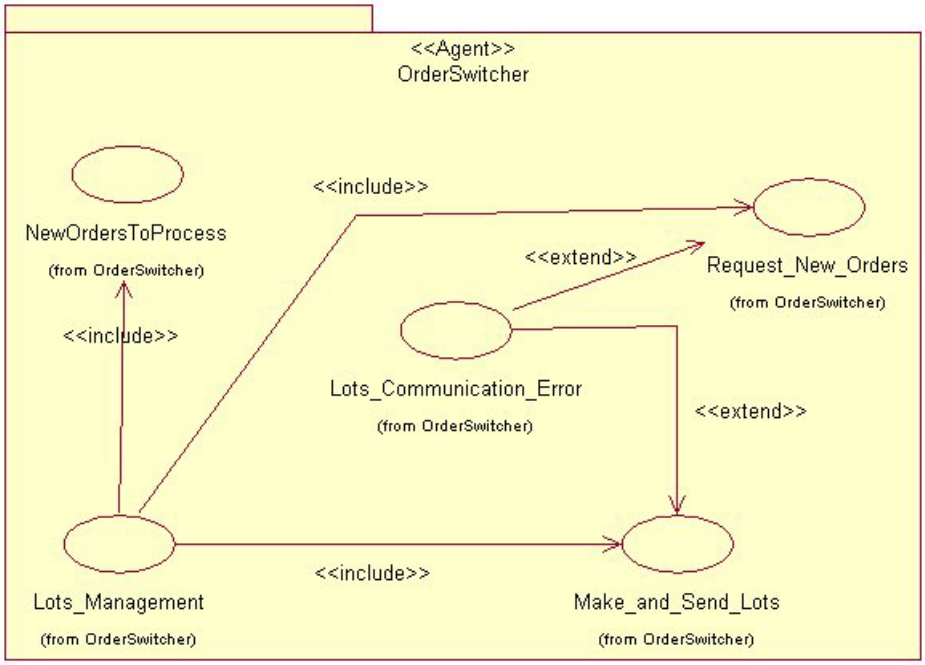
Participating actor	Iniziato dall'Addetto ufficio clienti Comunica col Database
Entry condition	1. L'addetto all'ufficio clienti clicca su uno dei bottoni posti nei vari pannelli dell'interfaccia atti a memorizzare/prelevare dati nel/dal Database. Questo caso d'uso estende il caso d'uso *Customer_Communication_Error*
Flow of Events	2. La richiesta viene interpretata e viene inizializzata la comunicazione con il software atto ad interagire col Database
Exit condition	3. Vengono ricevuti i dati e la comunicazione viene terminata
Special requirements	4. La connessione di rete con il Database deve essere attiva e funzionante

Customer_Communication_Error
Gestisce gli errori di comunicazione.

Entry condition	1. Il *Customer_Communication_Error* estende il caso d'uso *Send_and_Receive_Data* quando la connessione col Database è mancante o interrotta prima del completamento delle operazioni
Flow of Events	2. Viene interpretato il tipo di errore
Exit condition	3. Viene visualizzato un messaggio di errore in interfaccia

Agent: OrderSwitcher

L'agente che si occupa della lottizzazione degli ordini e dell'invio dei lotti agli stabilimenti.



Descrizione dei casi d'uso

Lots_Management
Gestisce la creazione e lo smistamento dei lotti facendo interagire l'operatore con una interfaccia grafica dalla quale poter svolgere tutte le operazioni a lui richieste.

Participating actor	Iniziato dall'Addetto alla produzione o dal Responsabile di produzione Comunica col Database
Entry condition	1. L'addetto alla produzione o il responsabile di produzione lanciano il programma
Flow of Events	2. Il programma si avvia con un'interfaccia e nel frattempo richiede al Database i dati sui nuovi ordini inseriti da lottizzare 3. Avvia il processo di creazione dei lotti cliccando sui bottoni dell'interfaccia
Exit condition	4. Termina le sue operazioni e chiude il programma
Special requirements	5. La connessione di rete con il Database deve essere attiva e funzionante

NewOrdersToProcess
Notifica all'utente l'inserimento nel Database di nuovi ordini da processare.

Participating actor	Iniziato dall'Addetto ufficio clienti Comunica con l'Addetto alla produzione o col Responsabile di produzione
Entry condition	1. L'addetto all'ufficio clienti completa la sua operazione di inserimento dell'ordine e salva i dati, alla fine di tale operazione tramite il caso d'uso *Send_and_Receive_Data* manda un messaggio di notifica al all'addetto alla produzione o al responsabile di produzione
Flow of Events	2. La notifica viene ricevuta e nell'interfaccia vengono visualizzati i bottoni di lettura degli ordini di colore rosso
Exit condition	3. L'Addetto alla produzione o il responsabile di produzione vedendo "accendersi" il bottone gli clicca sopra per processare i nuovi ordini inseriti

Request_new_Orders
Legge dal Database i nuovi ordini inseriti.

Participating actor	Iniziato dall'Addetto alla produzione o dal Responsabile di produzione Comunica col Database
Entry condition	4. L'addetto alla produzione clicca sul bottone per leggere gli ordini dal Database
Flow of Events	5. La richiesta viene interpretata e viene inizializzata la comunicazione con il software atto ad interagire col Database 6. Vengono ricevuti i dati e la comunicazione viene terminata
Exit condition	7. La connessione di rete con il Database deve essere attiva e funzionante

Make_and_Send_Lots
Crea i lotti e li invia per la memorizzazione al Database.

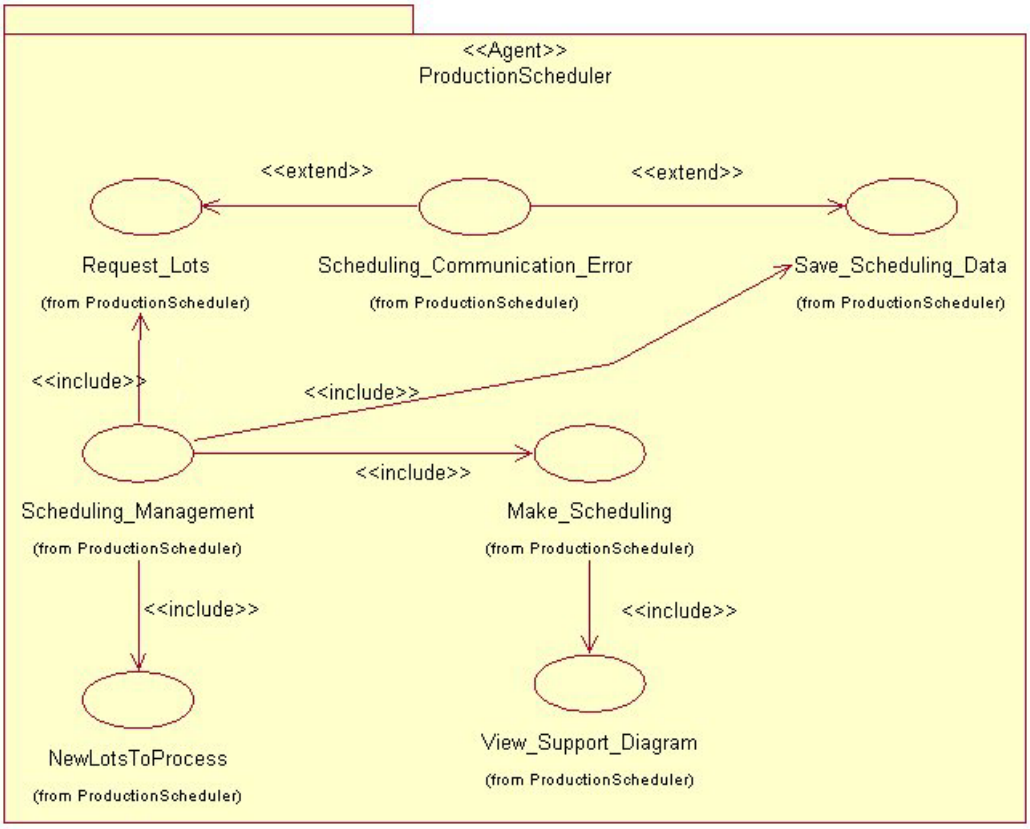
Participating actor	Iniziato dall'Addetto alla produzione o dal Responsabile di produzione Comunica col Database
Entry condition	1. L'addetto alla produzione clicca sul bottone per creare i lotti
Flow of Events	2. Interagisce col software tramite comandi in interfaccia 3. Completa l'operazione di creazione dei lotti di entrambi gli stabilimenti 4. Clicca sul bottone per l'invio di tali dati al Database 5. La richiesta viene interpretata e viene inizializzata la comunicazione con il software atto ad interagire col Database
Exit condition	6. Viene ricevuta una conferma di avvenuta memorizzazione dei dati e la comunicazione viene terminata
Special Requirements	7. La connessione di rete con il Database deve essere attiva e funzionante

Lots_Communication_Error
Gestisce gli errori di comunicazione.

Entry condition	1. Il *Lots_Communication_Error* estende i casi d'uso *Request_new_Orders* e *Make_and_Send_Lots* quando la connessione col Database è mancante o interrotta prima del completamento delle operazioni
Flow of Events	2. Viene interpretato il tipo di errore
Exit condition	3. Viene visualizzato un messaggio di errore in interfaccia

Agent: ProductionScheduler

L'agente che si occupa della schedulazione dei lotti di produzione.



Descrizione dei casi d'uso

Scheduling_Management
Gestisce la schedulazione dei lotti, facendo interagire l'operatore con una interfaccia grafica dalla quale poter svolgere tutte le operazioni a lui richieste.

Participating actor	Iniziato dal Responsabile di stabilimento Comunica col Database
Entry condition	1. Il Responsabile di stabilimento lancia il programma
Flow of Events	2. Il programma si avvia con un'interfaccia e nel frattempo richiede al Database i dati dei lotti da schedulare o già schedulati e non ancora prodotti 3. Il Responsabile di stabilimento interagisce con l'interfaccia e svolge le sue operazioni cliccando sugli appositi bottoni presenti nelle varie schermate
Exit condition	4. Termina le sue operazioni e chiude il programma
Special requirements	5. La connessione di rete con il Database deve essere attiva e funzionante

NewLotsToProcess
Riceve dell'agente OrderSwitcher un messaggio con l'informazione che ci sono nuovi lotti da schedulare.

Participating actor	Iniziato dall'Addetto alla produzione o dal Responsabile di stabilimento
Entry condition	Comunica con il Responsabile di stabilimento
Flow of Events	1. L'addetto alla produzione completa la sua operazione di creazione dei nuovi lotti e li salva nel Database; alla fine di tale operazione tramite il caso d'uso *Make_and_Send_Lots* manda un messaggio di notifica al Responsabile di stabilimento 2. La notifica viene ricevuta e nell'interfaccia viene visualizzato il bottone di lettura dei lotti da schedulare di colore rosso
Exit condition	3. Il Responsabile di stabilimento vedendo "accendersi" il bottone gli clicca sopra per leggere i nuovi lotti

Make_Scheduling
L'operatore sceglie le date di inizio produzione dei vari lotti visualizzando ed eventualmente modificando le date di schedulazione degli altri lotti non ancora prodotti.

Participating actor	Iniziato dal Responsabile di stabilimento
Entry condition	1. Il Responsabile di stabilimento visualizza nella sua interfaccia tutti i lotti schedulati e non che ancora devono essere prodotti
Flow of Events	2. Nell'apposita casella seleziona le date di inizio produzione dei lotti Come supporto a questa operazione viene adoperato il *View_Support_Diagram* caso d'uso
Exit condition	3. Il Responsabile di stabilimento completa l'operazione di schedulazione di tutti i lotti

View_Support_Diagram
Viene visualizzato un diagramma di supporto alla schedulazione.

Participating actor	Iniziato dal Responsabile di stabilimento
Entry condition	1. Il Responsabile di stabilimento visualizza nella sua interfaccia un diagramma che gli mostra in forma grafica l'attuale stato della schedulazione dei lotti
Flow of Events	2. Egli può scegliere se settare la data dalla casella corrispondente o se selezionarla tramite il diagramma direttamente
Exit condition	3. Il Responsabile di stabilimento completa l'operazione di schedulazione di tutti i lotti

Request_Lots
Legge dal Database tutti i lotti inseriti, schedulati e non,che devono essere ancora prodotti.

Participating actor	Iniziato dal Responsabile di stabilimento Comunica col Database
Entry condition	1. Il Responsabile di stabilimento clicca sul bottone per leggere i lotti dal Database
Flow of Events	2. La richiesta viene interpretata e viene inizializzata la comunicazione con il software atto ad interagire col Database
Exit condition	3. Vengono ricevuti i dati e la comunicazione viene terminata
Special Requirement	4. La connessione di rete con il Database deve essere attiva e funzionante

Save_Scheduling_Data
Salva la schedulazione dei lotti nel Database.

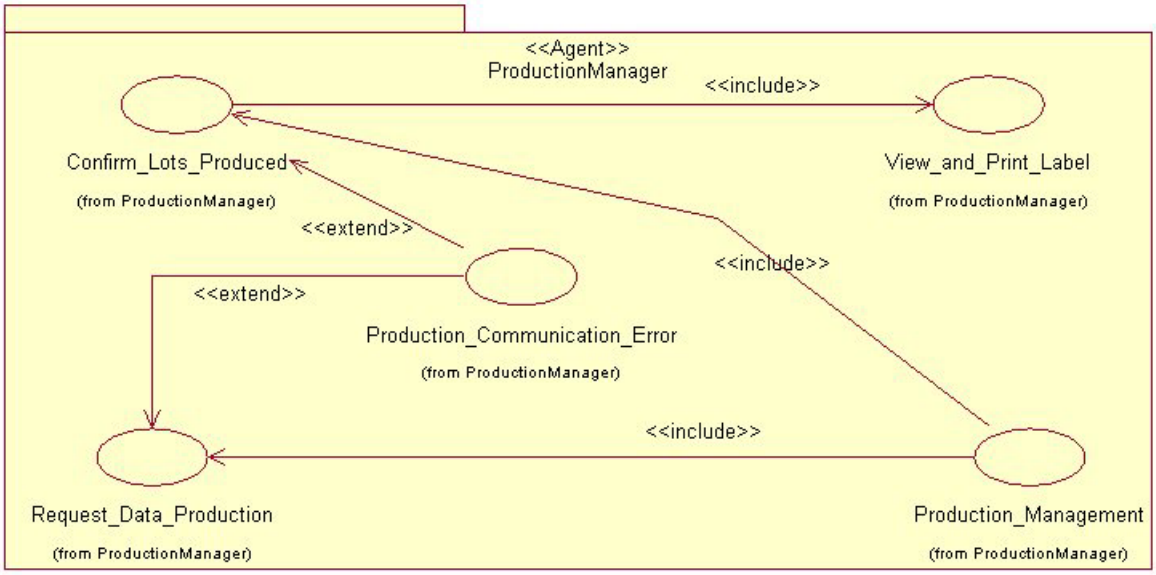
Participating actor	Iniziato dal Responsabile di stabilimento Comunica col Database
Entry condition	1. Il Responsabile di stabilimento clicca sul bottone per l'invio dei dati della schedulazione fatta al Database
Flow of Events	2. La richiesta viene interpretata e viene inizializzata la comunicazione con il software atto ad interagire col Database
Exit condition	3. Vengono ricevuta una conferma di avvenuta memorizzazione dei dati e la comunicazione viene terminata
Special Requirements	4. La connessione di rete con il Database deve essere attiva e funzionante

Scheduling_Communication_Error
Gestisce gli errori di comunicazione.

Entry condition	1. Lo *Scheduling_Communication_Error* estende i casi d'uso *Request_Lots* e *Save_Lots_Data* quando la connessione col Database è mancante o interrotta prima del completamento delle operazioni
Flow of Events	2. Viene interpretato il tipo di errore
Exit condition	3. Viene visualizzato un messaggio di errore in interfaccia

Agent: ProductionManager

L'agente che si occupa della conferma dei lotti prodotti e della creazione delle etichette da apporre nelle biciclette.



Descrizione dei casi d'uso

Production_Management
Gestisce l'inventario e la registrazione dei pezzi prodotti facendo interagire l'operatore con una interfaccia grafica dalla quale poter svolgere tutte le operazioni a lui richieste.

Participating actor	Iniziato dall'Operaio addetto alla produzione Comunica col Database
Entry condition	1. L' Operaio addetto alla produzione lancia il programma
Flow of Events	2. Il programma si avvia con un'interfaccia e nel frattempo richiede al Database i dati sui lotti schedulati in produzione 3. L' Operaio addetto alla produzione interagisce con l'interfaccia e svolge le sue operazioni cliccando sugli appositi bottoni presenti nelle varie schermate
Exit condition	4. Termina le sue operazioni e chiude il programma
Special requirements	5. La connessione di rete con il Database deve essere attiva e funzionante

Request_Data_Production
Legge dal Database i lotti già schedulati pronti per essere prodotti.

Participating actor	Iniziato dall'Operaio addetto alla produzione Comunica col Database
Entry condition	1. L'Operaio addetto alla produzione clicca sul bottone per leggere gli ordini dal Database
Flow of Events	2. La richiesta viene interpretata e viene inizializzata la comunicazione con il software atto ad interagire col Database
Exit condition	3. Vengono ricevuti i dati e la comunicazione viene terminata
Special Requirement	4. La connessione di rete con il Database deve essere attiva e funzionante

Confirm_Lots_Produced
Conferma l'avvenuta produzione dei lotti.

Participating actor	Iniziato dall'Operaio addetto alla produzione Comunica col Database e col Responsabile di magazzino
Entry condition	1. L'Operaio addetto alla produzione seleziona dall'interfaccia le caselle relative al lotto o ai lotti
Flow of Events	la cui produzione è già stata completata nel suo stabilimento
	2. Dopo aver selezionato i lotti, clicca sul bottone per inviare la conferma di avvenuta produzione sia al Database che al software che gestisce l'approvvigionamento delle materie prime
	3. La richiesta viene interpretata e viene inizializzata la comunicazione con il software atto ad interagire col Database o col Responsabile di magazzino
	4. Vengono ricevuti dal Database i dati delle etichette create
Exit condition	5. Cliccando sul bottone dell'interfaccia è possibile visualizzare l'anteprima dell'etichetta o stamparla tramite il caso d'uso *View_and_Print_Label*
Special Requirements	6. La connessione di rete con il Database deve essere attiva e funzionante

View_and_Print_Label
Visualizza le etichette relative alle biciclette prodotte e ne permette la stampa.

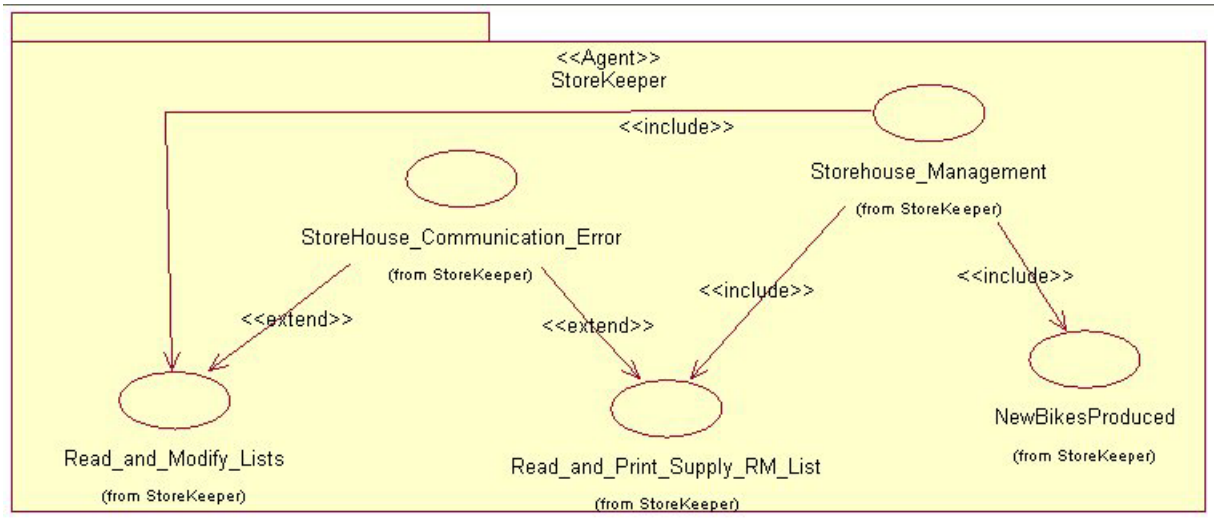
Participating actor	Iniziato dall'Operaio addetto alla produzione
Entry condition	1. L'Operaio addetto alla produzione seleziona dall'interfaccia l'etichetta da stampare e clicca sul
Flow of Events	bottone
	2. Viene aperta un'altra finestra con l'anteprima di stampa dell'etichetta selezionata
	3. L'Operaio può scegliere di stampare l'etichetta
Exit condition	4. L'Operaio chiude la finestra di anteprima dell'etichetta

Production_Communication_Error
Gestisce gli errori di comunicazione.

Entry condition	1. Il *Production_Communication_Error* estende i casi d'uso *Request_Data_Production* e *Confirm_Lots_Produced* quando la connessione col Database è mancante o interrotta prima del completamento delle operazioni
Flow of Events	2. Viene interpretato il tipo di errore
Exit condition	3. Viene visualizzato un messaggio di errore in interfaccia

Agent: StoreKeeper

L'agente che si occupa della conferma dei lotti prodotti e della creazione delle etichette da apporre nelle biciclette.



Descrizione dei casi d'uso

Storehouse_Management
Gestisce la creazione/modifica delle distinte e della lista degli approvvigionamenti facendo interagire l'operatore con una interfaccia grafica dalla quale poter svolgere tutte le operazioni a lui richieste.

Participating actor	Iniziato dal Responsabile di magazzino Comunica col Database
Entry condition	1. Il Responsabile di magazzino lancia il programma
Flow of Events	2. Il programma si avvia con un'interfaccia e nel frattempo richiede al Database i dati delle distinte 3. Il Responsabile di magazzino interagisce con l'interfaccia e svolge le sue operazioni cliccando sugli appositi bottoni presenti nelle varie schermate
Exit condition	4. Termina le sue operazioni e chiude il programma
Special requirements	5. La connessione di rete con il Database deve essere attiva e funzionante

NewBikesProduced
Riceve dal Production Manager l'informazione che sono state prodotte nuove biciclette.

Participating actor	Iniziato dall'Operaio addetto alla produzione Comunica con il Responsabile di magazzino
Entry condition	1. L' Operaio addetto alla produzione completa la sua operazione di conferma dei lotti prodotti e salva i dati; alla fine di tale operazione tramite il caso d'uso *Confirm_Lots_Produced* manda un messaggio di notifica al Responsabile di magazzino
Flow of Events	2. La notifica viene ricevuta e nell'interfaccia viene visualizzato il bottone di creazione della lista degli approvvigionamenti delle materie prime di colore rosso
Exit condition	3. Il Responsabile di magazzino vedendo il cambiamento di colore del bottone, clicca su di esso per creare la lista e procedere con la stampa e l'invio ai fornitori

Read_and_Print_Supply_RM_List
Legge dal Database i dati sul numero delle biciclette prodotte e crea automaticamente la lista degli approvvigionamenti che può essere visualizzata, impaginata e stampata.

Participating actor	Iniziato dal Responsabile di magazzino Comunica col Database
Entry condition	1. Il Responsabile di magazzino clicca sul bottone per leggere i dati delle biciclette prodotte dal Database.
Flow of Events	2. La richiesta viene interpretata e viene inizializzata la comunicazione con il software atto ad interagire col Database 3. Vengono ricevuti i dati e la comunicazione viene terminata 4. La lista degli approvvigionamenti viene creata automaticamente dal software e mostrata a video al Responsabile di magazzino
Exit condition	5. Egli provvede quindi all'impaginazione, alla stampa e invia la lista ai fornitori
Special Requirement	6. La connessione di rete con il Database deve essere attiva e funzionante

Read_and_Modify_List
Legge le distinte di produzione dal Database e ne permette la modifica.

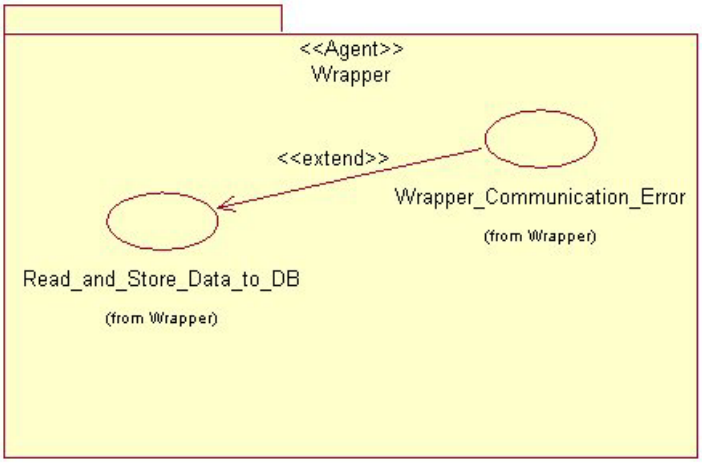
Participating actor	Iniziato dal Responsabile di magazzino Comunica col Database
Entry condition	1. Il Responsabile di magazzino clicca sul bottone per leggere le distinte di produzione dal Database.
Flow of Events	2. La richiesta viene interpretata e viene inizializzata la comunicazione con il software atto ad interagire col Database 3. Vengono ricevuti i dati e la comunicazione viene terminata 4. Le distinte vengono visualizzate nel monitor del terminale del Responsabile di magazzino
Exit condition	5. Egli provvede ad apportare le modifiche e clicca sul bottone per salvare le distinte modificate
Special Requirement	6. La connessione di rete con il Database deve essere attiva e funzionante

Storehouse_Communication_Error
Gestisce gli errori di comunicazione.

Entry condition	1. Lo *Storehouse_Communication_Error* estende i casi d'uso *Read_and_Modify_Lists* e *Read_and_Print_Supply_RM_List* quando la connessione col Database è mancante o interrotta prima del completamento delle operazioni
Flow of Events	2. Viene interpretato il tipo di errore
Exit condition	3. Viene visualizzato un messaggio di errore in interfaccia

Agent: Wrapper

L'agente che si interfaccia al database.



Descrizione dei casi d'uso

Read_and_Store_Data_to_DB
Gestisce tutte le comunicazioni in entrata ed in uscita per la memorizzazione o l'invio di dati dal Database.

Participating actor	Comincia ogni qual volta qualsiasi altro agente comunica con esso Comunica col Database
Entry condition	1. Un agente richiede un servizio di lettura/scrittura dati nel Database
Flow of Events	2. La richiesta viene interpretata e vengono memorizzati o prelevati i dati richiesti
Exit condition	3. Viene inviata una comunicazione di ritorno con l'esito dell'operazione svolta
Special Requirements	4. La connessione di rete con il Database deve essere attiva e funzionante

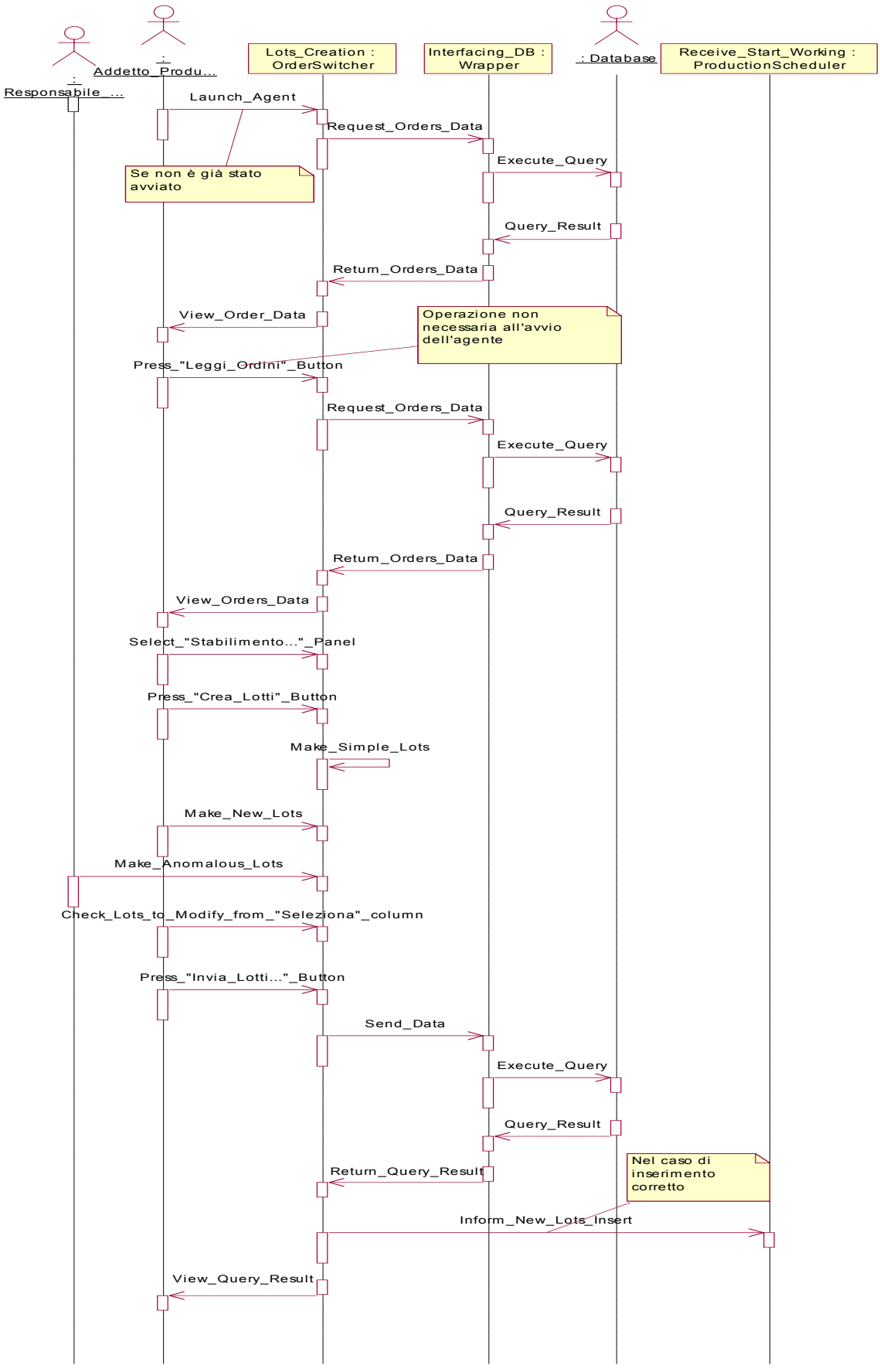
Wrapper_Communication_Error
Gestisce gli errori di comunicazione e di esecuzione delle query.

Entry condition	1. Il *Wrapper_Communication_Error* estende il caso d'uso *Read_and_Store_Data_to_DB* quando la connessione col Database è mancante o interrotta prima del completamento delle operazioni e quando si verificano errori SQL nell'esecuzione delle query
Flow of Events	2. Viene interpretato il tipo di errore
Exit condition	3. Viene inviata una comunicazione all'agente che ha richiesto il servizio con un messaggio di errore

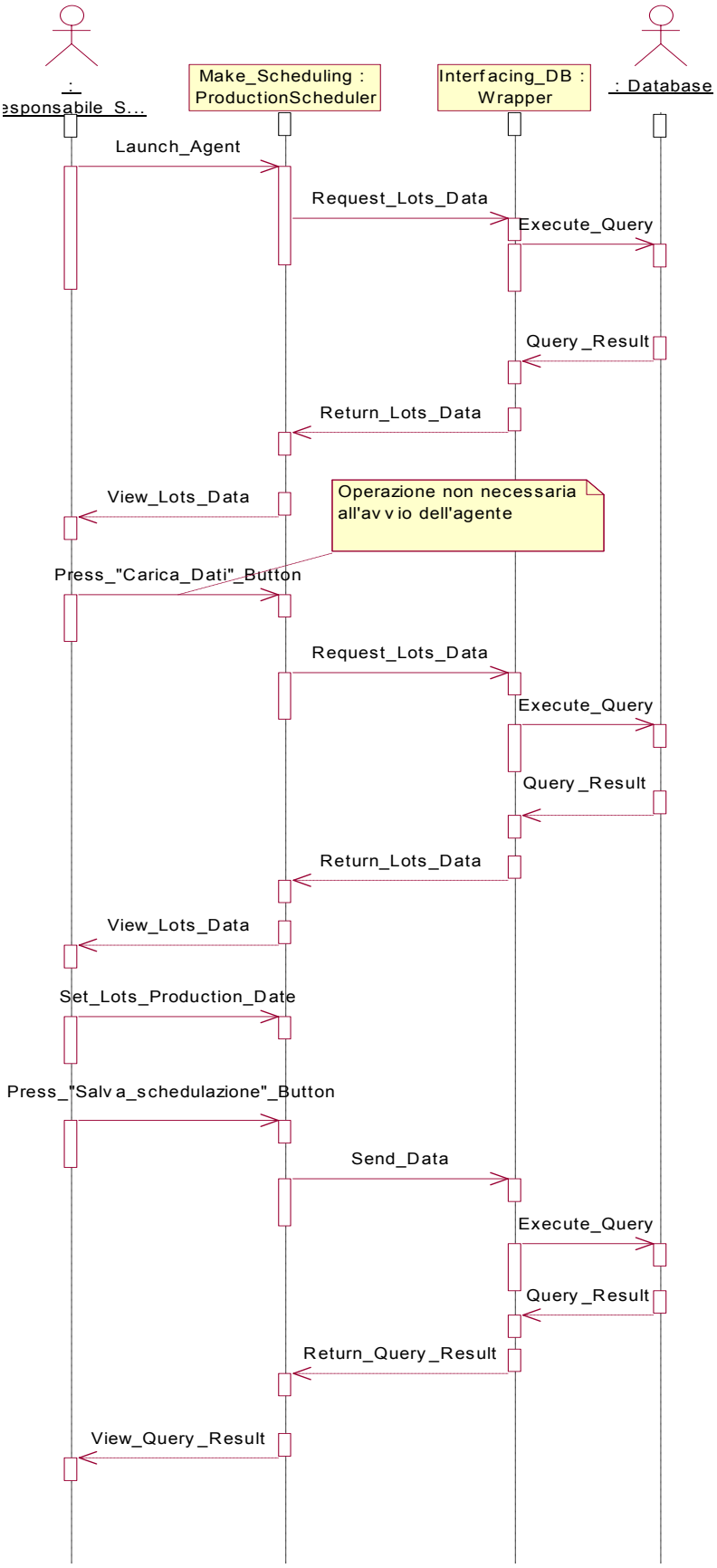
03-Role Identification phase

Una volta identificati gli agenti si comincia da questa fase a costruire quella che sarà alla fine del progetto la società di agenti che, interagendo tra di essi, permetteranno di offrire agli utenti tutti i vari servizi. Per questo motivo vengono individuati dei concetti sociali che caratterizzeranno ogni agente, i ruoli, e vengono realizzati dei diagrammi di sequenza che permettono di esprimere in maniera molto intuitiva tutti i più importanti scenari di utilizzo del software, evidenziando l'interazione tra i ruoli dei vari agenti e le comunicazioni tra di essi. In questa fase la descrizione dei ruoli non verrà fatta molto in dettaglio in quanto ogni ruolo verrà formalizzato nella fase del RDP.

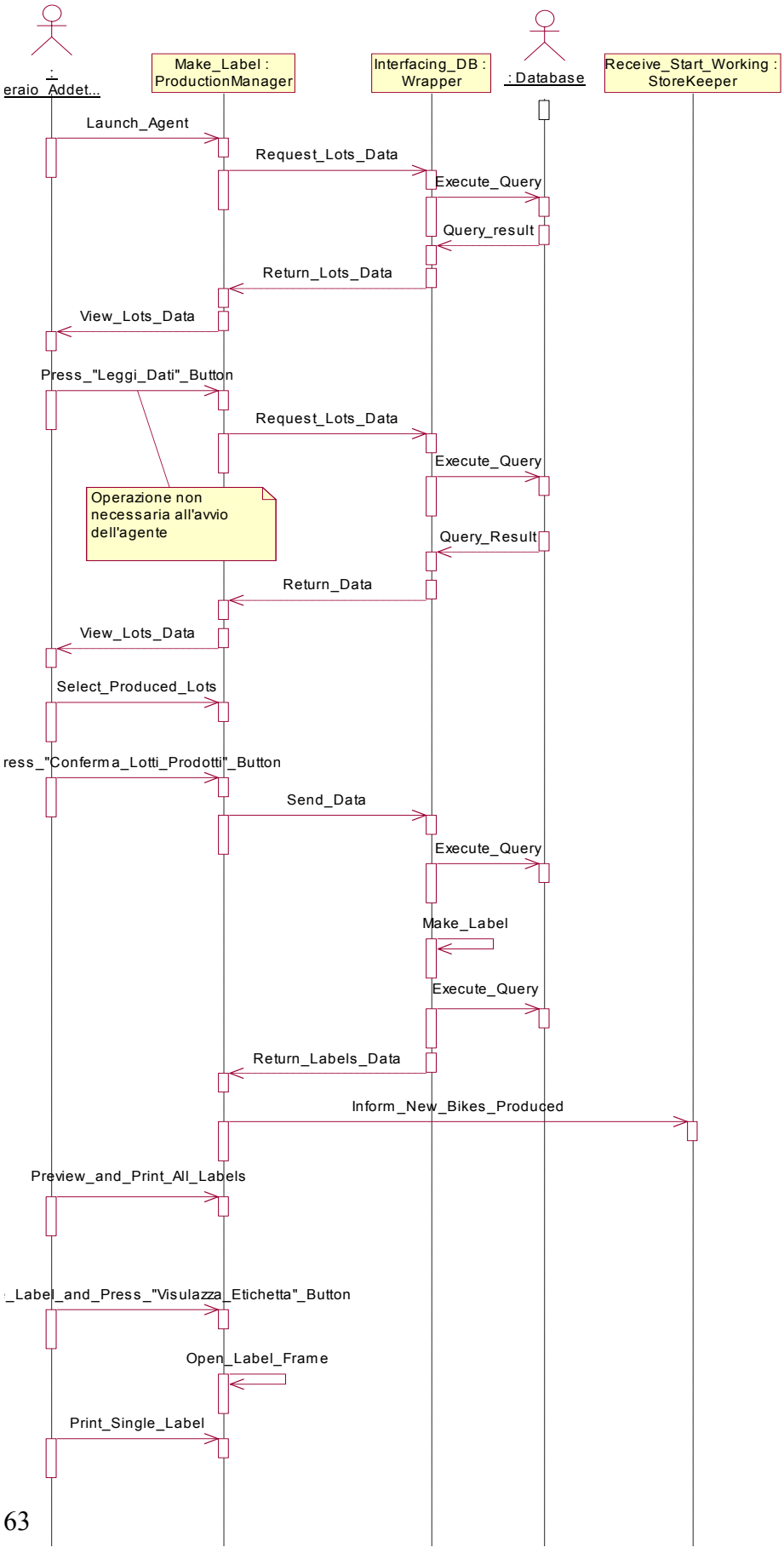
Make_Lots


: Responsabile_...
: Addetto_Produ...
Lots_Creation : OrderSwitcher
Interfacing_DB : Wrapper
: Database
Receive_Start_Working : ProductionScheduler
Launch_Agent
Request_Orders_Data
Execute_Query
Se non è già stato avviato
Query_Result
Return_Orders_Data
View_Order_Data
Operazione non necessaria all'avvio dell'agente
Press_"Leggi_Ordini"_Button
Request_Orders_Data
Execute_Query
Query_Result
Return_Orders_Data
View_Orders_Data
Select_"Stabilimento..."_Panel
Press_"Crea_Lotti"_Button
Make_Simple_Lots
Make_New_Lots
Make_Anomalous_Lots
Check_Lots_to_Modify_from_"Seleziona"_column
Press_"Invia_Lotti..."_Button
Send_Data
Execute_Query
Query_Result
Return_Query_Result
Nel caso di inserimento corretto
Inform_New_Lots_Insert
View_Query_Result

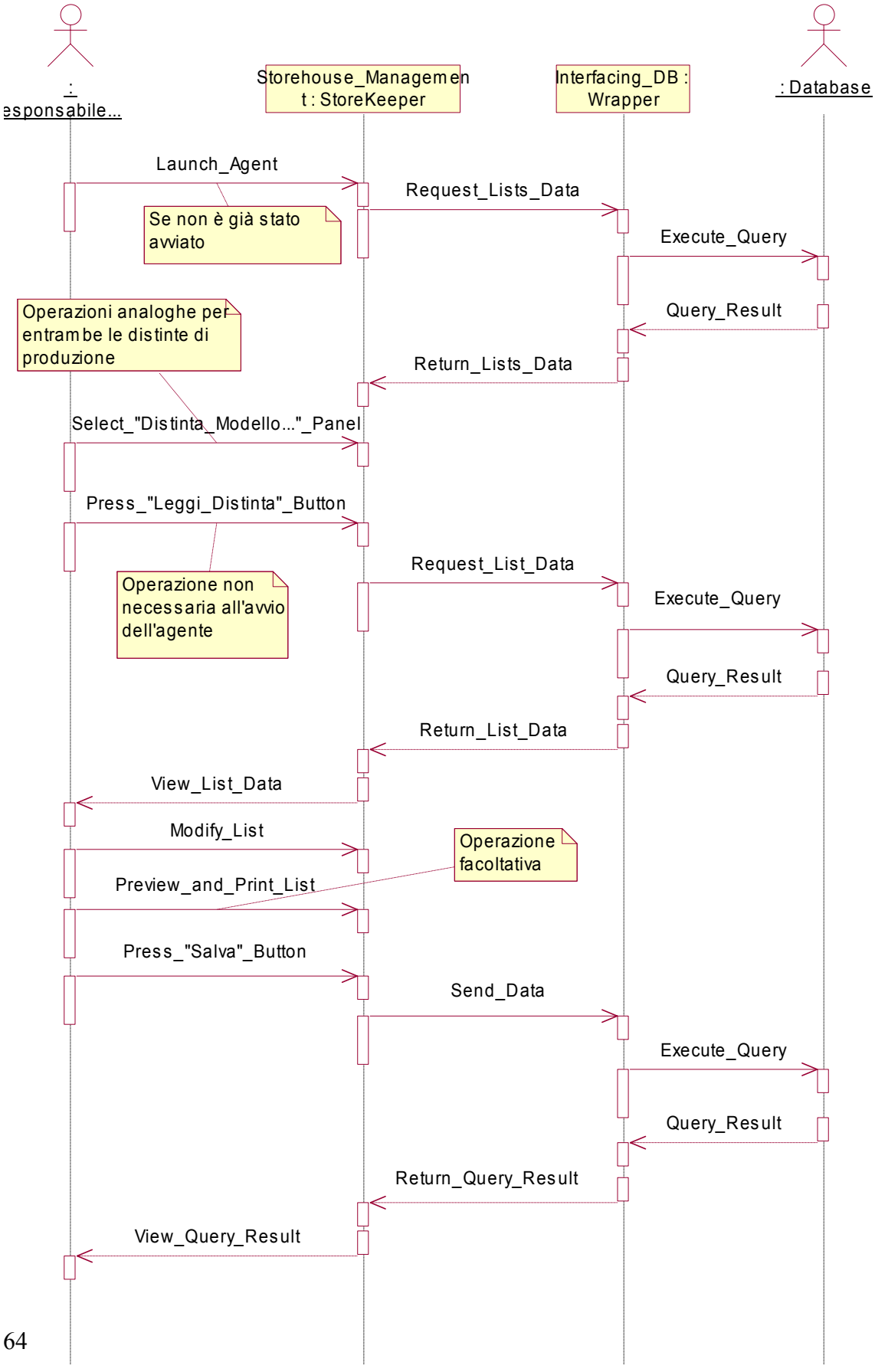
Scheduling


:
esponsabile_S...
Make_Scheduling :
ProductionScheduler
Interfacing_DB :
Wrapper
: Database
Launch_Agent
Request_Lots_Data
Execute_Query
Query_Result
Return_Lots_Data
View_Lots_Data
Operazione non necessaria
all'avvio dell'agente
Press_"Carica_Dati"_Button
Request_Lots_Data
Execute_Query
Query_Result
Return_Lots_Data
View_Lots_Data
Set_Lots_Production_Date
Press_"Salva_schedulazione"_Button
Send_Data
Execute_Query
Query_Result
Return_Query_Result
View_Query_Result

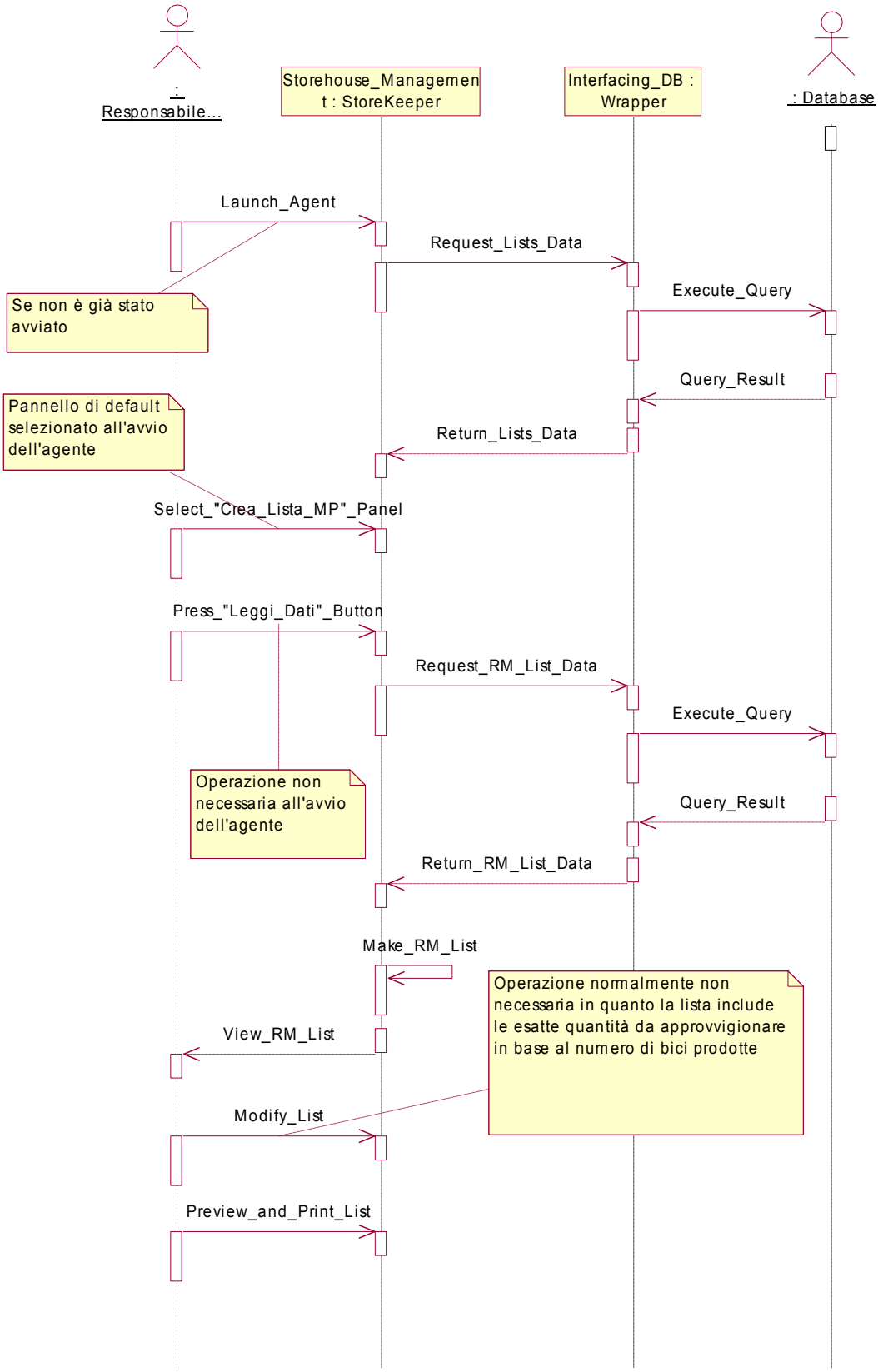
Label_Management



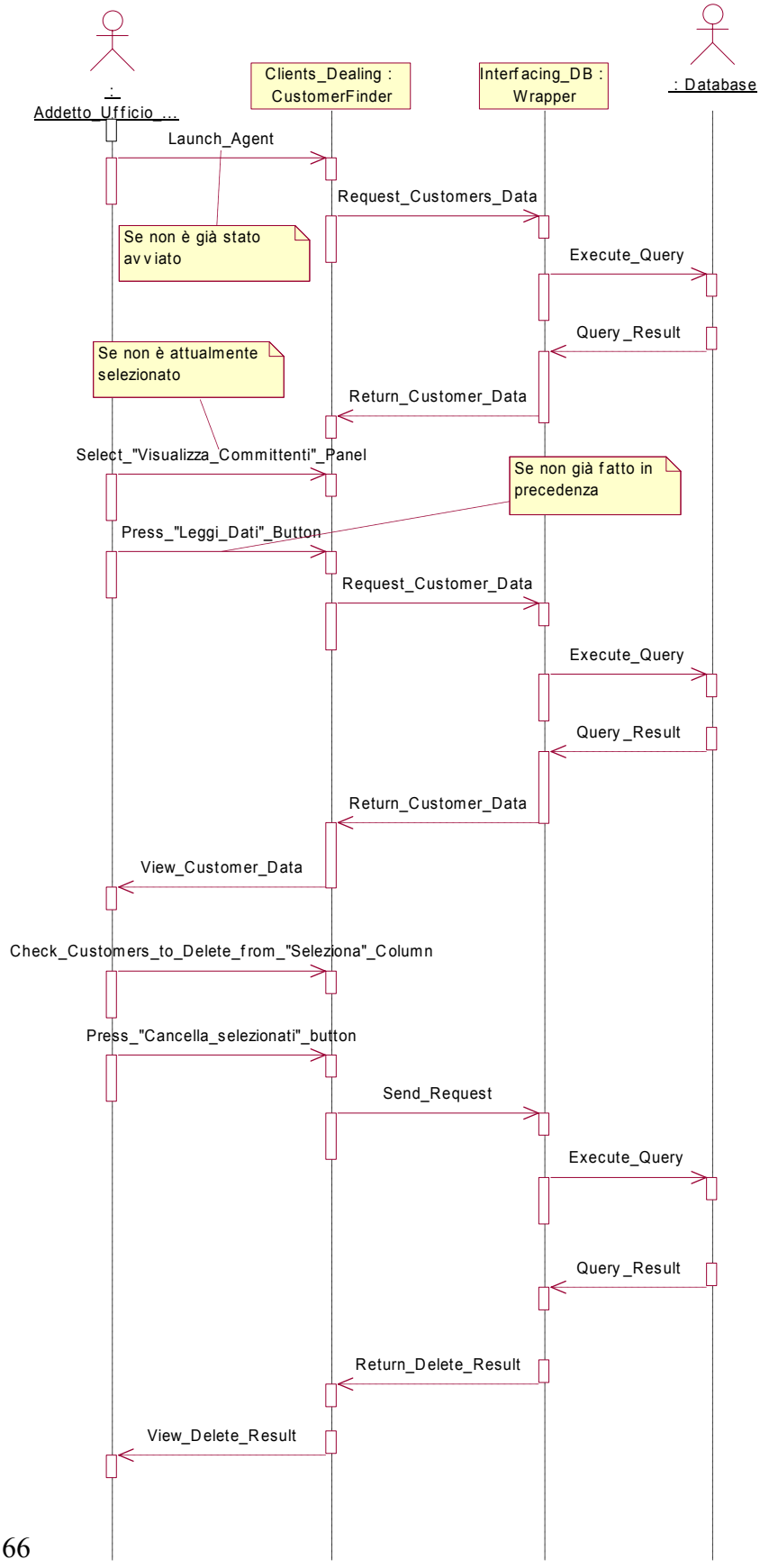
Modify_list


: esponsabile...
Storehouse_Management : StoreKeeper
Interfacing_DB : Wrapper
: Database
Launch_Agent
Se non è già stato avviato
Request_Lists_Data
Execute_Query
Query_Result
Operazioni analoghe per entrambe le distinte di produzione
Return_Lists_Data
Select_"Distinta_Modello..."_Panel
Press_"Leggi_Distinta"_Button
Operazione non necessaria all'avvio dell'agente
Request_List_Data
Execute_Query
Query_Result
Return_List_Data
View_List_Data
Modify_List
Operazione facoltativa
Preview_and_Print_List
Press_"Salva"_Button
Send_Data
Execute_Query
Query_Result
Return_Query_Result
View_Query_Result

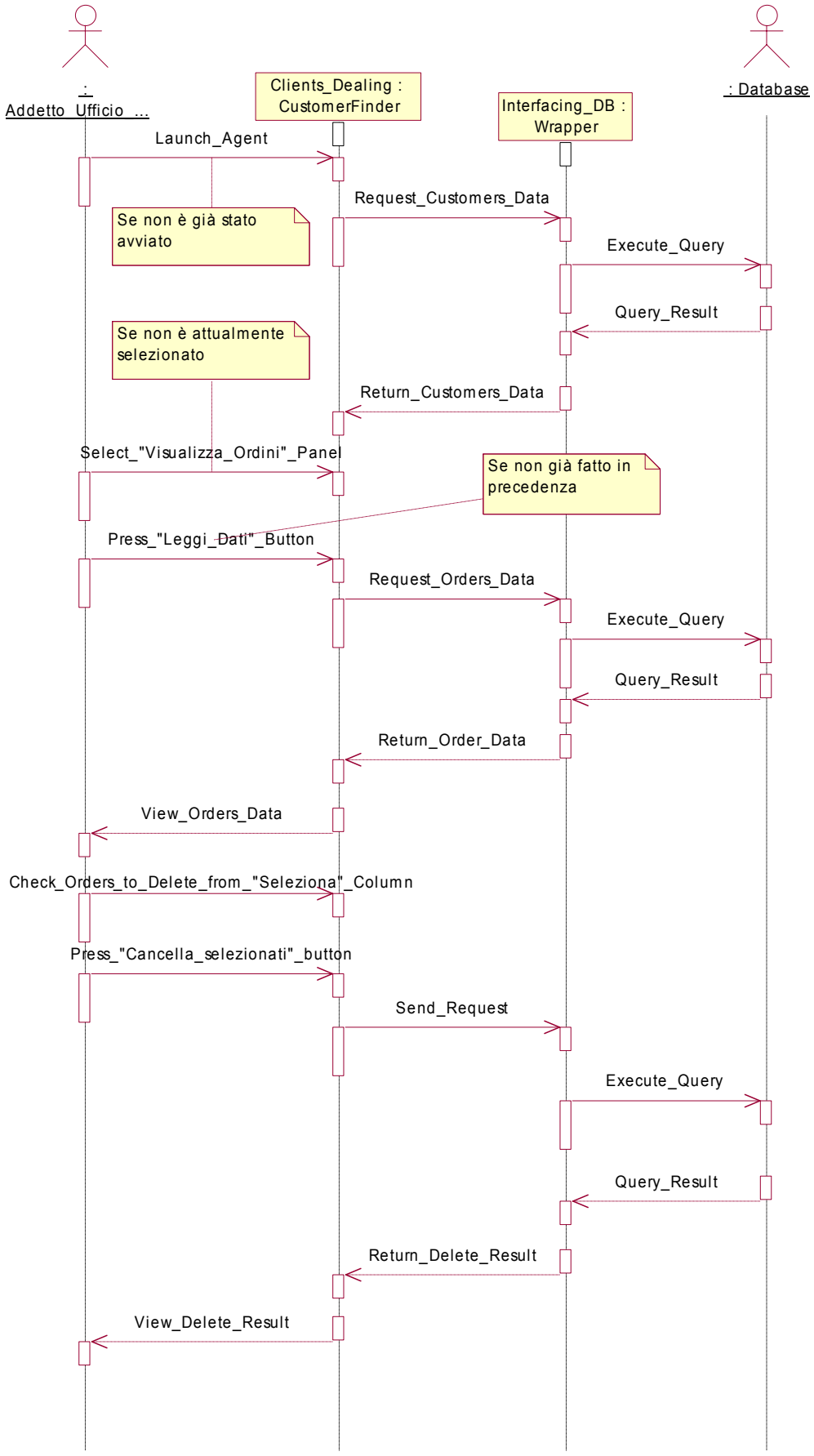
Make_RM_List


: Responsabile...
Storehouse_Management : StoreKeeper
Interfacing_DB : Wrapper
: Database
Launch_Agent
Request_Lists_Data
Execute_Query
Query_Result
Return_Lists_Data
Se non è già stato avviato
Pannello di default selezionato all'avvio dell'agente
Select_"Crea_Lista_MP"_Panel
Press_"Leggi_Dati"_Button
Request_RM_List_Data
Execute_Query
Query_Result
Return_RM_List_Data
Operazione non necessaria all'avvio dell'agente
Make_RM_List
Operazione normalmente non necessaria in quanto la lista include le esatte quantità da approvvigionare in base al numero di bici prodotte
View_RM_List
Modify_List
Preview_and_Print_List

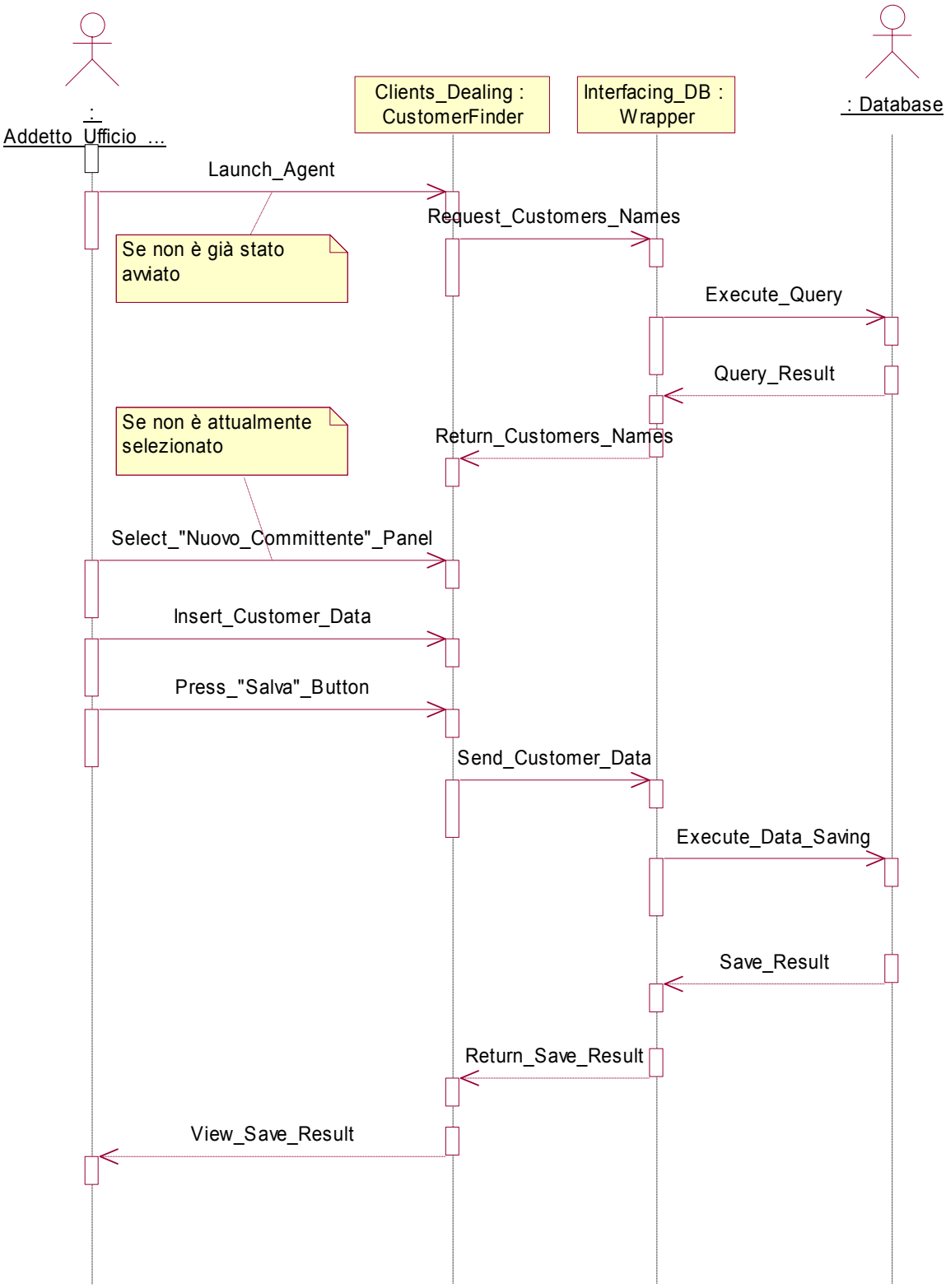
Delete_Clients



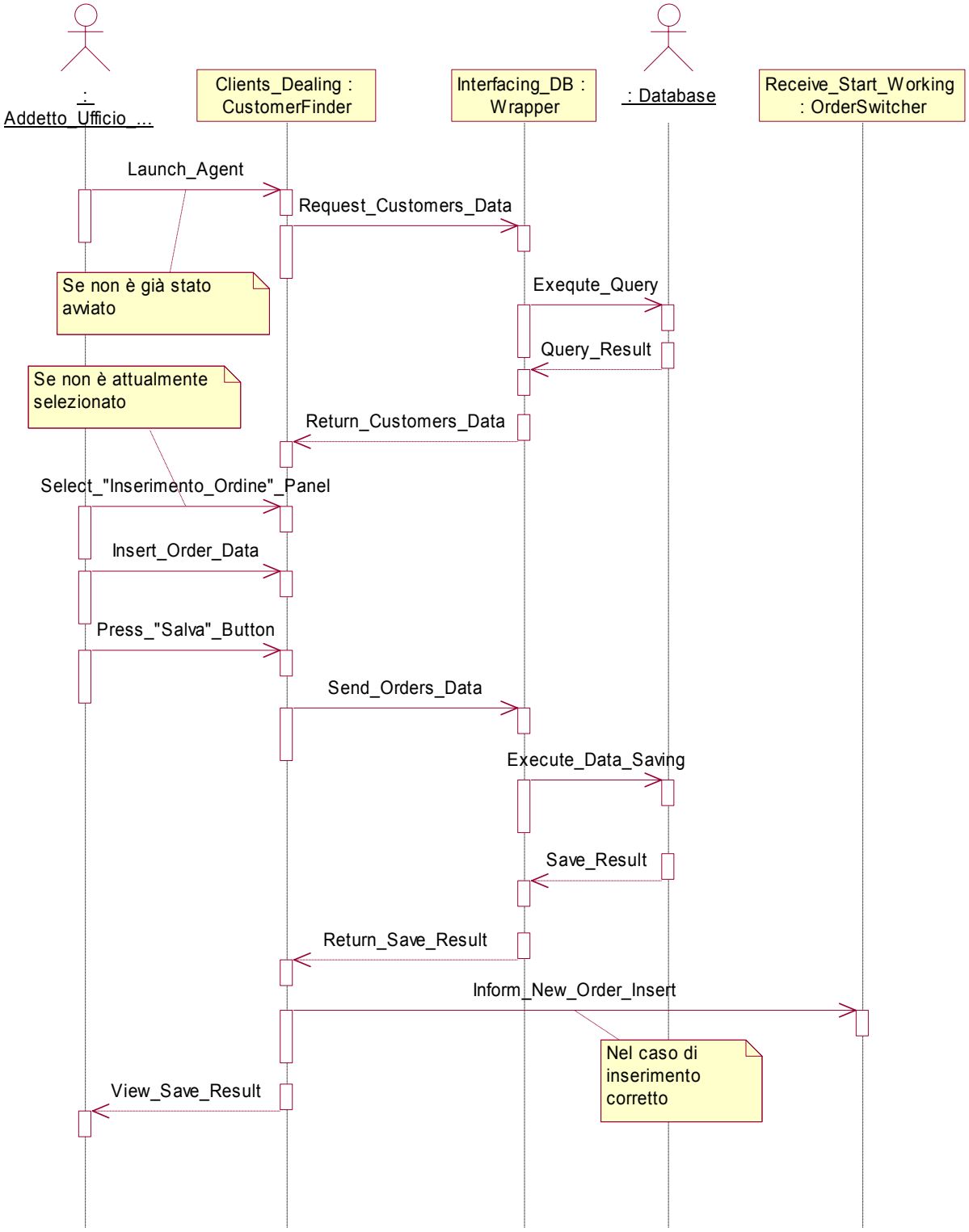
Delete_Orders



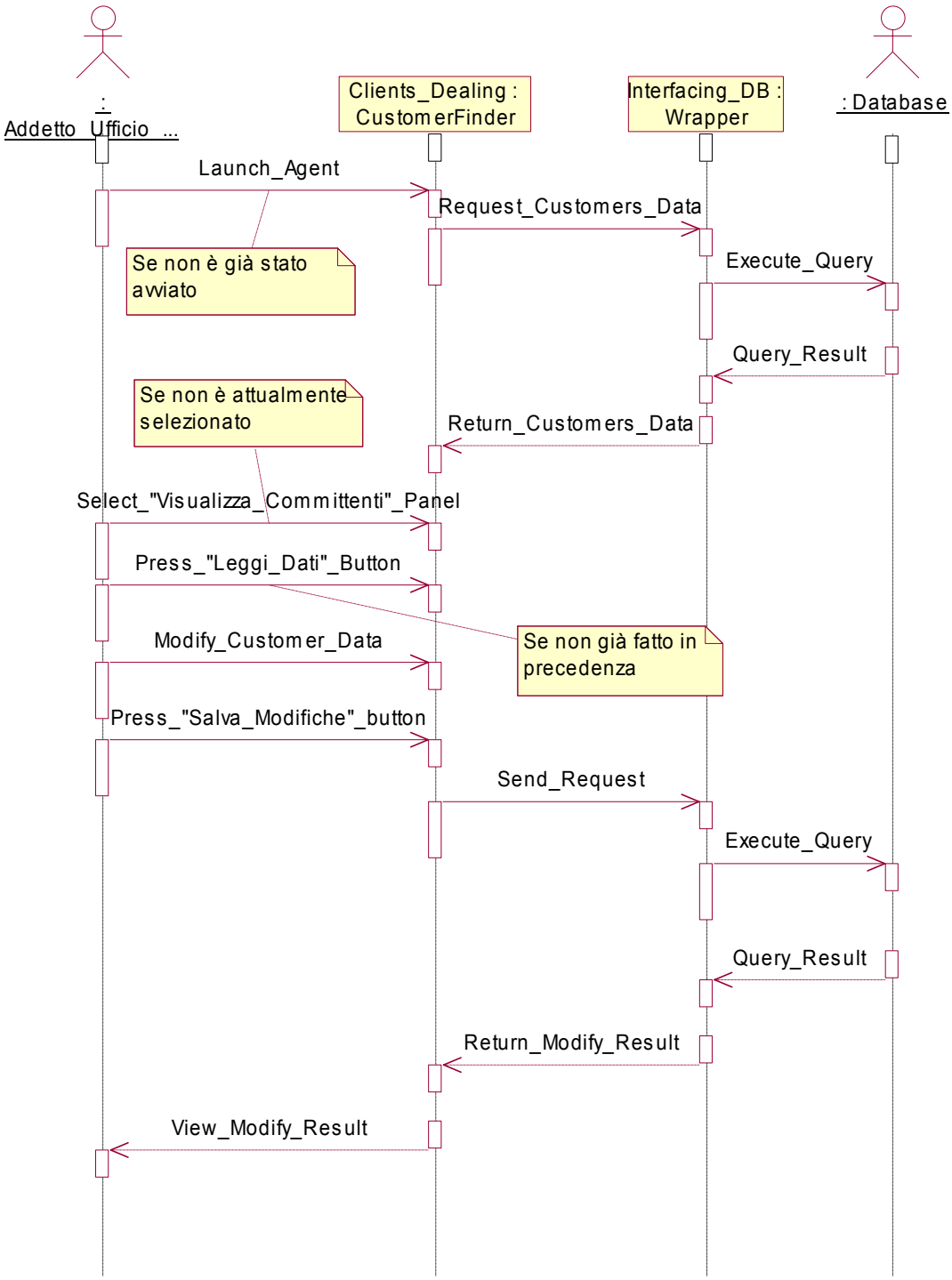
Insert_Customer



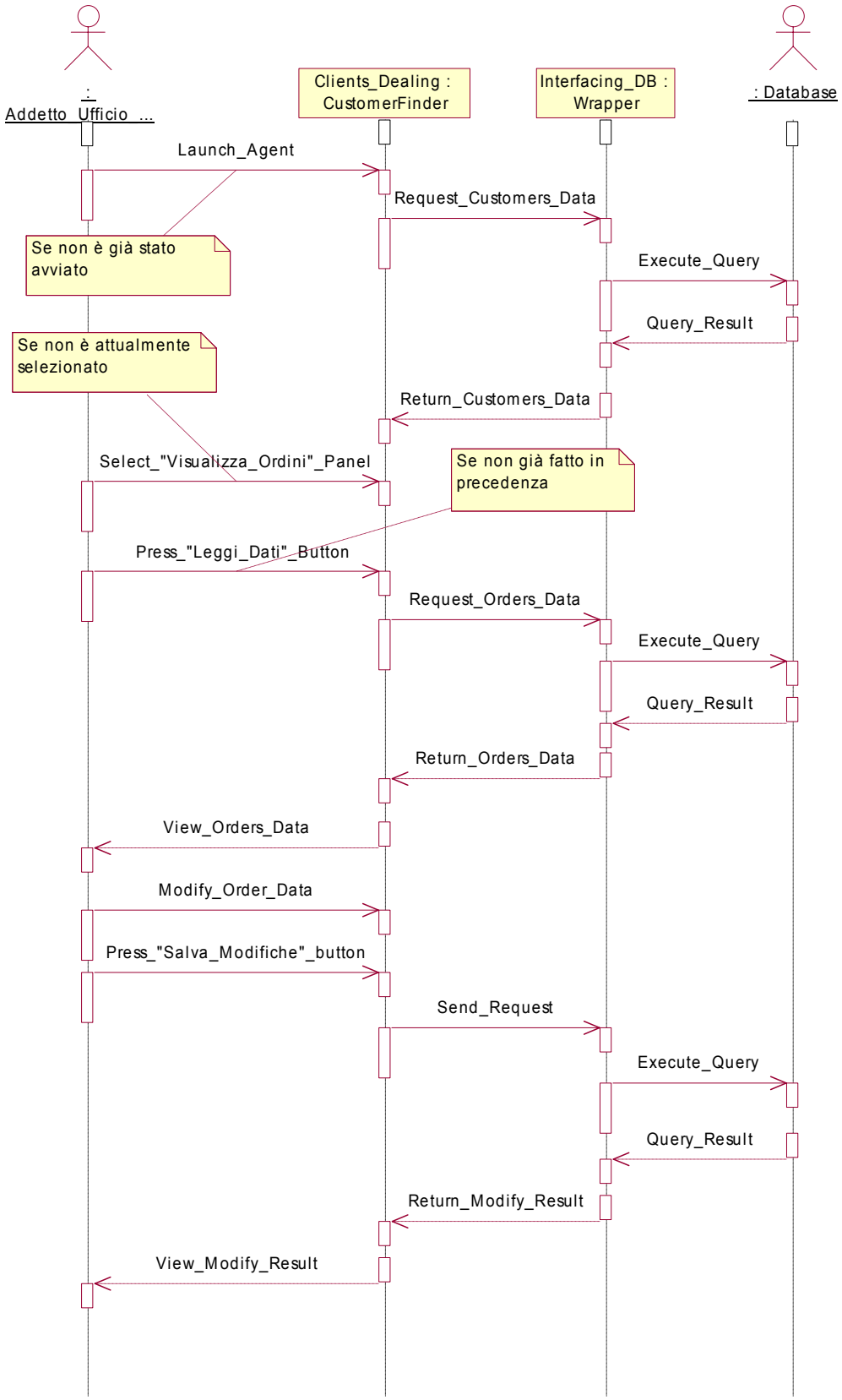
Insert_Order



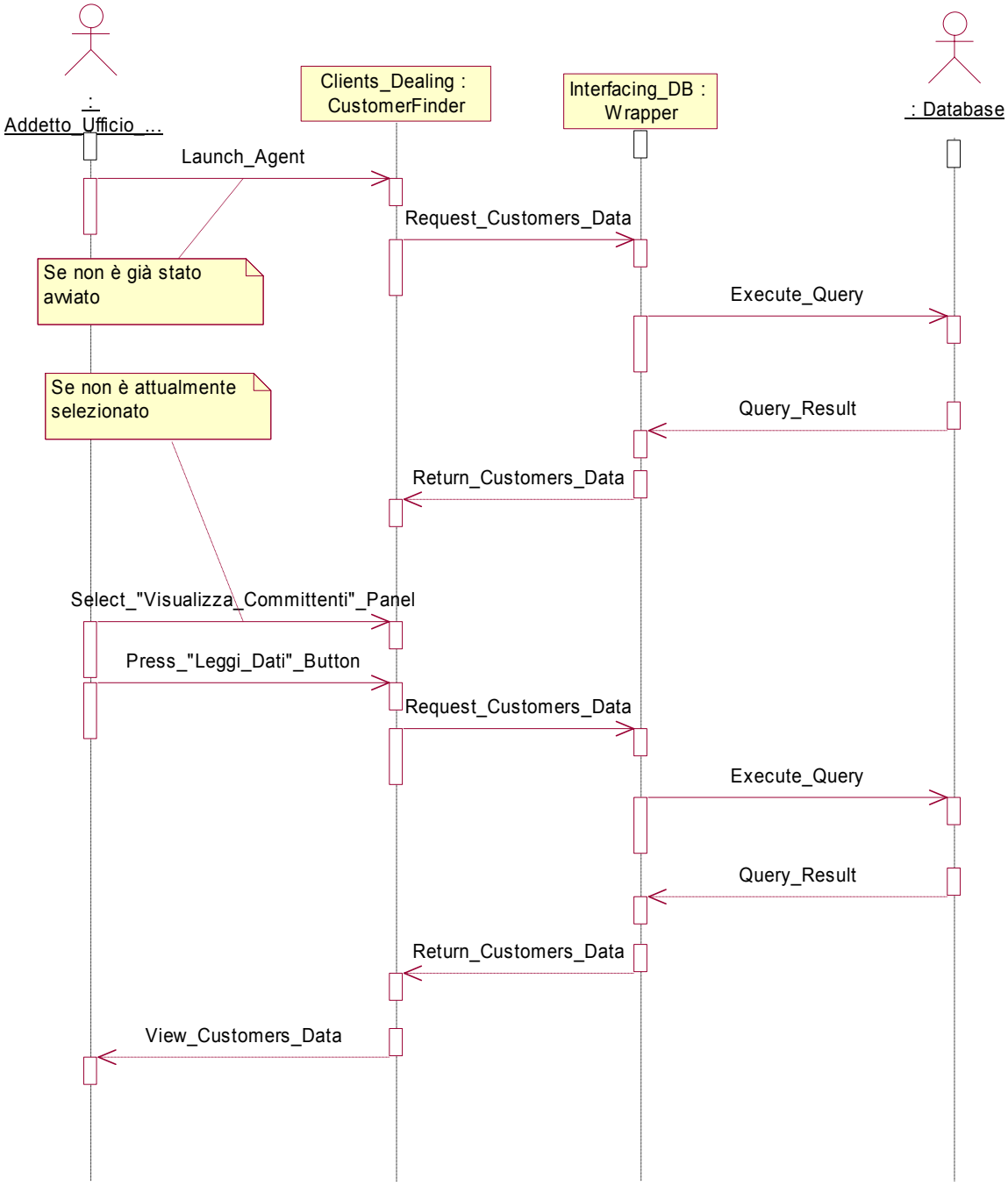
Modify_Customers



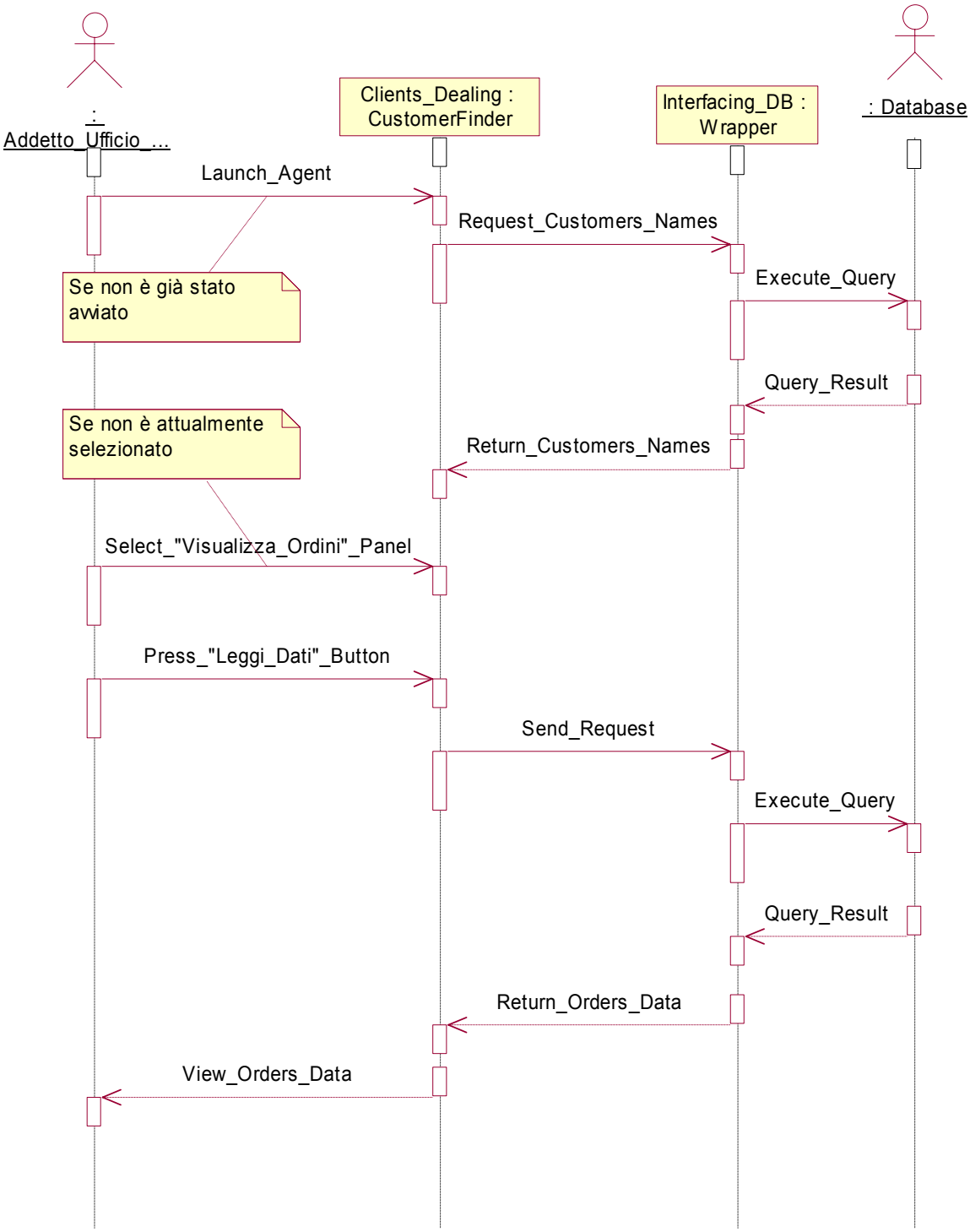
Modify_Orders



Read_Customers



Read_Orders


: Addetto_Ufficio_...
Clients_Dealing : CustomerFinder
Interfacing_DB : Wrapper
: Database
Launch_Agent
Request_Customers_Names
Execute_Query
Query_Result
Return_Customers_Names
Select_"Visualizza_Ordini"_Panel
Press_"Leggi_Dati"_Button
Send_Request
Execute_Query
Query_Result
Return_Orders_Data
View_Orders_Data
Se non è già stato avviato
Se non è attualmente selezionato

Name Role	Agent which plays it	Description	Responsibilities
Receive_Start_Working	OrderSwitcher	Riceve le comunicazioni di nuovi ordini inseriti e quindi nuovi lotti da creare	Ricezione della notifica di nuovi ordini inseriti visualizzando un messaggio sul terminale
Receive_Start_Working	ProductionScheduler	Riceve dall'order switcher un messaggio che indica che ci sono nuovi lotti inseriti da schedulare	Ricezione della notifica di nuovi lotti inseriti visualizzando un messaggio sul terminale
Receive_Start_Working	StoreKeeper	Riceve dal Production Manager l'informazione che sono state prodotte nuove biciclette e che quindi si può provvedere all'approvvigionamento dei materiali	Ricezione della notifica di nuove biciclette prodotte visualizzando un messaggio sul terminale
Clients_Dealing	CustomerFinder	Permette l'inserimento di un nuovo cliente e di un nuovo ordine e la visualizzazione, modifica e cancellazione degli ordini e dei dati dei clienti precedentemente inseriti	Gestione delle form di compilazione, visualizzazione e modifica degli ordini e dei committenti
Lots_Creation	OrderSwitcher	Crea i lotti a partire dai nuovi ordini inseriti	Interazione col software per la creazione dei lotti
Make_Scheduling	ProductionScheduler	Legge i dati relativi ai lotti da produrre e ne permette la schedulazione tramite un avanzato supporto grafico	Interazione dell'utente col software per la realizzazione della schedulazione dei lotti da produrre nello stabilimento
Make_Label	ProductionManager	Indica quali lotti sono stati prodotti e ne crea automaticamente le etichette che potranno essere visualizzate e stampate	Gestione della fase produttiva di conferma avvenuta produzione delle biciclette e preparazione di queste per la vendita
Storehouse_Management	StoreKeeper	Permette la visualizzazione e modifica delle distinte di produzione, e la stampa della lista degli approvvigionamenti delle materie prime	Gestione del magazzino e dei componenti per la produzione di entrambi i modelli

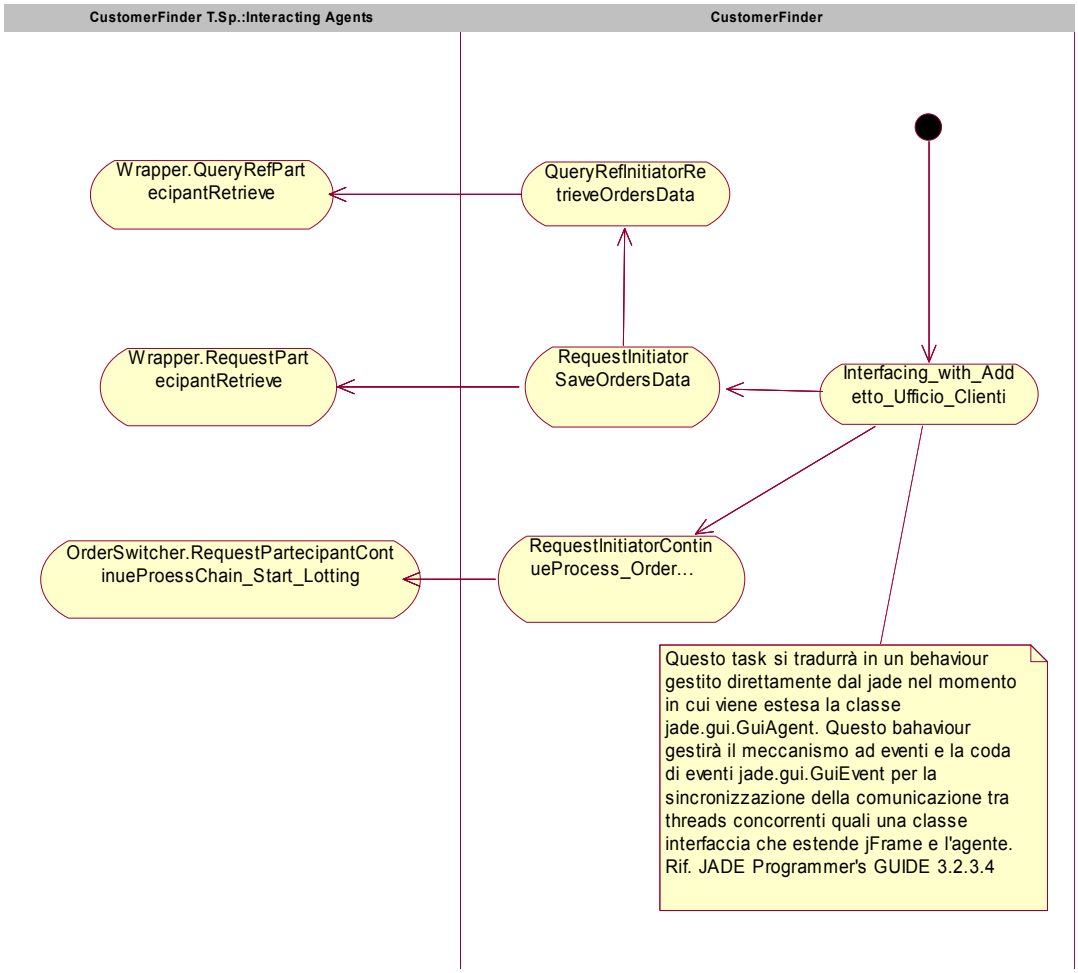
Interfacing_DB	Wrapper	Gestisce tutte transizioni dei dati col DB	Interfacciamento diretto col Database dal quale vengono prelevati ed inseriti tutti i dati dei vari processi produttivi

04-Tasks Specification phase

Individuati i ruoli e le loro interazioni per ogni agente si focalizza l'attenzione sugli specifici compiti che esso sarà in grado di svolgere e sui servizi che potrà offrire alla società. Le frecce indicano l'evento che fa iniziare il task. Nel caso di agenti con interfaccia grafica, tutti tranne il Wrapper, si evince subito come tali eventi scaturiscano per ogni agente da un task nominato "*Interfacing_with*_utente"; nella realtà questi tasks non saranno realizzati dal progettista in quanto intrinseci direttamente nella gestione della piattaforma Jade della classe *GuiAgent.* Si è deciso di inserirli ugualmente oltre che per una migliore lettura di tali diagrammi anche per una maggiore semplicità e coerenza nello sviluppo delle successive fasi progettuali.

Agent: CustomerFinder

L'agente che si occupa dell'inserimento dei dati degli ordini e dei clienti.



Task: RequestInitiatorSaveOrdersData

Description	Invia le richieste di salvataggio dei dati relativi a ordini e clienti ad altri agenti
Action	
Data	
Behavior	AchieveREInitiator

Task: RequestInitiatorContinueProcess_OrderStored

Description	Invia un messaggio ad altri agenti tramite il quale comunica di aver finito il proprio compito e nello specifico di aver inserito nuovi ordini da evadere
Action	
Data	
Behavior	AchieveREInitiator

Task: Interfacing_with_Addetto_Ufficio_Clienti

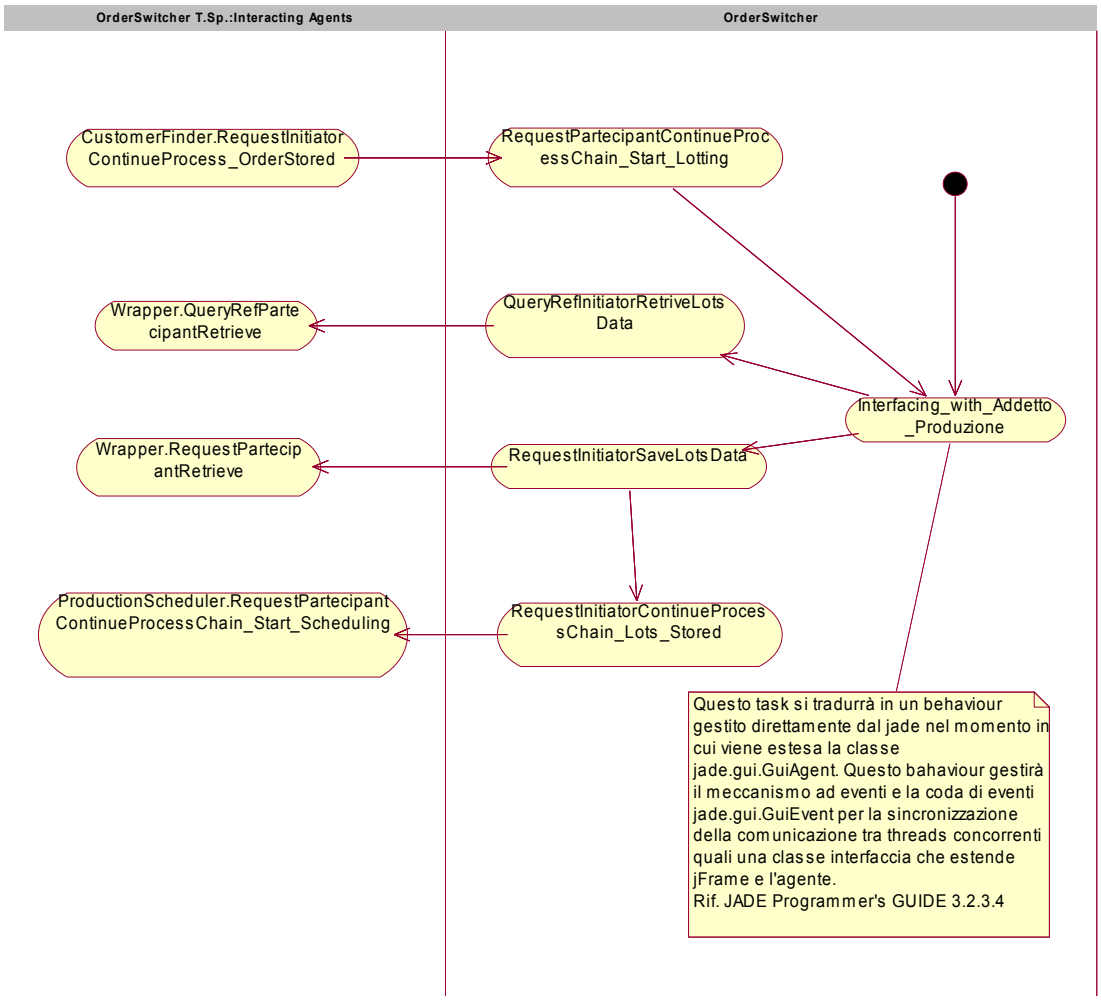
Description	Gestisce l'interfaccia di interazione con l'addetto all'ufficio clienti
Action	
Data	
Behavior	CyclicBehaviour (from GuiAgent)

Task: QueryRefInitiatorRetrieveOrdersData

Description	Invia le richieste di prelevamento dei dati relativi a ordini e clienti dal Database ad altri agenti
Action	
Data	
Behavior	AchieveREInitiator

Agent: OrderSwitcher

L'agente che si occupa della lottizzazione degli ordini e dell'invio dei lotti all'opportuno stabilimento.



Task: QueryRefInitiatorRetriveLotsData

Description	Invia ad altri agenti le richieste di dati e riceve da essi una comunicazione con tali dati annessi
Action	
Data	
Behavior	AchieveREInitiator

Task: RequestInitiatorSaveLotsData

Description	Invia ad altri agenti le richieste di salvataggio dei dati sui lotti creati
Action	
Data	
Behavior	AchieveREInitiator

Task: RequestInitiatorContinueProcessChain_Lots_Stored

Description	Invia un messaggio ad altri agenti, informandoli dell'avvenuto inserimento di nuovi lotti da schedulare
Action	
Data	receiver
Behavior	AchieveREInitiator

Task: Interfacing_with_Addetto_Produzione

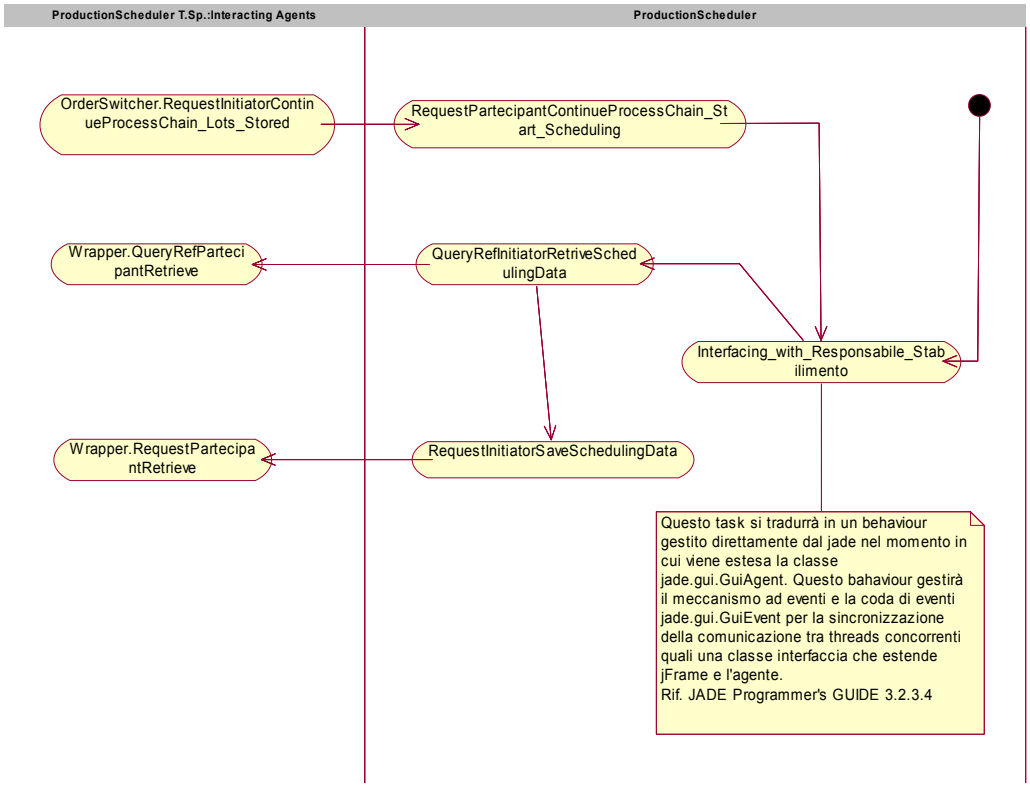
Description	Gestisce l'interazione tra l'agente e il personale addetto alla lottizzazione degli ordini
Action	
Data	
Behavior	CyclicBehaviour (from GuiAgent)

Task: RequestPartecipantContinueProcessChain_Start_Lotting

Description	Riceve il messaggio da altri agenti di avvenuto inserimento di nuovi ordini da schedulare
Action	
Data	receiver
Behavior	AchieveREResponder

Agent: ProductionScheduler

L'agente che si occupa della schedulazione dei lotti di produzione.



Task: RequestPartecipantContinueProcessChain_Start_Scheduling

Description	Riceve il messaggio da altri agenti di avvenuto inserimento di nuovi lotti da schedulare
Action	
Data	
Behavior	AchieveREResponder

Task: QueryRefInitiatorRetriveSchedulingData

Description	Invia ad altri agenti le richieste di dati relativi alla schedulazione e riceve da essi una comunicazione con tali dati annessi

Action	
Data	
Behavior	AchieveREInitiator

Task: RequestInitiatorSaveSchedulingData

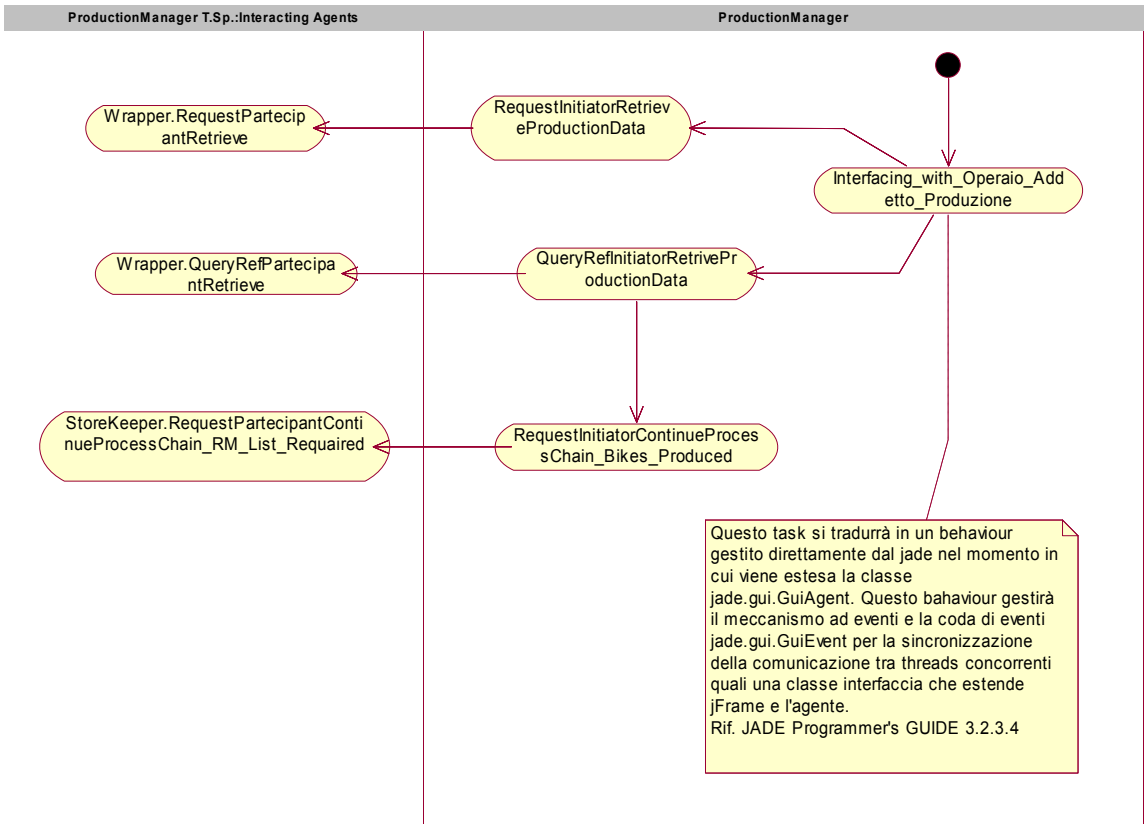
Description	Invia ad altri agenti la richiesta della memorizzazione dei dati della schedulazione fatta
Action	
Data	
Behavior	AchieveREInitiator

Task: Interfacing_with_Responsabile_Stabilimento

Description	Gestisce le interazioni tra l'agente e il responsabile di stabilimento che deve fare la schedulazione
Action	
Data	
Behavior	CyclicBehaviour (from GuiAgent)

Agent: ProductionManager

L'agente che si occupa della conferma dei lotti prodotti e della creazione delle etichette da apporre nelle biciclette.



Task: QueryRefInitiatorRetriveProductionData

Description	Invia ad altri agenti le richieste di dati della produzione e riceve da essi una comunicazione con tali dati annessi
Action	
Data	
Behavior	AchieveREInitiator

Task: RequestInitiatorContinueProcessChain_Bikes_Produced

Description	Invia ad altri agenti un messaggio che indica che nuove biciclette sono state prodotte
Action	
Data	
Behavior	AchieveREInitiator

Task: Interfacing_with_Operaio_Addetto_Produzione

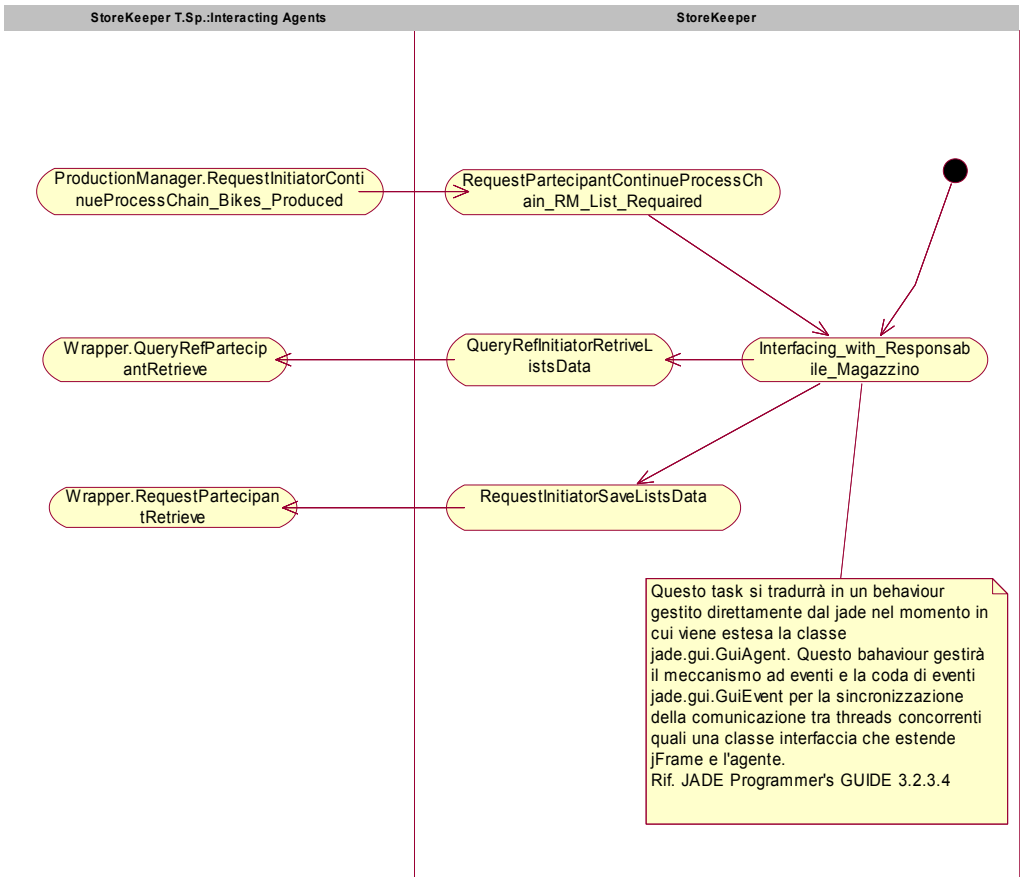
Description	Gestisce le interazioni tra l'agente e l'operaio addetto alla produzione delle biciclette
Action	
Data	
Behavior	CyclicBehaviour (from GuiAgent)

Task: RequestInitiatorRetrieveProductionData

Description	Invia ad altri agenti le richieste di salvataggio di dati della produzione e riceve da essi una comunicazione con l’esito dell’operazione
Action	
Data	
Behavior	AchieveREInitiator

Agent: StoreKeeper

L'agente che si occupa della creazione della lista per l'approvvigionamento delle materie prime e della modifica delle distinte di produzione.



Task: RequestPartecipantContinueProcessChain_RM_List_Requaired

Description	Riceve un messaggio da altri agenti che indica che nuove biciclette sono state prodotte e dunque è necessario fare l'approvvigionamento delle materie prime
Action	
Data	
Behavior	AchieveREResponder

Task: QueryRefInitiatorRetriveListsData

Description	Invia ad altri agenti le richieste dei dati delle distinte di produzione e riceve da essi una comunicazione con tali dati annessi
Action	
Data	
Behavior	AchieveREInitiator

Task: RequestInitiatorSaveListsData

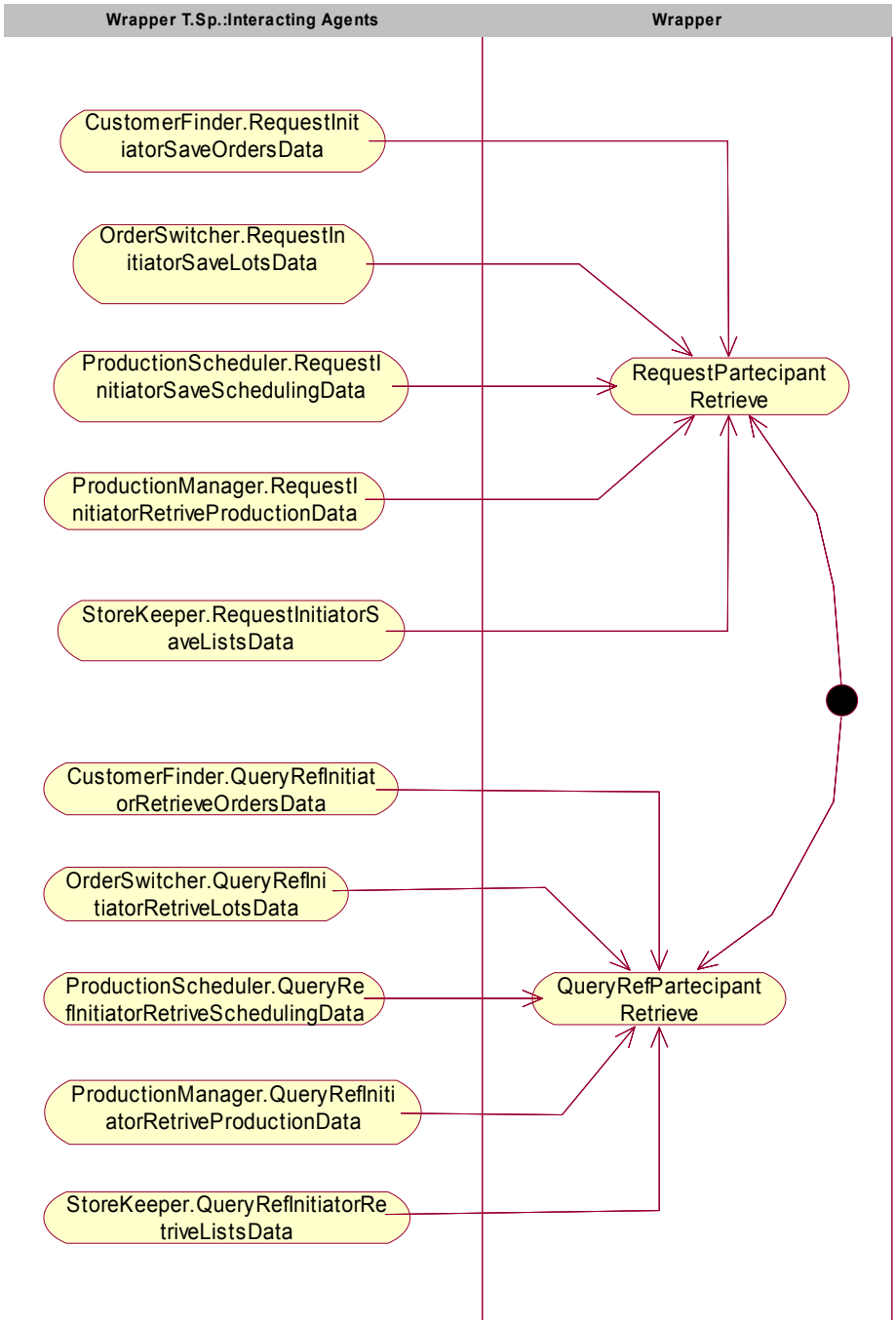
Description	Invia ad altri agenti la richiesta di salvataggio dei dati delle distinte di produzione
Action	
Data	
Behavior	AchieveREInitiator

Task: Interfacing_with_Responsabile_Magazzino

Description	Gestisce le interazioni tra l'agente ed il responsabile di magazzino
Action	
Data	
Behavior	CyclicBehaviour (from GuiAgent)

Agent: Wrapper

L'agente che si interfaccia al database.



Task: QueryRefPartecipantRetrieve

Description	Svolge tutti i servizi di prelevamento di dati dal database richiestigli dagli altri agenti
Action	
Data	
Behavior	AchieveREResponder

Task: RequestPartecipantRetrieve

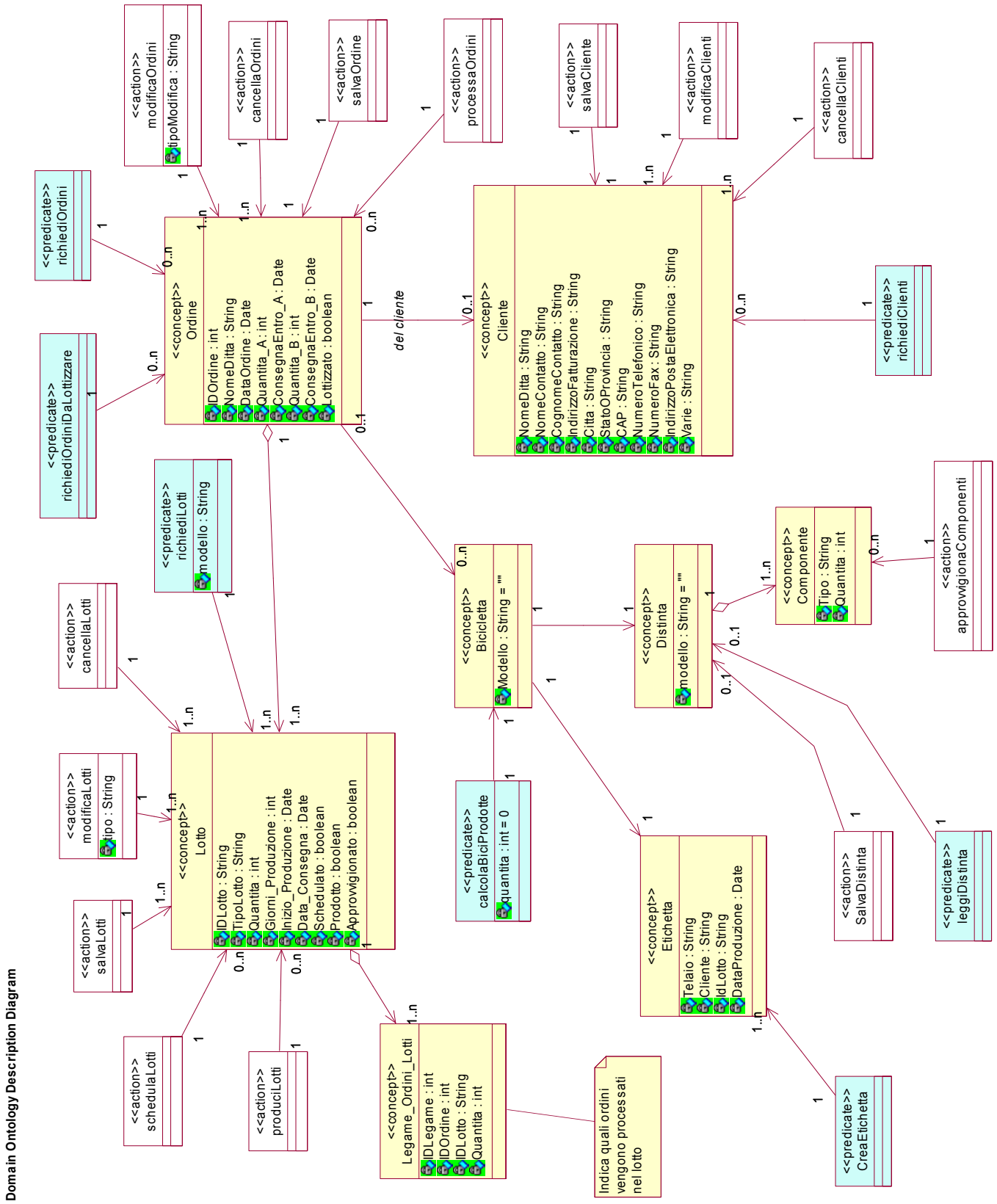
Description	Svolge tutti i servizi di memorizzazione dei dati richiestigli dagli altri agenti
Action	
Data	
Behavior	AchieveREResponder

05-Ontology Description phase

Nel realizzare una società di agenti è necessario andare a definire un ontologia comune a tutti gli agenti tramite la quale possono capirsi ed interagire; questo è realizzato mediante due differenti diagrammi, il Domain che descrive il dominio del sistema e il Communication dove si evidenziano le conoscenze di ogni singolo agente e le interazioni tra di essi.

Domain

Nel diagramma sottostante si evincono i concetti primitivi del mondo che si è voluto rappresentare (quello di una catena produttiva di biciclette), e le azioni e i predicati che agiscono su di essi



Elementi del Domain Ontology Diagram (DOD)

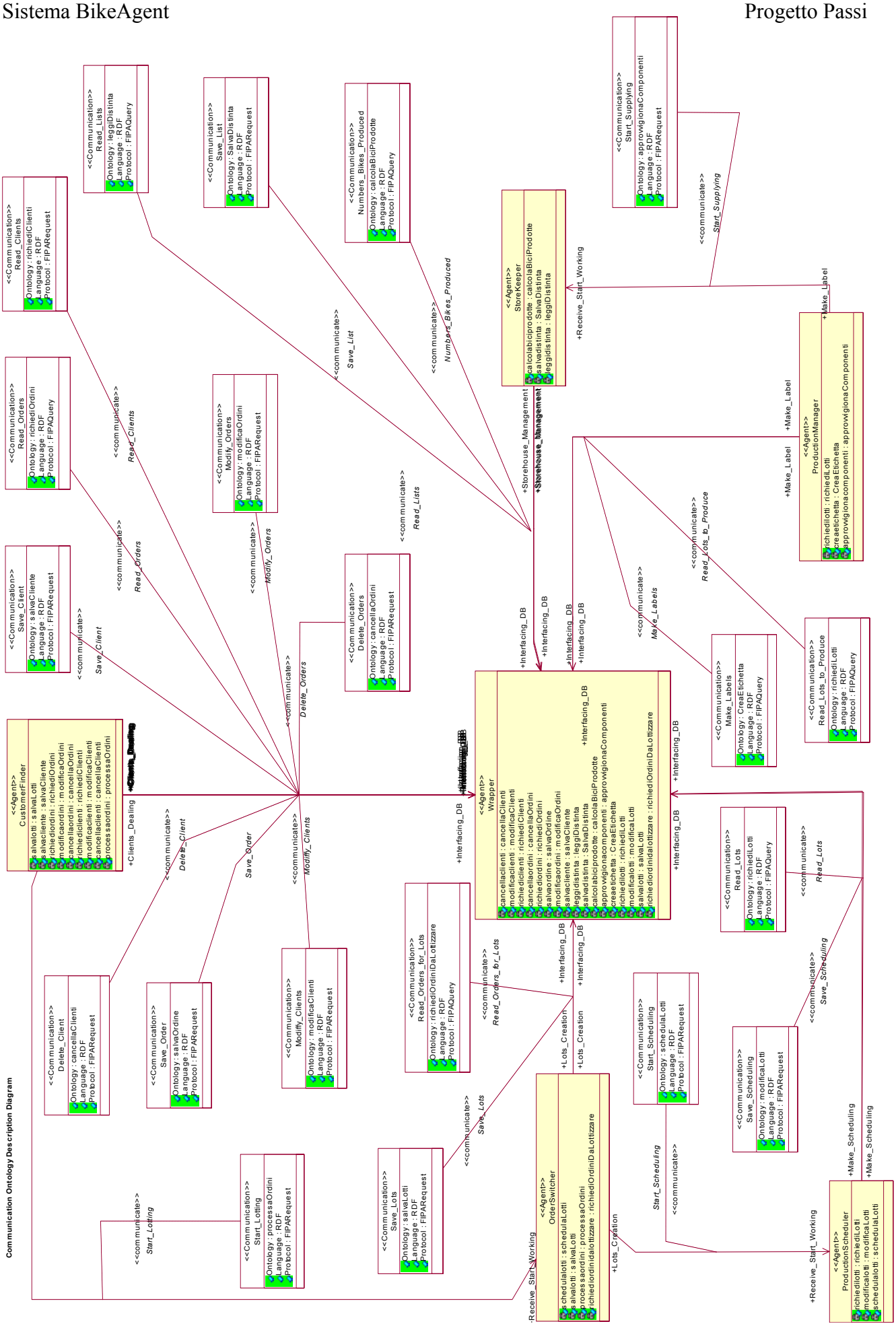
Data name	Stereotype	Field Name	Field Type	Description
salvaCliente	action			Salva un nuovo cliente nel Database
cancellaClienti	action			Cancella uno o più clienti contemporaneamente dal Database
modificaClienti	action			Modifica I dati dei clienti memorizzati
richiediClienti	predicate			Legge I dati dei clienti dal Database
salvaLotti	action			Salva I nuovi lotti nel Database
modificaLotti	action	tipo	String	Modifica i lotti già salvati
Legame_Ordini_Lotti	concept	IDLegame IDOrdine IDLotto Quantita	int int String int	Indica quail ordini sono elaborati in un unico lotto
cancellaLotti	action			Elimina i lotti creati dal Database
richiediLotti	predicate	modello	String	Legge i lotti dal Database
schedulaLotti	action			Invia un messaggio indicante che ci sono nuovi lotti da schedulare
produciLotti	action			Invia un messaggio indicante che ci sono nuovi lotti schedulati da produrre
CreaEtichetta	predicate			Crea e invia le etichette da apporre nei telai delle bici
salvaOrdine	action			Salva un ordine nel Database
modificaOrdini	action	tipoModifica	String	Modifica I parametri degli ordini memorizzati nel Database
cancellaOrdini	action			Cancella ordini dal Database
richiediOrdini	predicate			Legge gli ordini memorizzati
Cliente	concept	NomeDitta	String	Nome della sociatà del cliente o/e Ragione Sociale
		NomeContatto	String	Il nome del cliente

		CognomeContatto	String	Il cognome del cliente
		IndirizzoFatturazione	String	Indirizzo dove spedire le fatture
		Citta	String	Città
		StatoOProvincia	String	Provincia
		CAP	String	C.A.P.
		NumeroTelefonico	String	Numero telefonico
		NumeroFax	String	Numero di fax
		IndirizzoPostaElettronica	String	E-mail
		Varie	String	Note
richiediOrdiniDaLottizzare	predicate			Legge i nuovi ordini inseriti che devono essere lottizzati
processaOrdini	action			Invia un messaggio indicante che ci sono nuovi ordini da lottizzare
Lotto	concept	IDLotto	String	Identificativo unitario dei lotti
		TipoLotto	String	Se di modello “A” o “B”
		Quantita	int	Di quante bici è composto un lotto
		Giorni_Produzione	int	I giorni in cui il lotto verrà prodotto
		Inizio_Produzione	Date	Data inizio produzione lotto
		Data_Consegna	Date	Data consegna lotto
		Schedulato	boolean	Flag di gestione
		Prodotto	boolean	Flag di gestione
		Approvvigionato	boolean	Flag di gestione
Etichetta	concept	Telaio	String	Identificativo di ogni bicicletta
		Cliente	String	
		IDLotto	String	
		DataProduzione	Date	Quando è stato prodotto il lotto al quale la bici fa parte
Ordine	concept	IDOrdine	int	Codice univoco dell'ordine, automaticamente generato dal database bikeBD.
		NomeDitta	String	Codice univoco del cliente
		DataOrdine	Date	Data di inserimento dell'ordine
		Quantita_A	int	Quantità delle biciclette ordinate di tipo A.
		ConsegnaEntro_A	Date	Data di consegna per le biciclette di tipo A.
		Quantita_B	int	Quantità delle biciclette ordinate di tipo B.
		ConsegnaEntro_B	Date	Data di consegna per le biciclette di tipo B.
		Lottizzato	boolean	Flag che indica se l'ordine è stato processato e quindi

				lottizzato.
calcolaBiciProdotte	predicate	quantita	int	Restituisce due interi con il numero delle biciclette prodotte per poter stilare la lista delle materie prime
leggiDistinta	predicate			Legge le distinte di produzione
SalvaDistinta	action			Salva le modifica dei dati delle distinte di produzione
Bicicletta	concept	Modello	String	Concetto primitivo fondamentale dell'ontologia
Distinta	concept	modello	String	Insieme di componenti con relative quantità occorrenti per la produzione di una bicicletta
Componente	concept	Tipo Quantita	String int	Materie prime per la costruzione delle bici
approvvigionaComponenti	action			Invia un messaggio indicante che sono state prodotte nuove bici e quindi occorre produrre la lista di approvvigionamento delle materie prime

Communication

In questo diagramma vengono mostrate le conoscenze degli agenti (come attributi di ogni classe agente) e le comunicazioni che si hanno tra di essi con il protocollo e la base di conoscenza primitiva utilizzata nella comunicazione; come si vede dal diagramma non è necessario che ogni agente abbia come bagaglio di conoscenza propria tutto il dominio dell'ontologia, ma occorre che esso conosca i concetti, le azioni e i predicati utili al compimento dei propri servizi. E' facile vedere quindi come il Wrapper sia più "intelligente" (abbia più conoscenza) dello StoreKeeper ad esempio. Un agente con più conoscenza è un agente chiaramente più flessibile, perché si apre a più interazioni sociali con altri agenti, ma è un agente più pesante, per quel che riguarda la complessità computazionale, e più complesso da ottimizzare ed accrescere nel futuro.


Communication Ontology Description Diagram
<<Agent>>
CustomerFinder
<<Agent>>
Wrapper
<<Agent>>
OrderSwitcher
<<Agent>>
StoreKeeper
<<Agent>>
ProductionManager
<<Agent>>
ProductionScheduler
<<Communication>>
Read_Clients
<<Communication>>
Read_Lists
<<Communication>>
Save_List
<<Communication>>
Numbers_Bikes_Produced
<<Communication>>
Start_Supplying
<<Communication>>
Read_Orders
<<Communication>>
Modify_Orders
<<Communication>>
Save_Client
<<Communication>>
Delete_Orders
<<Communication>>
Delete_Client
<<Communication>>
Save_Order
<<Communication>>
Modify_Clients
<<Communication>>
Read_Orders_for_Lots
<<Communication>>
Save_Lots
<<Communication>>
Start_Lotting
<<Communication>>
Start_Scheduling
<<Communication>>
Save_Scheduling
<<Communication>>
Read_Lots
<<Communication>>
Make_Labels
<<Communication>>
Read_Lots_to_Produce
+Clients_Dealing
+Interfacing_DB
+Lots_Creation
+Storehouse_Management
+Receive_Start_Working
+Make_Label
+Make_Scheduling
Start_Lotting
Start_Scheduling
Start_Supplying
Read_Clients
Read_Orders
Read_Lists
Save_List
Numbers_Bikes_Produced
Save_Client
Delete_Orders
Delete_Client
Save_Order
Modify_Clients
Modify_Orders
Read_Orders_for_Lots
Save_Lots
Read_Lots
Save_Scheduling
Make_Labels
Read_Lots_to_Produce

Conoscenze degli agenti

Agent	Field Name	Data Type
ProductionManager	richiedilotti	richiediLotti
	creaetichetta	CreaEtichetta
	approvvigionacomponenti	approvvigionaComponenti
StoreKeeper	calcolabiciprodotte	calcolaBiciProdotte
	salvadistinta	SalvaDistinta
	leggidistinta	leggiDistinta
CustomerFinder	salvalotti	salvaLotti
	salvacliente	salvaCliente
	richiediordini	richiediOrdini
	modificaordini	modificaOrdini
	cancellaordini	cancellaOrdini
	richiediclienti	richiediClienti
	modificaclienti	modificaClienti
	cancellaclienti	cancellaClienti
	processaordini	processaOrdini
Wrapper	cancellaclienti	cancellaClienti
	modificaclienti	modificaClienti
	richiediclienti	richiediClienti
	cancellaordini	cancellaOrdini
	richiediordini	richiediOrdini
	salvaordine	salvaOrdine
	modificaordini	modificaOrdini
	salvacliente	salvaCliente
	leggidistinta	leggiDistinta
	salvadistinta	SalvaDistinta
	calcolabiciprodotte	calcolaBiciProdotte
	approvvigionacomponenti	approvvigionaComponenti
	creaetichetta	CreaEtichetta
	richiedilotti	richiediLotti
	modificalotti	modificaLotti
	salvalotti	salvaLotti
	richiediordinidalottizzare	richiediOrdiniDaLottizzare
OrderSwitcher	schedulalotti	schedulaLotti
	salvalotti	salvaLotti
	processaordini	processaOrdini
	richiediordinidalottizzare	richiediOrdiniDaLottizzare
ProductionScheduler	richiedilotti	richiediLotti
	modificalotti	modificaLotti
	schedulalotti	schedulaLotti

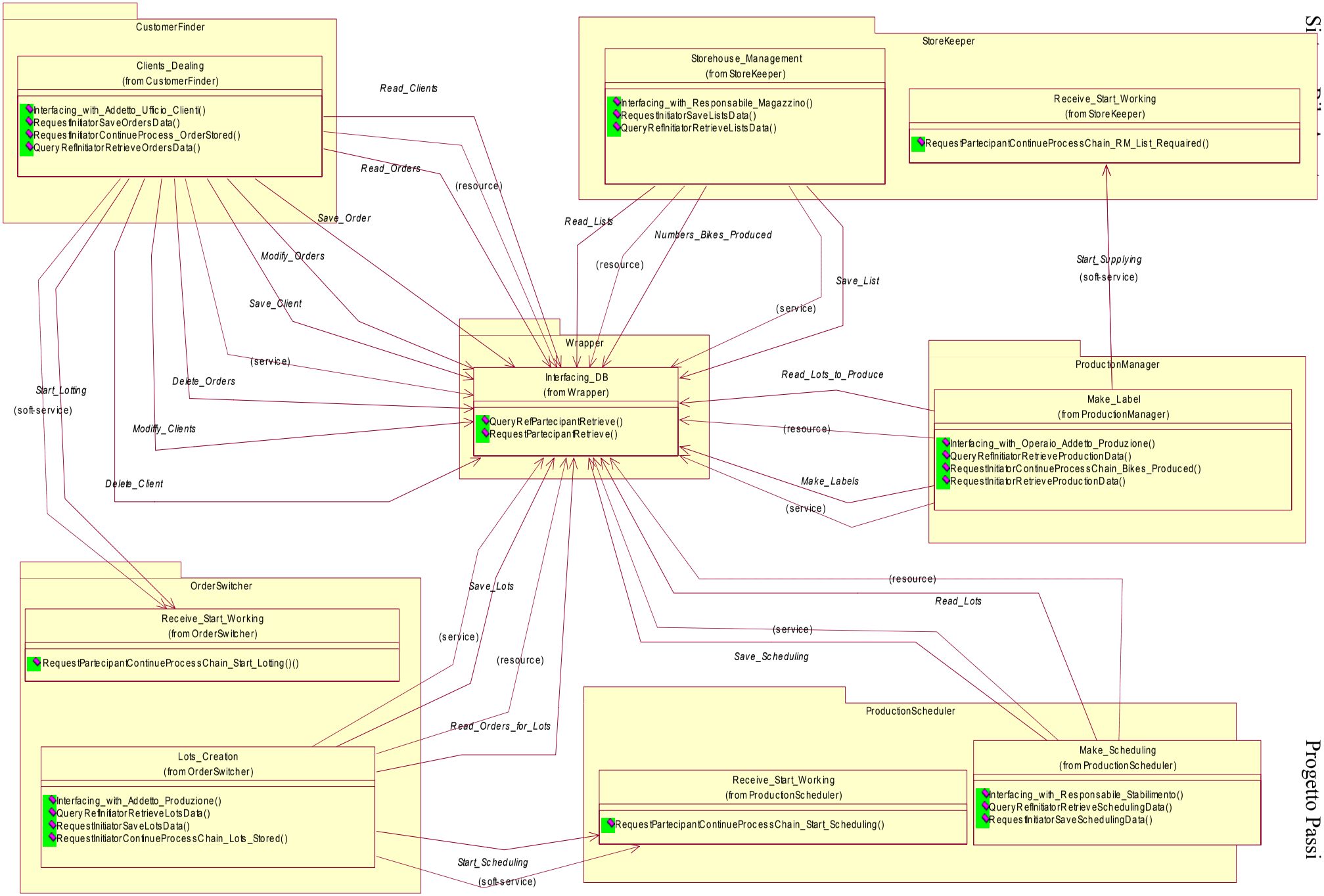
Message content

Name	MsgID	From Agent	To Agent	Language	Protocol	Data
Start_Supplying	14	ProductionManager	StoreKeeper	RDF	FIPARequest	approvvigionaComponenti
Save_Order	2	CustomerFinder	Wrapper	RDF	FIPARequest	salvaOrdine
Save_Client	3	CustomerFinder	Wrapper	RDF	FIPARequest	salvaCliente
Read_Orders	4	CustomerFinder	Wrapper	RDF	FIPAQuery	richiediOrdini
Modify_Orders	5	CustomerFinder	Wrapper	RDF	FIPARequest	modificaOrdini
Delete_Orders	6	CustomerFinder	Wrapper	RDF	FIPARequest	cancellaOrdini
Read_Clients	7	CustomerFinder	Wrapper	RDF	FIPARequest	richiediClienti
Modiffy_Clients	8	CustomerFinder	Wrapper	RDF	FIPARequest	modificaClienti
Delete_Client	9	CustomerFinder	Wrapper	RDF	FIPARequest	cancellaClienti
Make_Labels	15	ProductionManager	Wrapper	RDF	FIPAQuery	CreaEtichetta
Read_Lots_to_Produce	16	ProductionManager	Wrapper	RDF	FIPAQuery	richiediLotti
Read_Lists	17	StoreKeeper	Wrapper	RDF	FIPAQuery	leggiDistinta
Numbers_Bikes_Produced	18	StoreKeeper	Wrapper	RDF	FIPAQuery	calcolaBiciProdotte
Save_List	19	StoreKeeper	Wrapper	RDF	FIPARequest	SalvaDistinta
Start_Lotting	1	CustomerFinder	OrderSwitcher	RDF	FIPARequest	processaOrdini
Read_Orders_for_Lots	10	OrderSwitcher	Wrapper	RDF	FIPAQuery	richiediOrdiniDaLottizzare
Save_Lots	11	OrderSwitcher	Wrapper	RDF	FIPARequest	salvaLotti
Read_Lots	12	ProductionScheduler	Wrapper	RDF	FIPAQuery	richiediLotti
Save_Scheduling	13	ProductionScheduler	Wrapper	RDF	FIPARequest	modificaLotti
Start_Scheduling	20	OrderSwitcher	ProductionScheduler	RDF	FIPARequest	schedulaLotti

06-Roles Description phase

In questo diagramma viene mostrata abbastanza dettagliatamente la società degli agenti creata,il ruolo che ognuno di essi gioca nel suo interno e come si svolgono le relazioni sociali. Vengono infatti visualizzati i ruoli che svolgono gli agenti con specificati i comportamenti (metodi) che assumono per raggiungere i propri scopi ed offrire agli altri agenti i propri servizi.

Role Description Diagram



Communication

From	To	Name	Refer to C.O.D. msg (MsgID)
CustomerFinder:Clients_Dealing	OrderSwitcher:Receive_Start_Working	Start_Lotting	1
OrderSwitcher:Lots_Creation	ProductionScheduler:Receive_Start_Working	Start_Scheduling	20
ProductionManager:Make_Label	StoreKeeper:Receive_Start_Working	Start_Supplying	14
OrderSwitcher:Lots_Creation	Wrapper:Interfacing_DB	Read_Orders_for_Lots	10
ProductionScheduler:Make_Scheduling	Wrapper:Interfacing_DB	Read_Lots	12
ProductionManager:Make_Label	Wrapper:Interfacing_DB	Make_Labels	15
CustomerFinder:Clients_Dealing	Wrapper:Interfacing_DB	Save_Order	2
CustomerFinder:Clients_Dealing	Wrapper:Interfacing_DB	Save_Client	3
CustomerFinder:Clients_Dealing	Wrapper:Interfacing_DB	Read_Orders	4
CustomerFinder:Clients_Dealing	Wrapper:Interfacing_DB	Read_Clients	7
CustomerFinder:Clients_Dealing	Wrapper:Interfacing_DB	Modify_Orders	5
CustomerFinder:Clients_Dealing	Wrapper:Interfacing_DB	Delete_Orders	6
CustomerFinder:Clients_Dealing	Wrapper:Interfacing_DB	Modiffy_Clients	8
CustomerFinder:Clients_Dealing	Wrapper:Interfacing_DB	Delete_Client	9
OrderSwitcher:Lots_Creation	Wrapper:Interfacing_DB	Save_Lots	11
ProductionScheduler:Make_Scheduling	Wrapper:Interfacing_DB	Save_Scheduling	13
ProductionManager:Make_Label	Wrapper:Interfacing_DB	Read_Lots_to_Produce	16
StoreKeeper:Storehouse_Management	Wrapper:Interfacing_DB	Read_Lists	17
StoreKeeper:Storehouse_Management	Wrapper:Interfacing_DB	Numbers_Bikes_Produced	18
StoreKeeper:Storehouse_Management	Wrapper:Interfacing_DB	Save_List	19

Dependencies

Dependent/Client	Dependee/server	Kind of dependency	Description	Attached Note
Clients_Dealing	Interfacing_DB	(resource)	Richiesta di prelevamento dati degli ordini e/o dei clienti dal Database	Il CustomerFinder richiede al wrapper di poter disporre dei dati degli ordini e dei clienti necessari per svolgere le proprie attività
Clients_Dealing	Interfacing_DB	(service)	Richiesta di salvataggio dati per completare le proprie attività	Il CustomerFinder per raggiungere i suoi goal svolge determinate azioni di interazione con l'operatore e deve poi salvare i dati elaborati nel Database
Clients_Dealing	Receive_Start_Working	(soft-service)	L'agente OrderSwitcher comincia a svolgere le sue operazioni quando gli arriva una notifica dal CustomerFinder; non è però una operazione necessaria per il processo produttivo	Tramite questo messaggio viene dato solo un avviso all'operatore di inserimento di nuovi ordini, l'operazione di lottizzazione parte sempre da lui.
Lots_Creation	Interfacing_DB	(service)	Richiesta di salvataggio dei dati dei lotti creati per completare le proprie attività	L'OrderSwitcher per raggiungere i suoi goal svolge determinate azioni di interazione con l'operatore e deve poi salvare i dati elaborati nel Database
Lots_Creation	Interfacing_DB	(resource)	Richiesta di prelevamento dati dei nuovi ordini inseriti dal Database	L'OrderSwitcher per raggiungere i suoi goal necessita dei dati di produzione memorizzati nel Database
Lots_Creation	Receive_Start_Working	(soft-service)	L'agente ProductionScheduler	Tramite questo messaggio viene

			comincia a svolgere le sue operazioni quando gli arriva una notifica dal OrderSwitcher; non è però una operazione necessaria per il processo produttivo	dato solo un avviso all'operatore di inserimento di nuovi lotti da schedulare; l'operazione di schedulazione parte sempre da lui.
Make_Scheduling	Interfacing_DB	(service)	Richiesta di salvataggio dei dati della schedulazione	Il ProductionScheduler per raggiungere i suoi goal svolge determinate azioni di interazione con l'operatore e deve poi salvare i dati elaborati nel Database
Make_Scheduling	Interfacing_DB	(resource)	Richiesta di prelevamento dati dei lotti dal Database	Il ProductionScheduler per raggiungere i suoi goal necessita dei dati di produzione memorizzati nel Database
Make_Label	Interfacing_DB	(resource)	Richiesta di prelevamento dati delle etichette dal Database	Il ProductionManager per raggiungere i suoi goal necessita dei dati di produzione memorizzati nel Database
Make_Label	Interfacing_DB	(service)	Richiesta di salvataggio conferma produzione dei lotti e richiesta di creazione delle etichette	Il ProductionManager per raggiungere i suoi goal svolge determinate azioni di interazione con l'operatore e deve poi salvare i dati elaborati nel Database

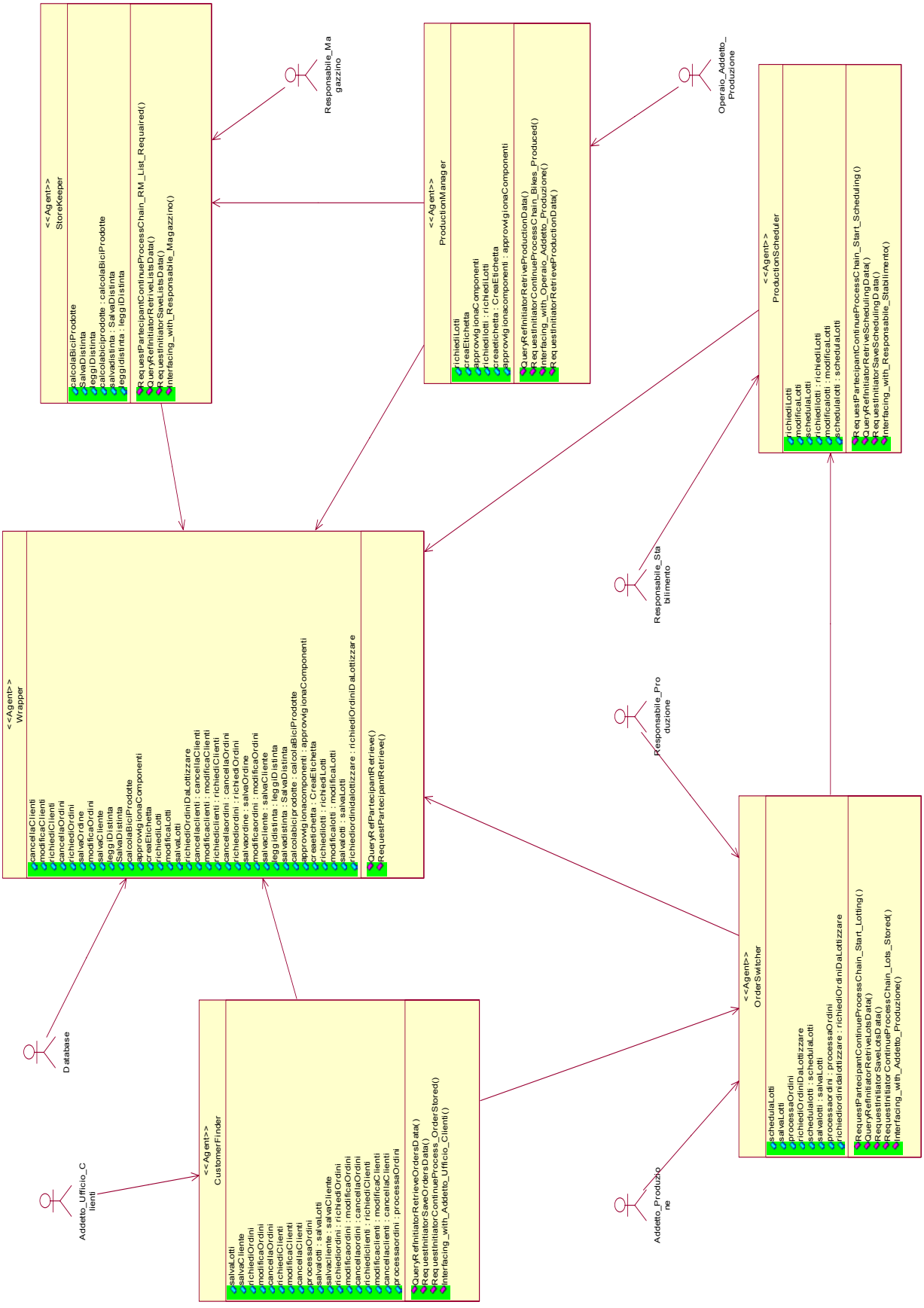
Make_Label	Receive_Start_Working	(soft-service)	L'agente StoreKeeper comincia a svolgere le sue operazioni quando gli arriva una notifica dal ProductionManager; non è però una operazione necessaria per il processo produttivo	Tramite questo messaggio viene dato solo un avviso all'operatore di produzione di nuove biciclette e quindi della necessità di approvvigionare i materiali ma l'operazione di creazione della lista deve partire sempre da lui.
Storehouse_Management	Interfacing_DB	(service)	Richiesta di salvataggio delle modifiche sui dati delle distinte di produzione schedulazione	Lo StoreKeeper per raggiungere i suoi goal svolge determinate azioni di interazione con l'operatore e deve poi salvare i dati elaborati nel Database
Storehouse_Management	Interfacing_DB	(resource)	Richiesta dei dati delle distinte e delle quantità di bici da approvvigionare	Lo StoreKeeper per raggiungere i suoi goal necessita dei dati di produzione memorizzati nel Database

07-Protocol Description phase

Per lo sviluppo di tale progetto non è stato necessario l'utilizzo di protocolli non FIPA standard; infatti gli unici protocolli utilizzati sono stati il FIPAQuery, nel caso di predicati, e il FIPARequest nel caso di azioni inviate da l'uno all'altro agente della società

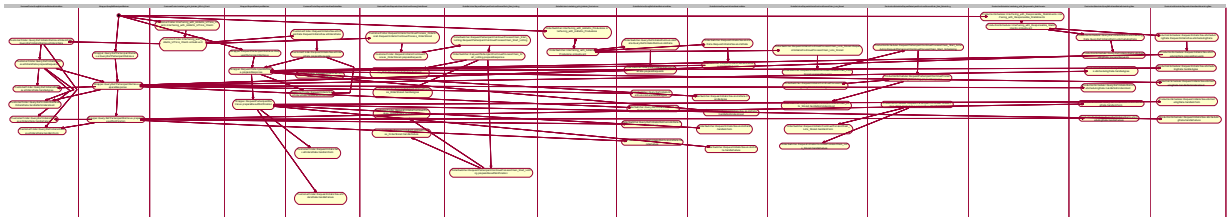
08-Multi-Agent Structure Definition phase

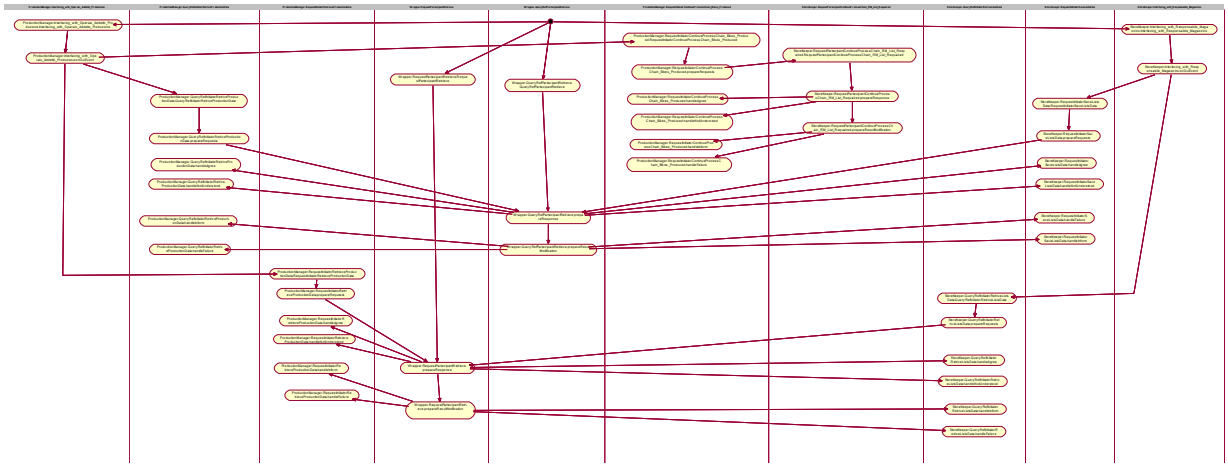
Questo diagramma mostra l'intera società degli agenti e per ognuno di essi evidenzia il bagaglio di conoscenza, i compiti (task) che svolge all'interno di tale società e gli attori con cui interagisce durante il processo produttivo


<<Agent>>
StoreKeeper
calcolaBiciProdotte
SalvaDistinta
leggiDistinta
calcolabiciprodotte : calcolaBiciProdotte
salvadistinta : SalvaDistinta
leggiDistinta : leggiDistinta
RequestPartecipantContinueProcessChain_RM_List_Requaired()
QueryRefinitiatorRetrieveListaData()
RequestInitiatorSaveListaData()
Interfacing_with_Responsabile_Magazzino()
Responsabile_Magazzino
<<Agent>>
ProductionManager
richiediLotti
creaEtichetta
approvvigionaComponenti
richiediLotti : richiediLotti
creaetichetta : CreaEtichetta
approvvigionacomponenti : approvvigionaComponenti
QueryRefinitiatorRetrieveProductionData()
RequestInitiatorContinueProcessChain_Bikes_Produced()
Interfacing_with_Operaio_Addetto_Produzione()
RequestInitiatorRetrieveProductionData()
Operaio_Addetto_Produzione
<<Agent>>
ProductionScheduler
richiediLotti
modificaLotti
scheduleLotti
richiediLotti : richiediLotti
modificalotti : modificaLotti
schedulalotti : scheduleLotti
RequestPartecipantContinueProcessChain_Start_Scheduling()
QueryRefinitiatorRetrieveSchedulingData()
RequestInitiatorSaveSchedulingData()
Interfacing_with_Responsabile_Stabilimento()
Responsabile_Stabilimento
<<Agent>>
Wrapper
cancellaClienti
modificaClienti
richiediClienti
cancellaOrdini
richiediOrdini
salvaOrdine
modificaOrdini
salvaCliente
leggiDistinta
SalvaDistinta
calcolaBiciProdotte
approvvigionaComponenti
creaEtichetta
richiediLotti
modificaLotti
salvaLotti
cancellaClienti
richiediOrdiniDaLottizzare
cancellaClienti : cancellaClienti
modificaclienti : modificaClienti
richiediclienti : richiediClienti
cancellaordini : cancellaOrdini
richiediordini : richiediOrdini
salvaordine : salvaOrdine
modificaordini : modificaOrdini
salvacliente : salvaCliente
leggidistinta : leggiDistinta
salvadistinta : SalvaDistinta
calcolabiciprodotte : calcolaBiciProdotte
approvvigionacomponenti : approvvigionaComponenti
creaetichetta : CreaEtichetta
richiedilotti : richiediLotti
modificalotti : modificaLotti
salvalotti : salvaLotti
richiediordinidalottizzare : richiediOrdiniDaLottizzare
QueryRefPartecipantRetrieve()
RequestPartecipantRetrieve()
Responsabile_Produzione
<<Agent>>
OrderSwitcher
scheduleLotti
salveLotti
processaOrdini
richiediOrdiniDaLottizzare
schedulalotti : scheduleLotti
salvalotti : salveLotti
processaordini : processaOrdini
richiediordinidalottizzare : richiediOrdiniDaLottizzare
RequestPartecipantContinueProcessChain_Start_Lotting()
QueryRefinitiatorRetrieveLotsData()
RequestInitiatorSaveLotsData()
RequestInitiatorContinueProcessChain_Lots_Stored()
Interfacing_with_Addetto_Produzione()
Addetto_Produzione
Database
<<Agent>>
CustomerFinder
salvaLotti
salvaCliente
richiediOrdini
modificaOrdini
cancellaOrdini
richiediClienti
modificaClienti
cancellaClienti
processaOrdini
salvalotti : salvaLotti
salvacliente : salvaCliente
richiediordini : richiediOrdini
modificaordini : modificaOrdini
cancellaordini : cancellaOrdini
richiedicLienti : richiediClienti
modificaclienti : modificaClienti
cancellaclienti : cancellaClienti
processaordini : processaOrdini
QueryRefinitiatorRetrieveOrdersData()
RequestInitiatorSaveOrdersData()
RequestInitiatorContinueProcess_OrderStored()
Interfacing_with_Addetto_Ufficio_Clienti()
Addetto_Ufficio_Clienti

09-Multi-Agent Behavior Description phase

In questo diagramma sono rappresentatiil flusso di eventi, I metodi utilizzati e I messaggi scambiati tra gli agenti

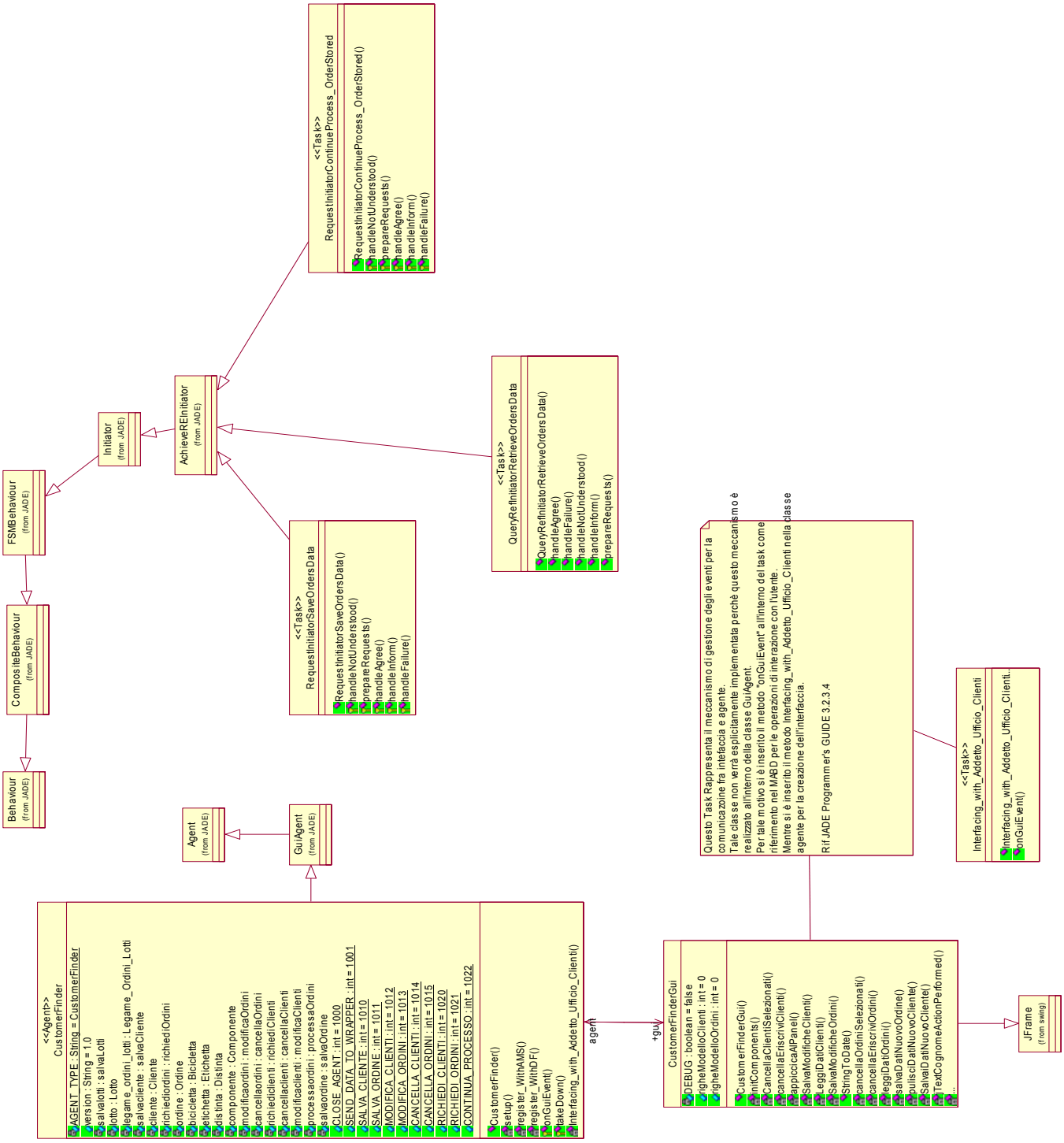




Single Agent Structure Definition phase

Agent: CustomerFinder

L'agente che si occupa dell'inserimento dei dati degli ordini e dei clienti.



Attributes

Class	Attribute	Data Type	Description
RequestInitiatorSaveOrdersData			Prepara i messaggi per l'invio di dati
RequestInitiatorContinueProcess_OrderStored			Prepara il messaggio di notifica inserimento ordini
Interfacing_with_Addetto_Ufficio_Clienti			Gestione degli eventi della GUI
QueryRefInitiatorRetrieveOrdersData			Prepara i messaggi per la richiesta di dati
GuiAgent			Gestisce gli eventi provenienti dall'interfaccia
AchieveREInitiator			Implementazione dei protocolli Fipa Request, Query e Propose
JFrame			La classe da cui ereditano le interfacce grafiche
CustomerFinder	AGENT_TYPE	String	
	version	String	
	salvalotti	salvaLotti	Elemento dell'ontologia
	lotto	Lotto	Elemento dell'ontologia
	legame_ordini_lotti	Legame_Ordini_Lotti	Elemento dell'ontologia
	salvacliente	salvaCliente	Elemento dell'ontologia
	cliente	Cliente	Elemento dell'ontologia
	richiediordini	richiediOrdini	Elemento dell'ontologia
	ordine	Ordine	Elemento dell'ontologia
	bicicletta	Bicicletta	Elemento dell'ontologia
	etichetta	Etichetta	Elemento dell'ontologia
	distinta	Distinta	Elemento dell'ontologia
	componente	Componente	Elemento dell'ontologia
	modificaordini	modificaOrdini	Elemento dell'ontologia
	cancellaordini	cancellaOrdini	Elemento dell'ontologia
	richiediclienti	richiediClienti	Elemento dell'ontologia
	cancellaclienti	cancellaClienti	Elemento dell'ontologia
	modificaclienti	modificaClienti	Elemento dell'ontologia
	processaordini	processaOrdini	Elemento dell'ontologia
	salvaordine	salvaOrdine	Elemento dell'ontologia
	CLOSE_AGENT	int	Costante
	SEND_DATA_TO_WRAPPER	int	Costante
	SALVA_CLIENTE	int	Costante
	SALVA_ORDINE	int	Costante
	MODIFICA_CLIENTI	int	Costante
	MODIFICA_ORDINI	int	Costante
	CANCELLA_CLIENTI	int	Costante
	CANCELLA_ORDINI	int	Costante
	RICHIEDI_CLIENTI	int	Costante
	RICHIEDI_ORDINI	int	Costante

	CONTINUA_PROCESSO	int	Costante
CustomerFinderGui	DEBUG	boolean	
	righeModelloClienti	int	Variabili utili per la correlazione agente-interfaccia
	righeModelloOrdini	int	

CompositeBehaviour
FSMBehaviour
Initiator

Methods

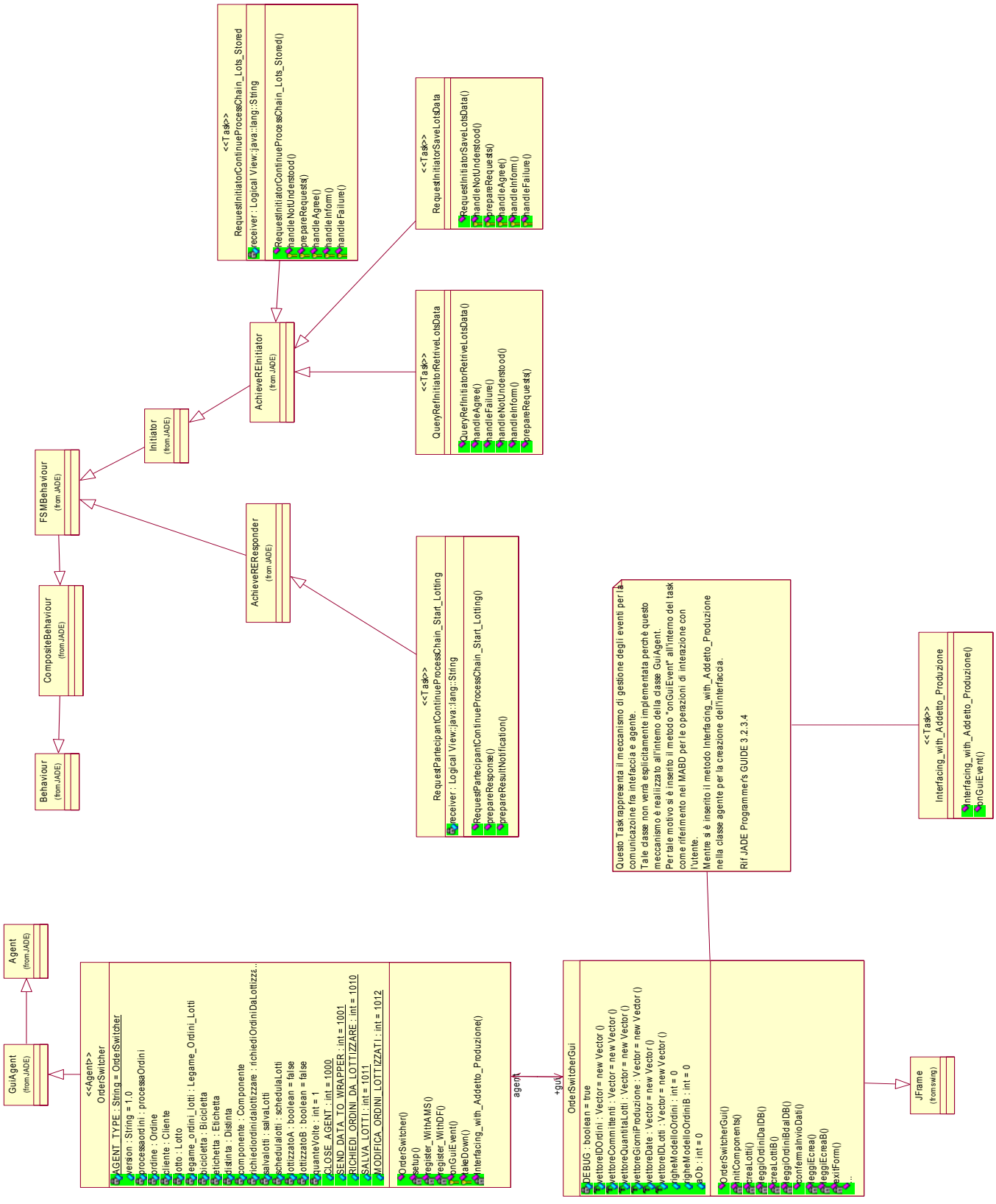
Class	Method	Parameters	Description
RequestInitiatorSaveOrdersData	RequestInitiatorSaveOrdersData		costruttore
	handleNotUnderstood	aCLMessage	Riceve un messaggio di ritorno in caso di incomprensione del messaggio inviato
	prepareRequests	msg	Prepara e invia il messaggio
	handleAgree	aCLMessage	Riceve un messaggio di accettazione della richiesta
	handleInform	aCLMessage	Riceve un messaggio con l'esito dell'azione svolta
	handleFailure	aCLMessage	Riceve un messaggio con l'informazione di fallimento dello svolgimento dell'azione richiesta
RequestInitiatorContinueProcess_OrderStored	RequestInitiatorContinueProcess_OrderStored		costruttore
	handleNotUnderstood	aCLMessage	Riceve un mesaggio di ritorno in caso di incomprensione del messaggio inviato
	prepareRequests	msg	Prepara e invia il messaggio
	handleAgree	aCLMessage	Riceve un messaggio di accettazione della richiesta
	handleInform	aCLMessage	Riceve un messaggio con l'esito dell'azione svolta
	handleFailure	aCLMessage	Riceve un messaggio con l'informazione di fallimento dello svolgimento dell'azione richiesta
Interfacing_with_Addetto_Ufficio_Clienti	Interfacing_with_Addetto_Ufficio_Clienti		costruttore

	onGuiEvent		Variabile che effettua il matching con gli eventi dell'interfaccia
QueryRefInitiatorRetrieveOrdersData	QueryRefInitiatorRetrieveOrdersData	owner	Variabile dell'agente
		prd	Variabile col predicato da mandare
	handleAgree	msg	Riceve un messaggio di accettazione della richiesta
	handleFailure	msg	Riceve un messaggio con l'informazione di fallimento dello svolgimento dell'azione richiesta
	handleNotUnderstood	msg	Riceve un messaggio di ritorno in caso di incomprensione del messaggio inviato
	handleInform	msg	Riceve il messaggio coi dati richiesti allegati
	prepareRequests	msg	Invia il messaggio col predicato da riempire
GuiAgent			Gestisce gli eventi provenienti dall'interfaccia
AchieveREInitiator			Implementazione dei protocolli Fipa Request, Query e Propose
JFrame			La classe da cui ereditano le interfacce grafiche
CustomerFinder	CustomerFinder	platform name ownership	
	setup	guiEvent	Inizializzazione dell'agente
	register_WithAMS		Registrazione nel AMS
	register_WithDF		Registrazione nel DF
	onGuiEvent		Matching col eventi dalla GUI
	takeDown		Deregistrazione e chiusura agente
	Interfacing_with_Addetto_Ufficio_Clienti		Gestione GUI
CustomerFinderGui	CustomerFinderGui	a	Classe interfaccia
	initComponents		Inizializza l'interfaccia
	CancellaClientiSelezionati	evt	Raccoglie I dati dei clienti da cancellare
	cancellaEriscriviClienti		Aggiorna la tabella clienti.
	appiccicaAlPanel		Crea il pannello degli ordini
	SalvaModificheClienti	evt	Raccoglie i dati delle modifiche da fare sui clienti
	LeggiDatiClienti	evt	Invia il trigger per l'avvio del task dell'agente per la lettura dei dati dei clienti
	SalvaModificheOrdini	evt	Raccoglie i dati delle modifiche da fare sugli ordini
	StringToDate	stringa	Stringa da convertire in data
	cancellaOrdiniSelezionati	evt	Raccoglie I dati degli ordini da cancellare

	cancellaEriscriviOrdini	evt	Aggiorna la tabella degli ordini
	leggiDatiOrdini	evt	Invia il trigger per l'avvio del task dell'agente per la lettura degli ordini
	salvaDatiNuovoOrdine	evt	Salva il nuovo ordine
	pulisciDatiNuovoClient e	evt	Pulisce le caselle di inserimento dati nel pannello dei clienti
	SalvaiDatiNuovoClient e	evt	Salva il nuovo ordine
	exitForm	evt	Evento di chiusura dell'agente
	main	args[]	
	aggiornaTabellaClienti	listaClienti	Riscrittura della tabella coi nuovi dati dei clienti ricevuti
	aggiornaTabellaOrdini	listaOrdini	Riscrittura della tabella coi nuovi dati dei clienti ricevuti
	fromDateToString	date	
CompositeBehaviour			
FSMBehaviour			
Initiator			

Agent: OrderSwitcher

L'agente che si occupa della lottizzazione degli ordini e dell'invio dei lotti all'opportuno stabilimento



Attributes

Class	Attribute	Data Type	Description
QueryRefInitiatorRetriveLotsData			Prepara i messaggi per la richiesta di dati
RequestInitiatorSaveLotsData			Prepara i messaggi per l'invio di dati
RequestInitiatorContinueProcessChain_Lots_Stored	receiver	String	Prepara il messaggio di notifica inserimento nuovi lotti
Interfacing_with_Addetto_Produzione			Gestione degli eventi della GUI
RequestPartecipantContinueProcessChain_Start_Lotting	receiver	String	Riceve il messaggio di notifica inserimento ordini
AchieveREInitiator			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio di messaggi
JFrame			La classe da cui ereditano le interfacce grafiche
AchieveREResponder			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio dei dati di risposta
GuiAgent			Gestisce gli eventi provenienti dall'interfaccia
OrderSwitcher	AGENT_TYPE	String	
	version	String	
	processaordini	processaOrdini	Elemento dell'ontologia
	ordine	Ordine	Elemento dell'ontologia
	cliente	Cliente	Elemento dell'ontologia
	lotto	Lotto	Elemento dell'ontologia
	legame_ordini_lotti	Legame_Ordini_Lotti	Elemento dell'ontologia
	bicicletta	Bicicletta	Elemento dell'ontologia
	etichetta	Etichetta	Elemento dell'ontologia
	distinta	Distinta	Elemento dell'ontologia
	componente	Componente	Elemento dell'ontologia
	richiediordinidalottizzare	richiediOrdiniDaLottizzare	Elemento dell'ontologia
	salvalotti	salvaLotti	Elemento dell'ontologia
	schedulalotti	schedulaLotti	Elemento dell'ontologia
	lottizzatoA	boolean	Variabile di gestione
	lottizzatoB	boolean	Variabile di gestione
	quanteVolte	int	Variabile di gestione

	CLOSE_AGENT	int	Costante
	SEND_DATA_TO_WRAPPER	int	Costante
	RICHIEDI_ORDINI_DA_LOTTIZZARE	int	Costante
	SALVA_LOTTI	int	Costante
	MODIFICA_ORDINI_LOTTIZZATI	int	Costante
OrderSwitcherGui	DEBUG	boolean	
	vettoreIDOrdini	Vector	Vettore coi dati da visualizzare nell'interfaccia utente
	vettoreCommittenti	Vector	Vettore coi dati da visualizzare nell'interfaccia utente
	vettoreQuantitaLotti	Vector	Vettore coi dati da visualizzare nell'interfaccia utente
	vettoreGiorniProduzione	Vector	Vettore coi dati da visualizzare nell'interfaccia utente
	vettoreDate	Vector	Vettore coi dati da visualizzare nell'interfaccia utente
	vettoreIDLotti	Vector	Vettore coi dati da visualizzare nell'interfaccia utente
	righeModelloOrdini	int	Righe della tabella ordini del modello A
	righeModelloOrdiniB	int	Righe della tabella ordini del modello A
	aOb	int	Variabile per la gestione dell'interfaccia
FSMBehaviour			
CompositeBehaviour			
Initiator			

Methods

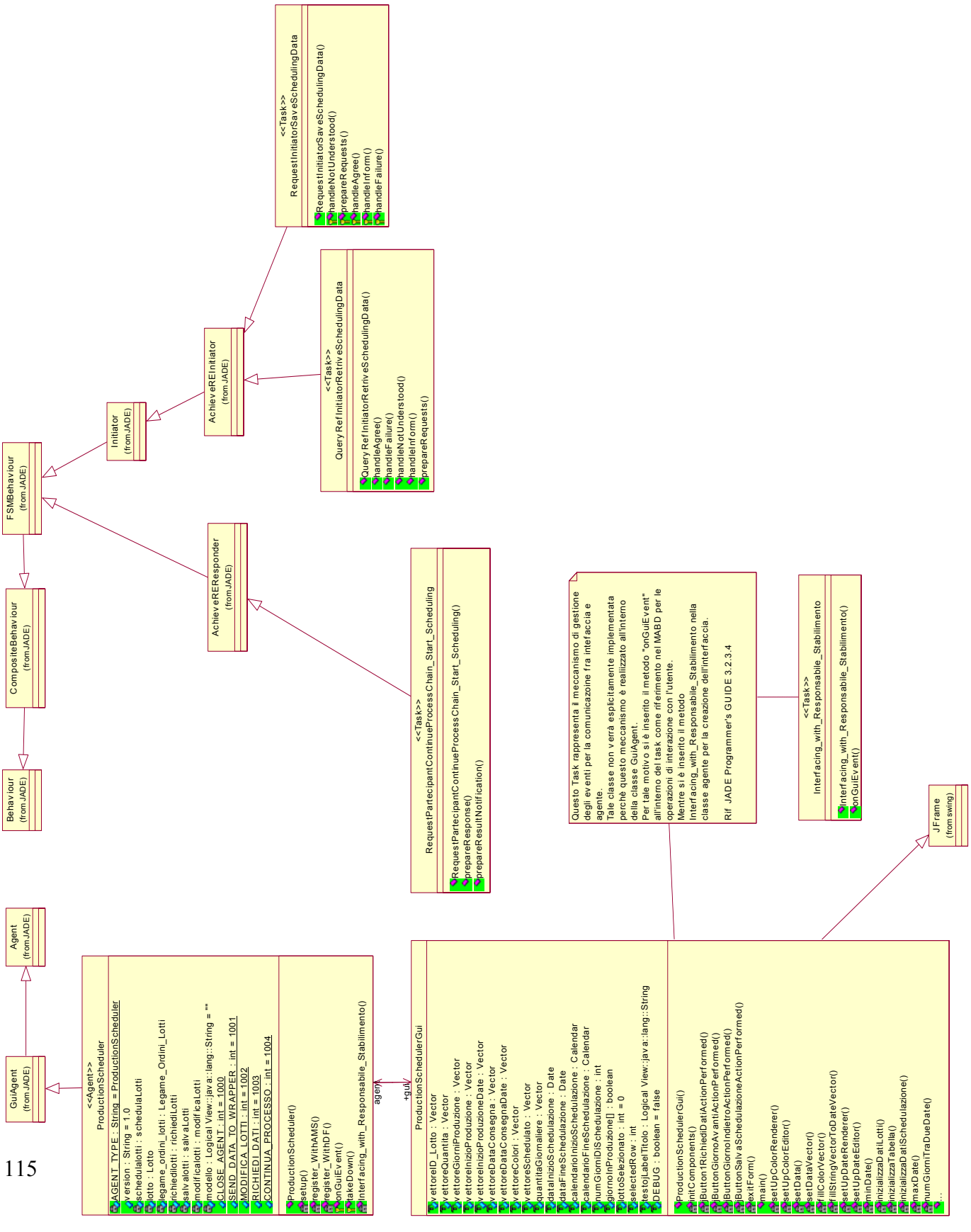
Class	Method	Parameters	Description
QueryRefInitiatorRetriveLotsData	QueryRefInitiatorRetriveLotsData		
	handleAgree	msg	Riceve un messaggio di accettazione della richiesta
	handleFailure	msg	Riceve un messaggio con l'informazione di fallimento dello svolgimento dell'azione richiesta

	handleNotUnderstood	msg	Riceve un messaggio di ritorno in caso di incomprensione del messaggio inviato
	handleInform	msg	Riceve il messaggio coi dati richiesti allegati
	prepareRequests	msg	Invia il messaggio col predicato da riempire
RequestInitiatorSaveLotsData	RequestInitiatorSaveLotsData		costruttore
	handleNotUnderstood	aCLMessage	Riceve un messaggio di ritorno in caso di incomprensione del messaggio inviato
	prepareRequests	msg	Prepara e invia il messaggio
	handleAgree	aCLMessage	Riceve un messaggio di accettazione della richiesta
	handleInform	aCLMessage	Riceve un messaggio con l'esito dell'azione svolta
	handleFailure	aCLMessage	Riceve un messaggio con l'informazione di fallimento dello svolgimento dell'azione richiesta
RequestInitiatorContinueProcessChain_Lots_Stored	RequestInitiatorContinueProcessChain_Lots_Stored		Costruttore
	handleNotUnderstood	aCLMessage	Riceve un messaggio di ritorno in caso di incomprensione del messaggio inviato
	prepareRequests	msg	Prepara e invia il messaggio
	handleAgree	aCLMessage	Riceve un messaggio di accettazione della richiesta
	handleInform	aCLMessage	Riceve un messaggio con l'esito dell'azione svolta
	handleFailure	aCLMessage	Riceve un messaggio con l'informazione di fallimento dello svolgimento dell'azione richiesta
Interfacing_with_Addetto_Produzione	Interfacing_with_Addetto_Produzione		costruttore
	onGuiEvent		Variabile che effettua il matching con gli eventi dell'interfaccia
RequestPartecipantContinueProcessChain_Start_Lotting	RequestPartecipantContinueProcessChain_Start_Lotting		Costruttore
		request	
	prepareResponse	request	Riceve il messaggio
	prepareResultNotification	response	Invia un messaggio di ricezione
AchieveREInitiator			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio dei dati

JFrame			La classe da cui ereditano le interfacce grafiche
AchieveREResponder			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio delle risposte coi dati
GuiAgent			Gestisce gli eventi provenienti dall'interfaccia
OrderSwitcher	OrderSwitcher	platform	Inizializzazione dell'agente
	setup	name	
	register_WithAMS	ownership	Registrazione nel AMS
	register_WithDF	guiEvent	Registrazione nel DF
	onGuiEvent		Matching col eventi dalla GUI
	takeDown		Deregistrazione e chiusura agente
	Interfacing_with_Addetto_Produzione		Gestione GUI
OrderSwitcherGui	OrderSwitcherGui	a	Classe interfaccia
	initComponents	evt	Inizializza l'interfaccia
	creaLotti		Creazione automatica dei lotti A
	leggiOrdiniDalDB	evt	Richiesta per la lettura degli ordini
	creaLottiB		Creazione automatica dei lotti B
	confermaInvioDati	evt	Invia i lotti prodotti
	leggiEcrea		Mostra la tabella per la creazione dei lotti anomali di A
	leggiEcreaB		Mostra la tabella per la creazione dei lotti anomali di B
	exitForm	evt	Evento di chiusura dell'agente
	aggiornaTabellaOrdini	listaOrdini	Riscrive i dati degli ordini letti
	fromDateToString	date	La data da convertire in stringa
	main	args[]	
FSMBehaviour			
CompositeBehaviour			
Initiator			

Agent: ProductionScheduler

L'agente che si occupa della schedulazione dei lotti di produzione



Attributes

Class	Attribute	Data Type	Description
RequestPartecipantContinueProcessChain_Start_Scheduling			Riceve il messaggio di notifica inserimento ordini
QueryRefInitiatorRetriveSchedulingData			Prepara i messaggi per la richiesta di dati
RequestInitiatorSaveSchedulingData			Prepara i messaggi per l'invio di dati
Interfacing_with_Responsabile_Stabilimento			Gestione degli eventi della GUI
AchieveREResponder			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio delle risposte coi dati
AchieveREInitiator			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio dei dati
GuiAgent			Gestisce gli eventi provenienti dall'interfaccia
JFrame			La classe da cui ereditano le interfacce grafiche
ProductionSchedulerGui	vettoreID_Lotto	Vector	variabile relativa ai lotti
	vettoreQuantita	Vector	variabile relativa ai lotti
	vettoreGiorniProduzione	Vector	variabile relativa ai lotti
	vettoreInizioProduzione	Vector	variabile relativa ai lotti
	vettoreInizioProduzioneDate	Vector	variabile relativa ai lotti
	vettoreDataConsegna	Vector	variabile relativa ai lotti
	vettoreDataConsegnaDate	Vector	variabile relativa ai lotti
	vettoreColori	Vector	variabile per la schedulazione
	vettoreSchedulato	Vector	variabile per la schedulazione
	quantitaGiornaliere	Vector	variabile per la schedulazione
	dataInizioSchedulazione	Date	variabile per la schedulazione
	dataFineSchedulazione	Date	variabile per la schedulazione
	calendarioInizioSchedulazione	Calendar	variabile per la schedulazione
	calendarioFineSchedulazione	Calendar	variabile per la schedulazione
	numGiorniDiSchedulazione	int	variabile per la schedulazione
	giornoInProduzione[]	Boolean	variabile per la schedulazione
	lottoSelezionato	int	
	selectedRow	int	variabile per la schedulazione

	testojLabelTitolo	String	variabile per la schedulazione
	DEBUG	boolean	variabile per la schedulazione
ProductionScheduler	AGENT_TYPE	String	
	version	String	
	schedulalotti	schedulaLotti	Elemento dell'ontologia
	lotto	Lotto	Elemento dell'ontologia
	legame_ordini_lotti	Legame_Ordini_Lotti	Elemento dell'ontologia
	richiedilotti	richiediLotti	Elemento dell'ontologia
	salvalotti	salvaLotti	Elemento dell'ontologia
	modificalotti	modificaLotti	Elemento dell'ontologia
	modello	String	
	CLOSE_AGENT	String	
	SEND_DATA_TO_WRAPPER	int	Costante
	MODIFICA_LOTTI	int	Costante
	RICHIEDI_DATI	int	Costante
	CONTINUA_PROCESSO		
CompositeBehaviour			
FSMBehaviour			
Initiator			

Methods

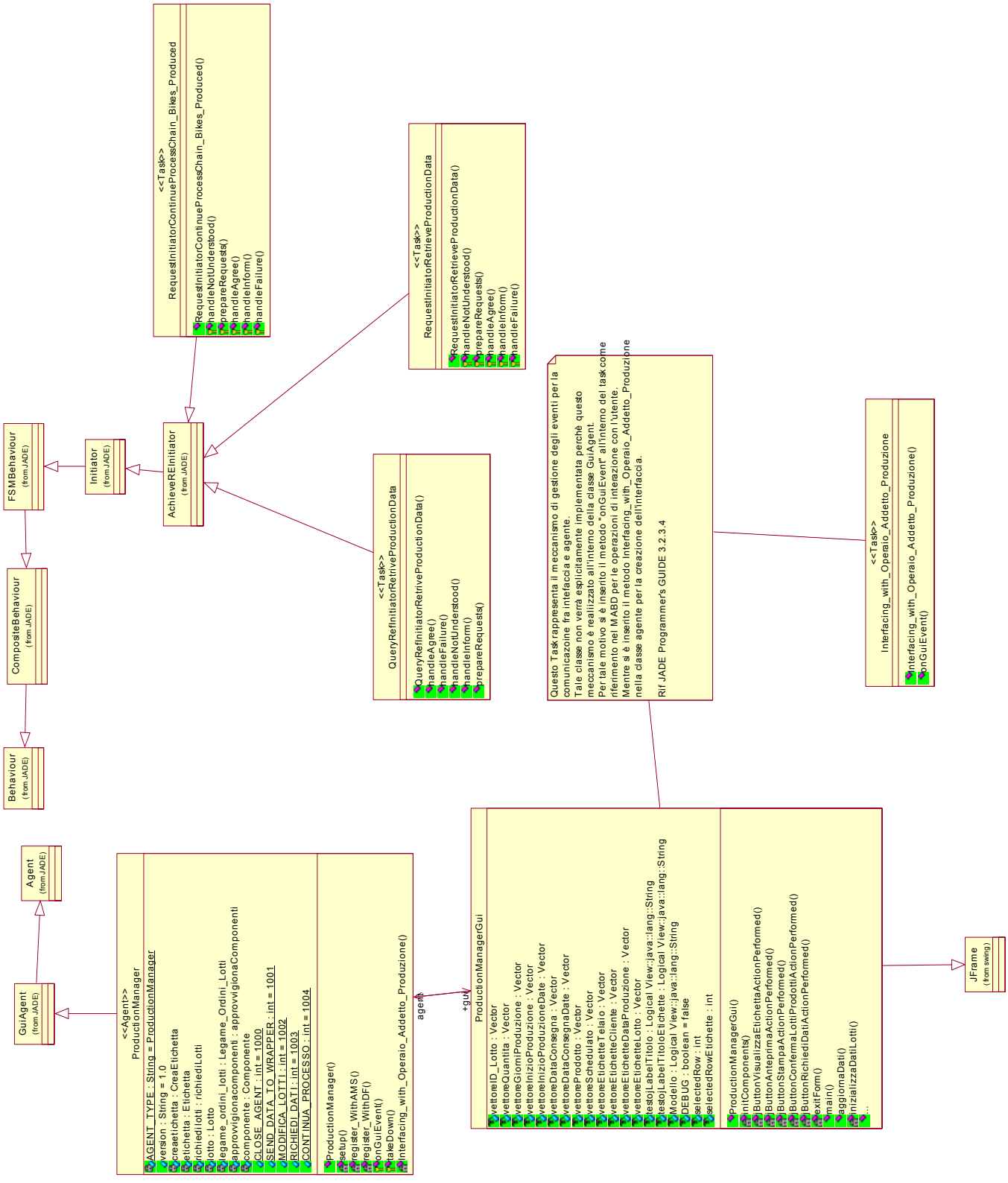
Class	Method	Parameters	Description
RequestPartecipantContinueProcessChain_Start_Scheduling	RequestPartecipantContinueProcessChain_Start_Scheduling	request	Costruttore
	prepareResponse	request	Riceve il messaggio
	prepareResultNotification	response	Invia un messaggio di ricezione
QueryRefInitiatorRetriveSchedulingData	QueryRefInitiatorRetriveSchedulingData		
	handleAgree	msg	Riceve un messaggio di accettazione della richiesta
	handleFailure	msg	Riceve un messaggio con l'informazione di fallimento dello svolgimento dell'azione richiesta
	handleNotUnderstood	msg	Riceve un messaggio di ritorno in caso di incomprensione del messaggio inviato

	handleInform	msg	Riceve il messaggio coi dati richiesti allegati
	prepareRequests	msg	Invia il messaggio col predicato da riempire
RequestInitiatorSaveSchedulingData	RequestInitiatorSaveSchedulingData		costruttore
	handleNotUnderstood	aCLMessage	Riceve un messaggio di ritorno in caso di incomprensione del messaggio inviato
	prepareRequests	msg	Prepara e invia il messaggio
	handleAgree	aCLMessage	Riceve un messaggio di accettazione della richiesta
	handleInform	aCLMessage	Riceve un messaggio con l'esito dell'azione svolta
	handleFailure	aCLMessage	Riceve un messaggio con l'informazione di fallimento dello svolgimento dell'azione richiesta
Interfacing_with_Responsabile_Stabilimento	Interfacing_with_Responsabile_Stabilimento		costruttore
	onGuiEvent		Variabile che effettua il matching con gli eventi dell'interfaccia
AchieveREResponder			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio delle risposte coi dati
AchieveREInitiator			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio dei dati
GuiAgent			Gestisce gli eventi provenienti dall'interfaccia
JFrame			La classe da cui ereditano le interfacce grafiche
ProductionSchedulerGui	ProductionSchedulerGui	a modello	Costruttore interfaccia
	initComponents		Inizializza la GUI

	jButton1RichiediDatiActionPerformed	evt	Legge I dati
	jButtonGiornoAvantiActionPerformed	evt	Sposta di un giorno il lotto nel grafico
	jButtonGiornoIndietroActionPerformed	evt	Sposta di un giorno il lotto nel grafico
	jButtonSalvaSchedulazioneActionPerformed	evt	Salva la schedulazione
	exitForm	evt	Chiude la GUI
	main	args[]	
	setUpColorRenderer	table	Variabile per il grafico
	setUpColorEditor	table	Variabile per il grafico
	setData	colorVector	Variabile per il grafico
	setDataVector	lenght	Variabile per il grafico
	fillColorVector	stringVector	Variabile per il grafico
	fillStringVectorToDateVector	dateVector	Variabile per il grafico
	setUpDateRenderer	table	Variabile per il grafico
	setUpDateEditor	table	Variabile per il grafico
	minDate	dateVector	Variabile per il grafico
	inizializzaDatiLotti		Variabile per il grafico
	inizializzaTabella		Variabile per il grafico
	inizializzaDatiSchedulazione	dateVector	Variabile per il grafico
	maxDate	calendar1	Variabile per il grafico
	numGiorniTraDueDate	calendar2	Variabile per il grafico
	selectLotto	sel	Variabile per il grafico
	aggiornaDati	richiedilotti	Variabile per il grafico
	setLabelState	string	Variabile per il grafico
ProductionScheduler	ProductionScheduler	platform	
	setup	name ownership	Inizializzazione dell'agente
	register_WithAMS		Registrazione nel AMS
	register_WithDF		Registrazione nel DF
	onGuiEvent	guiEvent	Matching col eventi dalla GUI
	takeDown		Deregistrazione e chiusura agente
	Interfacing_with_Responsabile_Stabilimento		Gestione GUI
CompositeBehaviour			
FSMBehaviour			
Initiator			

Agent: ProductionManager

L'agente che si occupa della conferma dei lotti prodotti e della creazione delle etichette da apporre nelle biciclette.



Attributes

Class	Attribute	Data Type	Description
QueryRefInitiatorRetrieveProductionData			Prepara i messaggi per la richiesta di dati
RequestInitiatorContinueProcessChain_Bikes_Produced			Prepara il messaggio di notifica inserimento nuovi lotti
Interfacing_with_Operaio_Addetto_Produzione			Gestione degli eventi della GUI
RequestInitiatorRetrieveProductionData			Prepara i messaggi per l'invio di dati
AchieveREInitiator			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio delle risposte coi dati
GuiAgent			Gestisce gli eventi provenienti dall'interfaccia
JFrame			La classe da cui ereditano le interfacce grafiche
ProductionManager	AGENT_TYPE	String	
	version	String	
	creaetichetta	CreaEtichetta	Elemento dell'ontologia
	etichetta	Etichetta	Elemento dell'ontologia
	richiedilotti	richiediLotti	Elemento dell'ontologia
	lotto	Lotto	Elemento dell'ontologia
	legame_ordini_lotti	Legame_Ordini_Lotti	Elemento dell'ontologia
	approvvigionacomponenti	approvvigionaComponenti	Elemento dell'ontologia
	componente	Componente	Elemento dell'ontologia
	CLOSE_AGENT	int	Costante
	SEND_DATA_TO_WRAPPER	int	Costante
	MODIFICA_LOTTI	int	Costante
	RICHIEDI_DATI	int	Costante
	CONTINUA_PROCESSO	int	Costante
ProductionManagerGui	vettoreID_Lotto	Vector	Vettore coi dati da visualizzare
	vettoreQuantita	Vector	Vettore coi dati da visualizzare
	vettoreGiorniProduzione	Vector	Vettore coi dati da visualizzare
	vettoreInizioProduzio	Vector	Vettore coi dati da

	ne		visualizzare
	vettoreInizioProduzione	Vector	Vettore coi dati da visualizzare
	vettoreDataConsegna	Vector	Vettore coi dati da visualizzare
	vettoreDataConsegnaDate	Vector	Vettore coi dati da visualizzare
	vettoreProdotto	Vector	Vettore coi dati da visualizzare
	vettoreSchedulato	Vector	Vettore coi dati da visualizzare
	vettoreEtichetteTelaio	Vector	Vettore coi dati da visualizzare
	vettoreEtichetteCliente	Vector	Vettore coi dati da visualizzare
	vettoreEtichetteDataProduzione	Vector	Vettore coi dati da visualizzare
	vettoreEtichetteLotto	Vector	Vettore coi dati da visualizzare
	testojLabelTitolo	String	
	testojLabelTitoloEtichette	String	
	Modello	String	Tipo bicicletta A o B
	DEBUG	boolean	
	selectedRow	int	
	selectedRowEtichette	int	

CompositeBehaviour
FSMBehaviour
Initiator

Methods

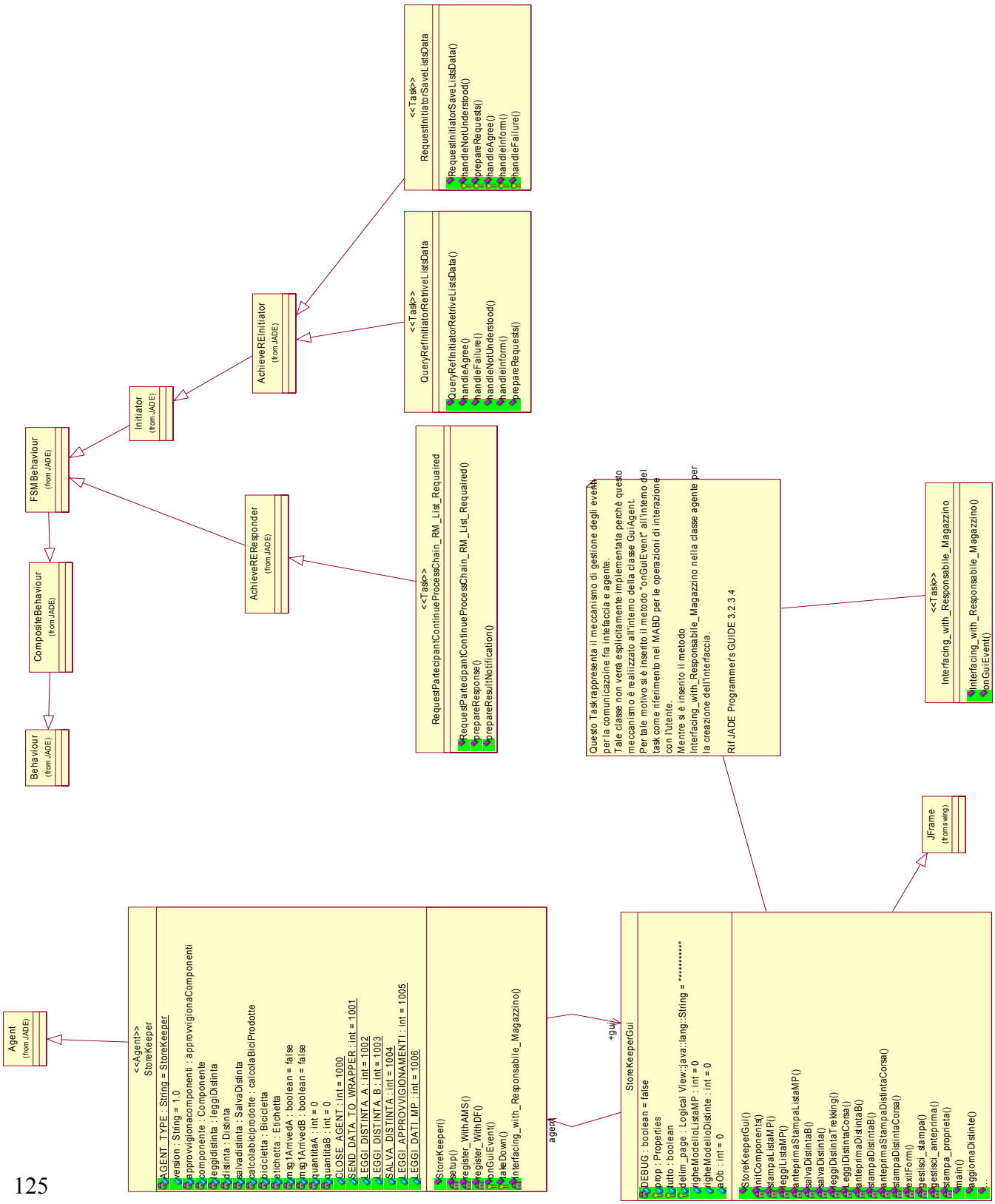
Class	Method	Parameters	Description
QueryRefInitiatorRetrieveProductionData	QueryRefInitiatorRetrieveProductionData		
	handleAgree	msg	Riceve un messaggio di accettazione della richiesta
	handleFailure	msg	Riceve un messaggio con l'informazione di fallimento dello svolgimento dell'azione richiesta
	handleNotUnderstood	msg	Riceve un messaggio di ritorno in caso di incomprensione del
	handleInform	msg	messaggio inviato Riceve il messaggio coi dati

	prepareRequests	msg	richiesti allegati Invia il messaggio col predicato da riempire
RequestInitiatorContinueProcessChain_Bikes_Produced	RequestInitiatorContinueProcessChain_Bikes_Produced		Costruttore
	handleNotUnderstood	aCLMessage	Riceve un messaggio di ritorno in caso di incomprensione del messaggio inviato
	prepareRequests	msg	Prepara e invia il messaggio
	handleAgree	aCLMessage	Riceve un messaggio di accettazione della richiesta
	handleInform	aCLMessage	Riceve un messaggio con l'esito dell'azione svolta
	handleFailure	aCLMessage	Riceve un messaggio con l'informazione di fallimento dello svolgimento dell'azione richiesta
Interfacing_with_Operaio_Addetto_Produzione	Interfacing_with_Operaio_Addetto_Produzione		costruttore
	onGuiEvent		Variabile che effettua il matching con gli eventi dell'interfaccia
RequestInitiatorRetrieveProductionData	RequestInitiatorRetrieveProductionData	a act	Costruttore
	handleNotUnderstood	aCLMessage	Riceve un messaggio di ritorno in caso di incomprensione del messaggio inviato
	prepareRequests	msg	Prepara e invia il messaggio
	handleAgree	aCLMessage	Riceve un messaggio di accettazione della richiesta
	handleInform	msg	Invia un messaggio di avvenuta ricezione
	handleFailure	aCLMessage	Riceve un messaggio con l'informazione di fallimento dello svolgimento dell'azione richiesta
AchieveREInitiator			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio dei dati
GuiAgent			Gestisce gli eventi

			provenienti dall'interfaccia
JFrame			La classe da cui ereditano le interfacce grafiche
ProductionManager	ProductionManager	platform	
	setup	name	Inizializzazione dell'agente
	register_WithAMS	ownership	Registrazione nel AMS
	register_WithDF	guiEvent	Registrazione nel DF
	onGuiEvent		Matching col eventi dalla GUI
	takeDown		Deregistrazione e chiusura agente
	Interfacing_with_Operaio_Addetto_Produzione		Gestione GUI
ProductionManagerGui	ProductionManagerGui	a	
	initComponents	modello	Inizializzazione della GUI
	jButtonVisualizzaEtichettaActionPerformed	evt	Visualizza le etichette
	jButtonAnteprimaActionPerformed	evt	Visualizza l'anteprima di stampa
	jButtonStampaActionPerformed	evt	Stampa l'etichetta
	jButtonConfermaLottiProdottiActionPerformed	evt	Invia i dati dei lotti prodotti
	jButtonRichiediDatiActionPerformed	evt	Richiede i dati dal database
	exitForm	evt	Chiusura dell'interfaccia
	main	args[]	
	aggiornaDati	richiedilotti	Riscrive la tabella
	inizializzaDatiLotti	creaetichettaj	Crea una nuova tabella con le etichette
	aggiornaDatiEtichette	TableStampa	Aggiorna la tabella delle etichette
	gestisci_anteprima	jTableStampa	Gestisce l'anteprima
	gestisci_stampa		Gestisce la stampa
CompositeBehaviour			
FSMBehaviour			
Initiator			

Agent: StoreKeeper

L'agente che si occupa della creazione della lista per l'approvvigionamento delle materie prime e della modifica delle distinte di produzione.



Attributes

Class	Attribute	Data Type	Description
RequestPartecipantContinueProcessChain_RM_List_Requaired			Riceve il messaggio di notifica avvenuta produzione di nuove biciclette
QueryRefInitiatorRetriveListsData			Prepara i messaggi per la richiesta di dati
RequestInitiatorSaveListsData			Prepara i messaggi per l'invio di dati
Interfacing_with_Responsabile_Magazzino			Gestione degli eventi della GUI
JFrame			La classe da cui ereditano le interfacce grafiche
AchieveREResponder			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio dei dati di risposta
AchieveREInitiator			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio di messaggi
StoreKeeper	AGENT_TYPE	String	
	version	String	
	approvvigionacomponenti	approvvigionaComponenti	Elemento dell'ontologia
	componente	Componente	Elemento dell'ontologia
	leggidistinta	leggiDistinta	Elemento dell'ontologia
	distinta	Distinta	Elemento dell'ontologia
	salvadistinta	SalvaDistinta	Elemento dell'ontologia
	calcolabiciprodotte	calcolaBiciProdotte	Elemento dell'ontologia
	bicicletta	Bicicletta	Elemento dell'ontologia
	etichetta	Etichetta	Elemento dell'ontologia
	msg1ArrivedA	boolean	Flag di servizio
	msg1ArrivedB	boolean	Flag di servizio
	quantitaA	int	Variabile di utilizzo
	quantitaB	int	Variabile di utilizzo
	CLOSE_AGENT	int	Costante
	SEND_DATA_TO_WRAPPER	int	Costante
	LEGGI_DISTINTA_A	int	Costante
	LEGGI_DISTINTA_B	int	Costante
	SALVA_DISTINTA	int	Costante
	LEGGI_APPROVVIGIONAMENTI	int	Costante
	LEGGI_DATI_MP		Costante

StoreKeeperGui	DEBUG	boolean	
	prop	Properties	Cattura il frame da stampare
	tutto	boolean	Dice se stampare tutto o solo il testo
	delim_page	String	Delimitatore di pagina
	righeModelloListaMP	int	Per la gestione della tabella delle MP
	righeModelloDistinte	int	Per la gestione della tabella delle distinte
	aOb	int	Variabile di utilizzo
CompositeBehaviour			
FSMBehaviour			
Initiator			

Methods

Class	Method	Parameters	Description
RequestPartecipantContinueProcessChain_RM_List_Requaired	RequestPartecipantContinueProcessChain_RM_List_Requaired	request	Costruttore
	prepareResponse	request	Riceve il messaggio
	prepareResultNotification	response	Invia un messaggio di ricezione
QueryRefInitiatorRetriveListsData	QueryRefInitiatorRetriveListsData		Costruttore
	handleAgree	msg	Riceve un messaggio di accettazione della richiesta
	handleFailure	msg	Riceve un messaggio con l'informazione di fallimento dello svolgimento dell'azione richiesta
	handleNotUnderstood	msg	Riceve un messaggio di ritorno in caso di incomprensione del messaggio inviato
	handleInform	msg	Riceve il messaggio coi dati richiesti allegati
	prepareRequests	msg	Invia il messaggio col predicato da riempire

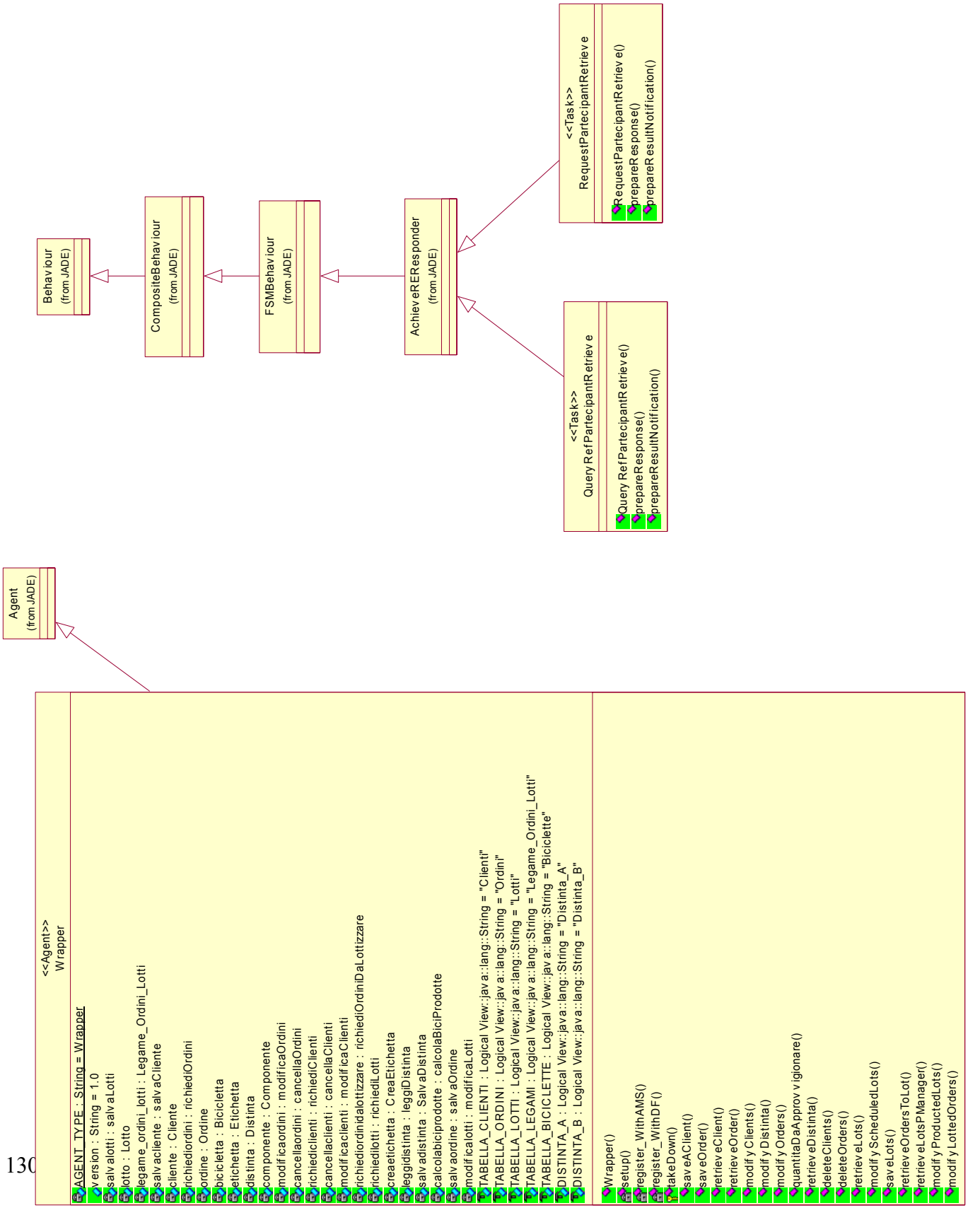
RequestInitiatorSaveListsData	RequestInitiatorSaveListsData		costruttore
	handleNotUnderstood	aCLMessage	Riceve un messaggio di ritorno in caso di incomprensione del messaggio inviato
	prepareRequests	msg	Prepara e invia il messaggio
	handleAgree	aCLMessage	Riceve un messaggio di accettazione della richiesta
	handleInform	aCLMessage	Riceve un messaggio con l'esito dell'azione svolta
	handleFailure	aCLMessage	Riceve un messaggio con l'informazione di fallimento dello svolgimento dell'azione richiesta
Interfacing_with_Responsabile_Magazzino	Interfacing_with_Responsabile_Magazzino		costruttore
	onGuiEvent		Variabile che effettua il matching con gli eventi dell'interfaccia
JFrame			La classe da cui ereditano le interfacce grafiche
AchieveREResponder			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio delle risposte coi dati
AchieveREInitiator			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio dei dati
StoreKeeper	StoreKeeper	platform	Inizializzazione dell'agente
	setup	name	
		ownership	
	register_WithAMS		Registrazione nel AMS
	register_WithDF		Registrazione nel DF
	onGuiEvent	guiEvent	Matching con gli eventi dalla GUI
	takedown		Deregistrazione e chiusura agente
	Interfacing_with_Responsabile_Magazzino		Gestione GUI

StoreKeeperGui	StoreKeeperGui	a	Puntatore all'agente
	initComponents		Inizializza la GUI
	stampaListaMP	evt	Stampa la lista approv
	leggiListaMP	evt	Legge le distinte
	anteprimaStampaListaMP	evt	Apre l'anteprima di stampa della lista MP
	salvaDistintaB	evt	Salva la distinta B
	salvaDistinta	evt	Salva la distinta A
	leggiDistintaTrekking	evt	Legge la distinta B
	LeggiDistintaCorsa	evt	Legge la distinta A
	anteprimaDistintaB	evt	Apre l'anteprima di stampa della distinta B
	stampaDistintaB	evt	Stampa la distinta B
	anteprimaStampaDistintaCorsa	evt	Apre l'anteprima di stampa della distinta A
	stampaDistintaCorsa	evt	Stampa la distinta A
	exitForm	evt	Chiusura della GUI
	gestisci_stampa	jTableStampa	Gestione della stampa
	gestisci_anteprima	jTableStampa	Gestione dell'anteprima
	stampa_proprieta	prop	Stampa l'oggetto
	main	args[]	
	aggiornaDistinte	distinta	Riscrive i dati della tabella distinta
	aggiornaListaMP	QntApprA QntApprB	Crea la lista degli approvvigionamenti

CompositeBehaviour
FSMBehaviour
Initiator

Agent: Wrapper

L'agente che si interfaccia al database.



Attributes

Class	Attribute	Data Type	Description
Wrapper	AGENT_TYPE	String	
	version	String	
	salvalotti	salvaLotti	Elemento dell'ontologia
	lotto	Lotto	Elemento dell'ontologia
	legame_ordini_lotti	Legame_Ordini_Lotti	Elemento dell'ontologia
	salvacliente	salvaCliente	Elemento dell'ontologia
	cliente	Cliente	Elemento dell'ontologia
	richiediordini	richiediOrdini	Elemento dell'ontologia
	ordine	Ordine	Elemento dell'ontologia
	bicicletta	Bicicletta	Elemento dell'ontologia
	etichetta	Etichetta	Elemento dell'ontologia
	distinta	Distinta	Elemento dell'ontologia
	componente	Componente	Elemento dell'ontologia
	modificaordini	modificaOrdini	Elemento dell'ontologia
	cancellaordini	cancellaOrdini	Elemento dell'ontologia
	richiediclienti	richiediClienti	Elemento dell'ontologia
	cancellaclienti	cancellaClienti	Elemento dell'ontologia
	modificaclienti	modificaClienti	Elemento dell'ontologia
	richiediordinidalottizzare	richiediOrdiniDaLottizzare	Elemento dell'ontologia
	richiedilotti	richiediLotti	Elemento dell'ontologia
	creaetichetta	CreaEtichetta	Elemento dell'ontologia
	leggidistinta	leggiDistinta	Elemento dell'ontologia
	salvadistinta	SalvaDistinta	Elemento dell'ontologia
	calcolabiciprodotte	calcolaBiciProdotte	Elemento dell'ontologia
	salvaordine	salvaOrdine	Elemento dell'ontologia
	modificalotti	modificaLotti	Utilizzata nelle query
	TABELLA_CLIENTI	String	Utilizzata nelle query
	TABELLA_ORDINI	String	Utilizzata nelle query
	TABELLA_LOTTI	String	Utilizzata nelle query
	TABELLA_LEGAMI	String	Utilizzata nelle query
	TABELLA_BICICLETTE	String	Utilizzata nelle query
	DISTINTA_A	String	Utilizzata nelle query
	DISTINTA_B	String	
QueryRefParticipantRetrieve			Svolge le richieste di prelevamento dei dati da parte degli altri agenti ed invia il risultato
RequestParticipantRetrieve			Svolge le richieste di memorizzazione dei dati da parte degli altri agenti ed invia un messaggio di conferma

AchieveREResponder	Implementazione dei protocolli Fipa Request, Query e Propose per l'invio delle risposte coi dati
CompositeBehaviour	
FSMBehaviour	

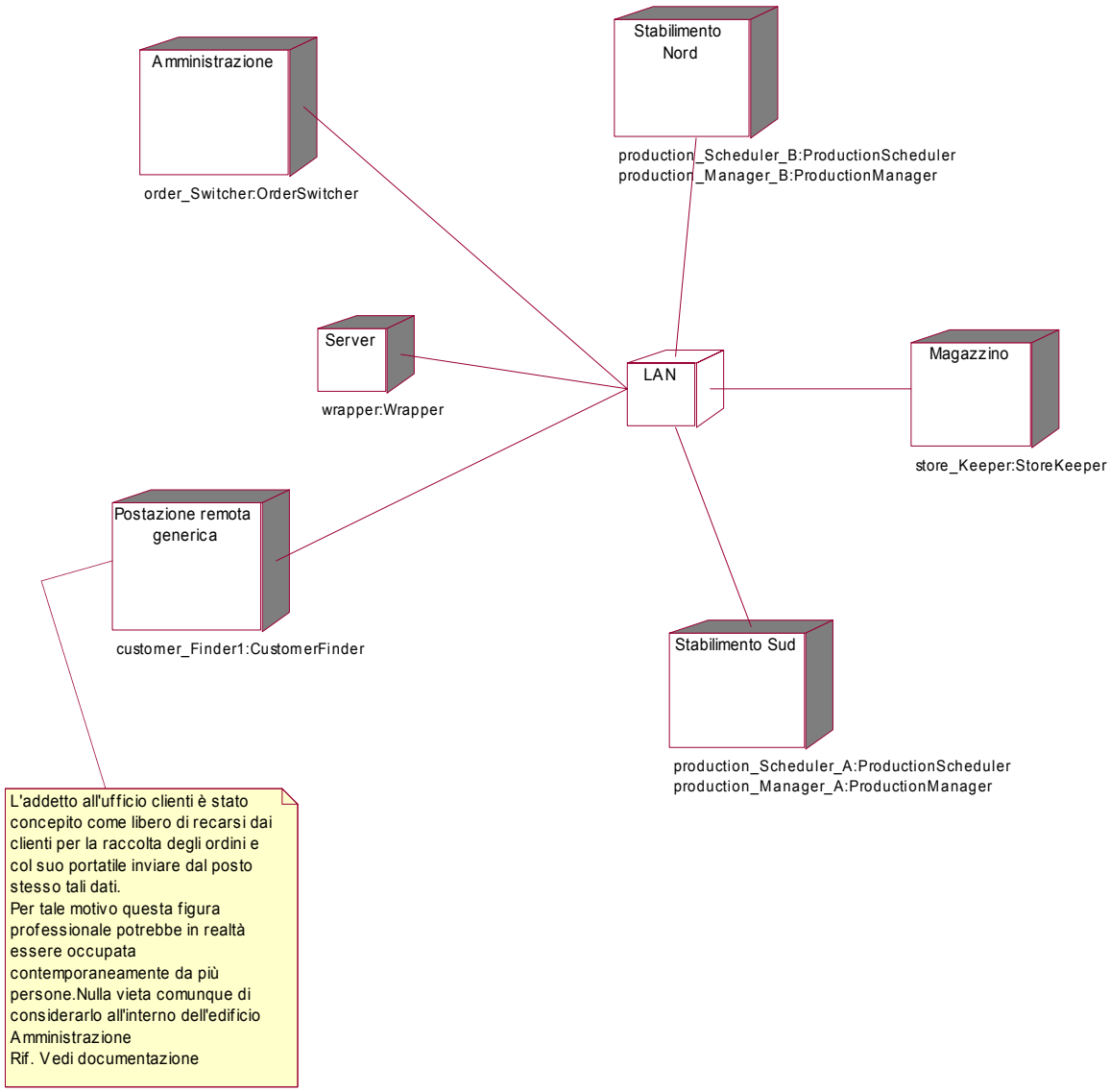
Methods

Class	Method	Parameters	Description
Wrapper	Wrapper	platform	
	setup	name	
		ownership	
	register_WithAMS		
	register_WithDF		
	takeDown		
	saveAClient	svClt	Salva i dati ricevuti dei clienti
	saveOrder	svOrd	Salva i dati ricevuti dell'ordine
	retrieveClient	reqClt	Legge i dati dei clienti
	retrieveOrder	reqOrd	Legge i dati degli ordini
	modifyClients	modificaclienti	Modifica i dati dei clienti
	modifyDistinta	salvadistinta	Modifica i dati delle distinte
	modifyOrders	modificaordini	Modifica i dati degli ordini
	quantitaDaApprovvigionare	calcolabiciprodotte	Calcola il numero di biciclette prodotte dall'ultimo approvvigionamento dei materiali
	retrieveDistinta	legDis	Legge i dati delle distinte
	deleteClients	cancellaclienti	Cancella i dati dei clienti
	deleteOrders	cancellaordini	Cancella i dati degli ordini
	retrieveLots	richiedilotti	Legge i dati dei lotti
	modifyScheduledLots	modificalotti	Modifica i dati dei lotti
	saveLots	salvalotti	Salva i dati ricevuti dei lotti
	retrieveOrdersToLot	richiediOrdini	Legge gli ordini che devono essere lottizzati
	retrieveLotsPManager	richiedilotti	Legge i lotti che non sono stati ancora prodotti
	modifyProductedLots	modificalotti	Modifica i dati dei lotti
	modifyLottedOrders	creaetichetta	Crea le etichette per i lotti prodotti
QueryRefParticipantRetrieve	QueryRefPartecipantRetrieve	request	Costruttore
	prepareResponse	equest	Riceve la richiesta, la interpreta e avvia la query relativa
	prepareResultNotification	response	Invia il risultato della query

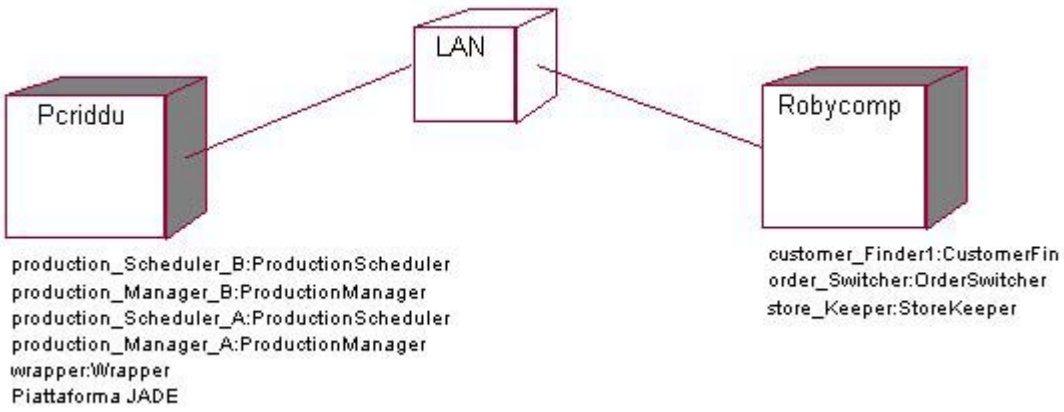
RequestParteciPantRetrieve	RequestPartecipantRetrieve	request	Costruttore
	prepareResponse	request	Riceve la richiesta coi dati da salvare, la interpreta e avvia la select relativa
	prepareResultNotification	response	Invia il risultato della operazione
AchieveREResponder			Implementazione dei protocolli Fipa Request, Query e Propose per l'invio delle risposte coi dati
CompositeBehaviour FSMBehaviour			

Deployment Configuration phase

In questo diagramma vengono mostrate le locazioni fisiche degli stabilimenti di produzione e quali agenti dovranno esistere all'interno di essi. Come da specifiche di progetto, sono stati concepiti come edifici distinti uno per il settore amministrativo e due stabilimenti, Nord e Sud, per quello produttivo. La figura dell'Addetto all'ufficio clienti è stata concepita come non fissa in un luogo ma libera di muoversi alla ricerca di nuovi clienti e con la possibilità di interfacciarsi alla rete tramite desktop; il magazzino è stato concepito come edificio a sé stante per una più semplice gestione amministrativa dei fornitori e il server è stato immaginato come localizzato anch'esso in un edificio estraneo agli altri. Sia ben chiaro comunque che tali considerazioni aumentano solamente i gradi di libertà del sistema e non sono per nulla restrittivi dal momento che è possibile accorparli in uno o nell'altro edificio a piacimento del committente senza andare ad inficiare sulle funzionalità del sistema. Unico accorgimento che invece deve essere preso è quello di avere una piattaforma jade perennemente attiva nel server ed un agente Wrapper sempre vivo in essa e, naturalmente, una rete LAN efficiente per l'ottimizzazione delle comunicazioni tra gli agenti


Amministrazione
order_Switcher:OrderSwitcher
Stabilimento Nord
production_Scheduler_B:ProductionScheduler
production_Manager_B:ProductionManager
Server
wrapper:Wrapper
LAN
Magazzino
store_Keeper:StoreKeeper
Postazione remota generica
customer_Finder1:CustomerFinder
Stabilimento Sud
production_Scheduler_A:ProductionScheduler
production_Manager_A:ProductionManager
L'addetto all'ufficio clienti è stato concepito come libero di recarsi dai clienti per la raccolta degli ordini e col suo portatile inviare dal posto stesso tali dati.
Per tale motivo questa figura professionale potrebbe in realtà essere occupata contemporaneamente da più persone.Nulla vieta comunque di considerarlo all'interno dell'edificio Amministrazione
Rif. Vedi documentazione

Di seguito è mostrato il diagramma che rispecchia la configurazione che verrà utilizzata nella presentazione del software al committente.



Amministrazione

- **order_Switcher:OrderSwitcher**

Stabilimento Nord

- **production_Scheduler_B:ProductionScheduler**
- **production_Manager_B:ProductionManager**

Stabilimento Sud

- **production_Scheduler_A:ProductionScheduler**
- **production_Manager_A:ProductionManager**

Magazzino

- **store_Keeper:StoreKeeper**

Server

- **wrapper:Wrapper**

Postazione remota generica

- **customer_Finder1:CustomerFinder**

PROGETTODATABASE

Sistema BikesAgent

Redatto da:	Caico Roberto Gargagliano Giuseppe Termine Francesco	_____	_____
Emesso da:	Caico Roberto Gargagliano Giuseppe Termine Francesco	_____	_____
Approvato da:	Caico Roberto Gargagliano Giuseppe Termine Francesco	_____	_____

Codice documento: PDB1 .01 .01
Commessa *Num. Doc.* *Vers.*

Prima emissione: 12/04/2004

Distribuzione: Interna

Data di stampa: 18/04/04 18.25

Path del file: *C:\documenti\tesina*

La validità di questa informazione è garantita al momento della stampa del documento.

Introduzione

Lo scopo di questo documento è di dare tutte le informazioni possibili sulla struttura del DataBase che si è usato per la gestione dei dati del sistema BikesAgent. Verrà dunque descritto il Progetto Concettuale, il progetto logico ed infine lo schema fisico.
Si è deciso di gestire i dati dell'azienda produttrice di Biciclette in Microsoft Access essendo questo il più semplice DBMS di tipo relazionale in commercio. Per la realizzazione dei diagrammi si è utilizzato Microsoft Visio.

Definizioni

Acronimo / Termine	*Significato*
Entità o Tabella	Un qualsiasi oggetto concettuale che caratterizza l'applicazione in questione e che può essere individuato e distinto dagli altri.
Attributo	Insieme di valori che caratterizzano un'entità.
Attributo Chiave	Insieme degli attributi sufficienti ad identificare univocamente un'entità all'interno di un certo insieme.
Relazione	Collegamento Logico tra entità.
Occorrenza	Valore dell'entità.

Progetto Concettuale

Descrizione del Progetto Concettuale

Il sistema BikesAgent richiede un database nel quale archiviare le informazioni riguardanti la gestione di un'azienda che si occupa di produrre due modelli differenti di biciclette, un modello da corsa ed uno da trekking (Tipo A e il Tipo B). Tutte le informazioni che verranno memorizzate nella struttura dati riguarderanno i clienti, i tipi di biciclette e tutte quelle informazioni che si andrà successivamente a dettagliare per la gestione del processo produttivo.

Per la rappresentazione del database si utilizza il modello relazionale. In un database relazionale, quale è Microsoft Access, si distinguono le seguenti componenti dell'archivio: le entità, gli attributi e le relazioni. Le entità rappresentano classi di oggetti che hanno proprietà comuni ad esistenza "autonoma" ai fini dell'applicazione di interesse; nello specifico campo della produzione delle biciclette, ad esempio, si possono avere Entità come: CLIENTI, COMPONENTI, DISTINTA. All'interno dell'entità si possono definire le occorrenze che rappresentano il valore delle stesse. Così ad esempio Mario Rossi può definire un'occorrenza dell'entità CLIENTI. In particolare le entità nel DBMS Microsoft Access prendono il nome di tabelle, in questa documentazione si farà dunque riferimento alla Tabella come ad una entità. Le relazioni rappresentano invece legami logici significativi per l'applicazione di interesse tra due o più entità. Ad esempio costituita da può rappresentare la relazione tra DISTINTA e COMPONENTI. Le tabelle sono connesse da legami che ne rappresentano le relazioni fisiche. I legami si possono dividere i tre categorie:

legami di tipo "1" a "1" dove ad un'occorrenza di una entità corrisponderà una e una sola occorrenza di un'altra entità;

legami di tipo "1" a "n" dove ad un'occorrenza di una entità corrisponderanno "n" occorrenze di un'altra entità;

legami di tipo "n" a "n" dove ad "n" occorrenze di una entità corrisponderanno "n" occorrenze di un'altra entità;

Infine gli attributi descrivono le proprietà elementari di entità o relazioni che sono di interesse ai fini dell'applicazione. Per esempio Cognome, Nome, Indirizzo sono possibili attributi dell'entità CLIENTI.

Tipi particolari di attributi sono le Chiavi Primarie che sono attributi che identificano in maniera univoca l'occorrenza dell'entità che si sta considerando. Ad esempio nel caso di CLIENTI un

attributo identificativo esclusivamente di un cliente che potrà essere definito come Chiave Primaria sarà sicuramente il NomeDitta.

La progettazione concettuale di una base di dati consiste nella costruzione di una schema Entità-Relazione in grado di descrivere al meglio le specifiche sui dati di un'applicazione.

Per la realizzazione del progetto concettuale si è scelta la strategia di tipo Top-Down secondo la quale lo schema concettuale viene prodotto mediante una serie di raffinamenti successivi a partire da una uno schema iniziale che descrive tutte le specifiche con pochi concetti molto astratti.

Descrizione dell'archivio

In questa fase procediamo alla descrizione dell'archivio ed identifichiamo i principali elementi che in esso si relazionano. Indicheremo di colore rosso le entità , e di colore verde gli attributi delle entità.

Nella entità BICICLETTE dovranno essere memorizzati i dati identificativi delle biciclette prodotte differenziati per modello. Si avrà dunque la seguente rappresentazione:

BICICLETTE(Telaio, DatiProduzione, Modello).

Nella entità CLIENTI saranno invece memorizzati i dati anagrafici di ogni singolo cliente che richiederà la fabbricazione delle biciclette prodotte nei nostri stabilimenti. Si avrà dunque la seguente rappresentazione:

CLIENTI(NomeDitta,NomeContatto,CognomeContatto,IndirizzoFatturazione,Citta,StatoOProvincia,CAP,NumeroTelefonico,NumeroFax,IndirizzoPostaElettronica,Varie). Si sottolinea che l'attributo Varie rappresenta le eventuali Note che possono essere memorizzate del singolo Cliente; si è deciso di scegliere tale identificativo del campo tenendo conto che la descrizione "Note" è una parola riservata di ACCESS e che quindi può creare problemi in fase di inserimento, modifica o cancellazione dei dati.

Nella entità COMPONENTI saranno memorizzate tutti i componenti primari e le materie prime necessari per la fabbricazione dei due tipi di Biciclette da produrre nei due stabilimenti. Si avrà dunque la seguente rappresentazione: COMPONENTI(IdComponente,NomeComponente).

Nella entità DISTINTA sarà memorizzato l'elenco dei componenti e delle quantità di questi ultimi necessarie per la produzione dei due modelli di biciclette; si avrà dunque la seguente rappresentazione DISTINTA(ID_Distinta, ID_Componente, Quantità,Modello).

I lotti prodotti saranno memorizzati nell' entità LOTTI così definita:

LOTTI(IDLotto,TipoLotto,Quantita,Giorni_Produzione,Inizio_Produzione,Data_Consegna,Schedulato,Prodotto,Approvviggonamento). Si sottolinea come in fase di schedulazione continui ad essere utilizzata tale entità per la memorizzazione dei dati ed in particolare gli attributi Inizio_Produzione dal quale viene settata la data di inizio produzione del lotto e Schedulato che è un flag che permette di identificare i lotti già schedulati da quelli che ancora non lo sono.

I differenti tipi di Biciclette prodotti con le loro rispettive descrizioni saranno memorizzati nell' entità MODELLI che sarà così rappresentata: MODELLI(Modello,Descrizione).

Gli ordini richiesti dai clienti saranno memorizzati nell'entità ORDINI che sarà così rappresentata: ORDINI(IDOrdine,NomeDitta,DataOrdine,Quantita_A,ConsegnaEntro_A,Quantita_B,ConsegnaEntroB,Lottizzato).

ERD - Diagramma Entità Relazioni

Descrizione del Diagramma ERD

Dalla descrizione del sistema vista precedentemente nella fase di progetto concettuale si possono trarre i diagrammi sotto riportati che indicano le entità e gli attributi che le costituiscono. Nello specifico verranno indicate con un rettangolo le entità e con i cerchi gli attributi. Sarà inoltre evidenziato in grassetto-sottolineato l'attributo chiave dell'entità, che distinguerà in maniera univoca le occorrenze in essa memorizzate.

Rappresentazione del Diagramma ERD

Tabella Biciclette

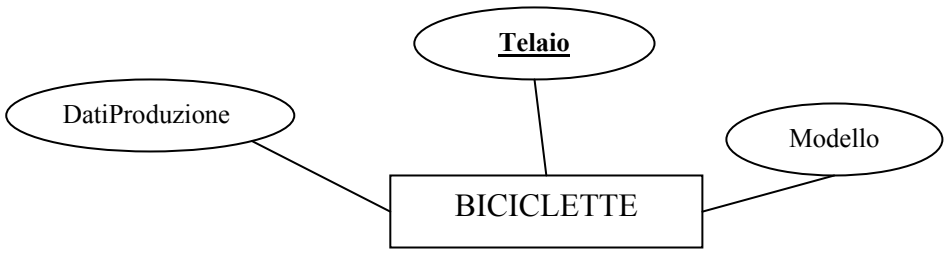


Tabella Clienti

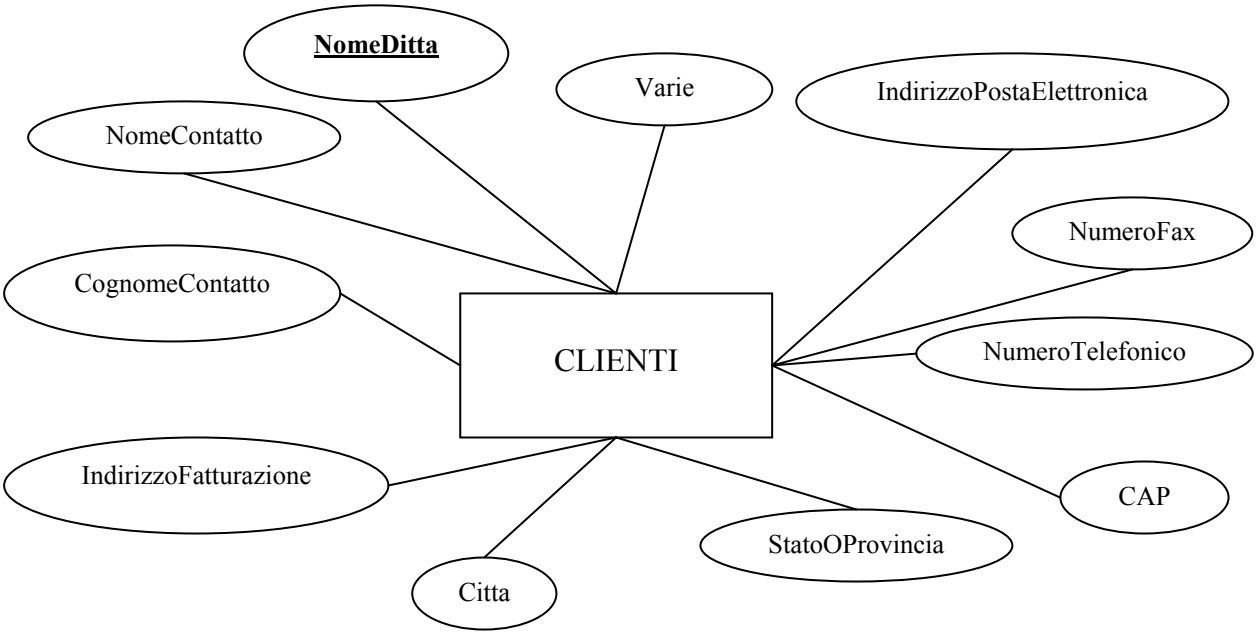


Tabella Componenti

Tabella Distinta

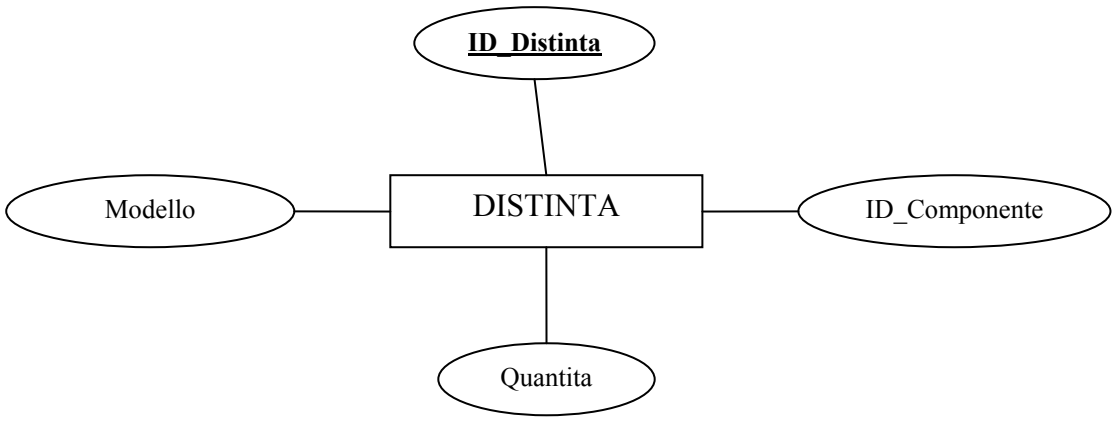


Tabella Legame_Ordini_Lotti

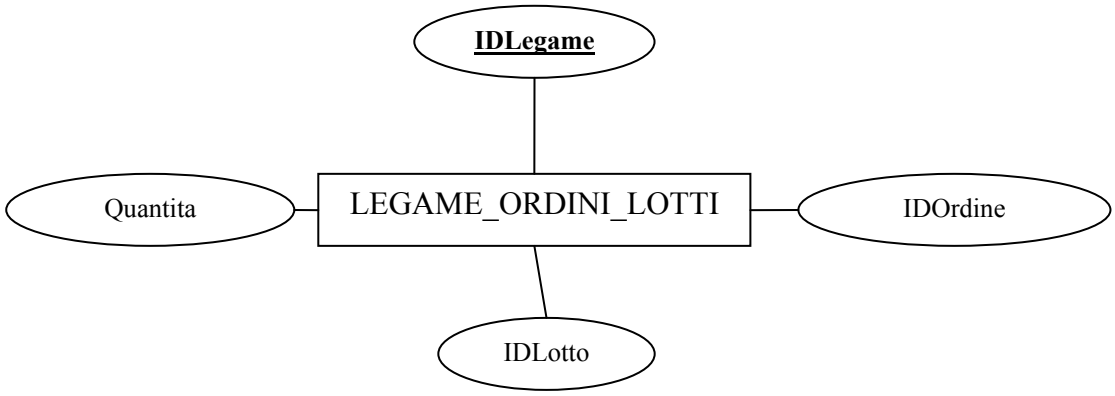


Tabella Lotti

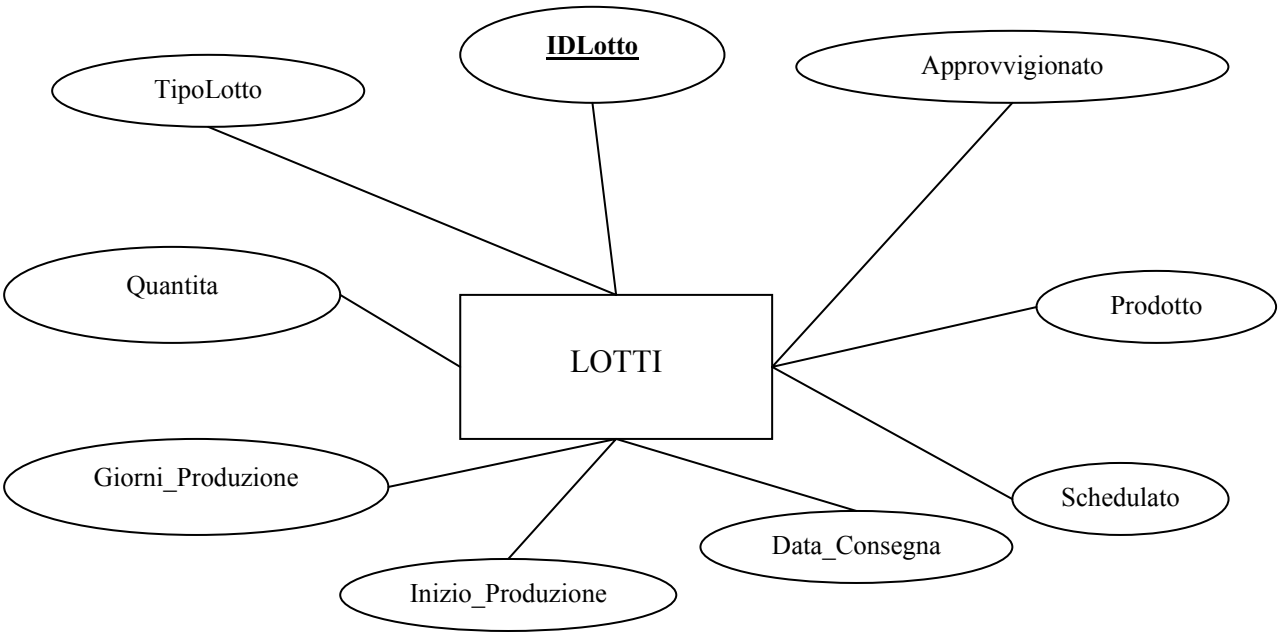
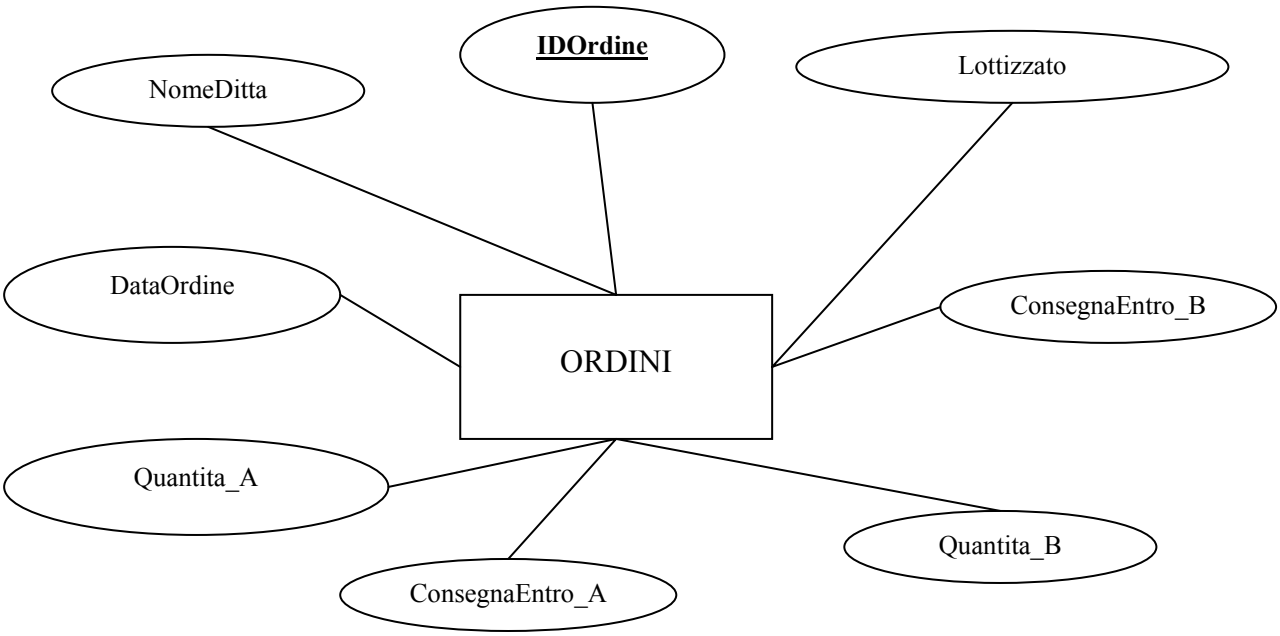
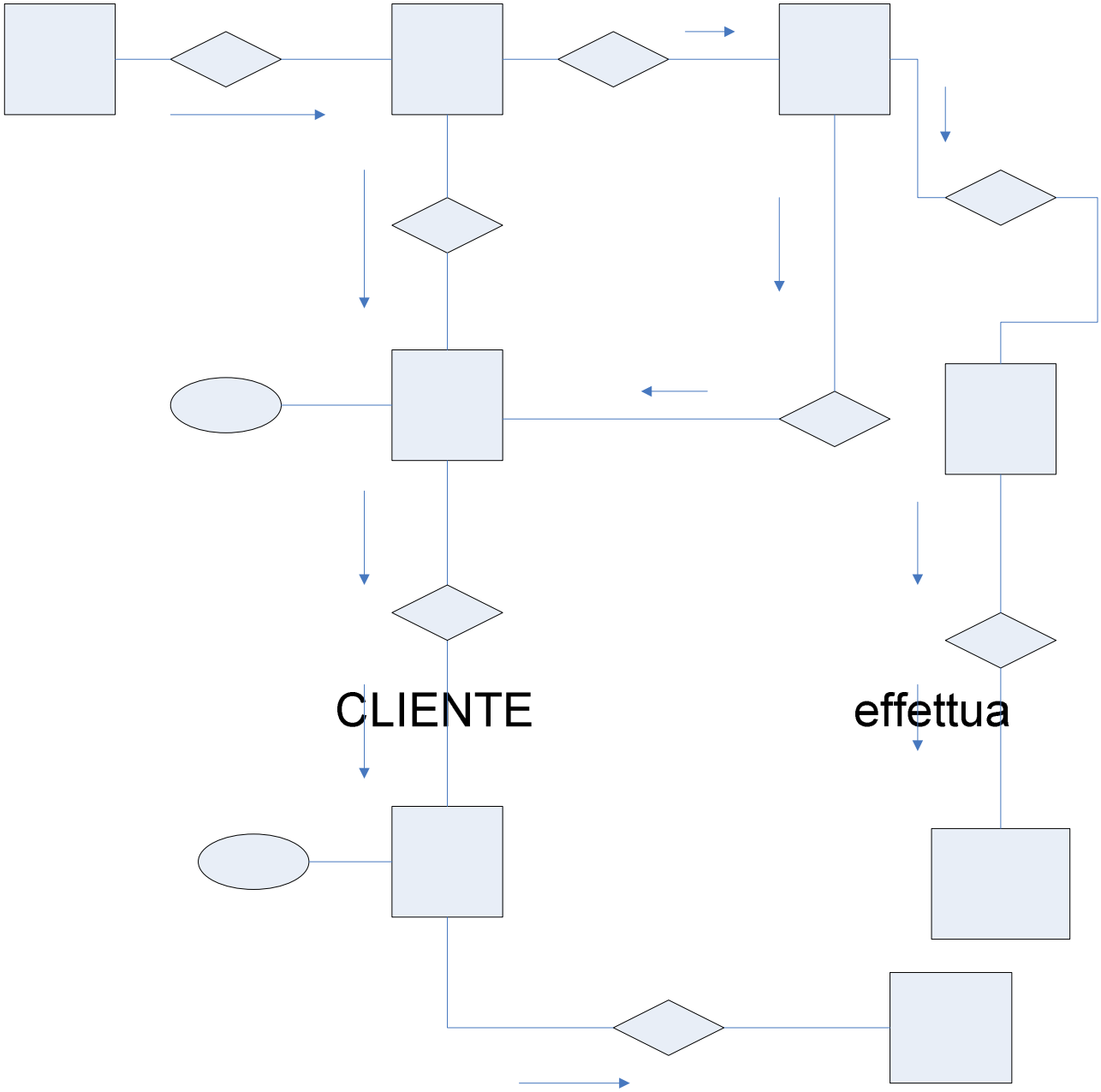


Tabella Modelli

Tabella Ordini



Relazioni

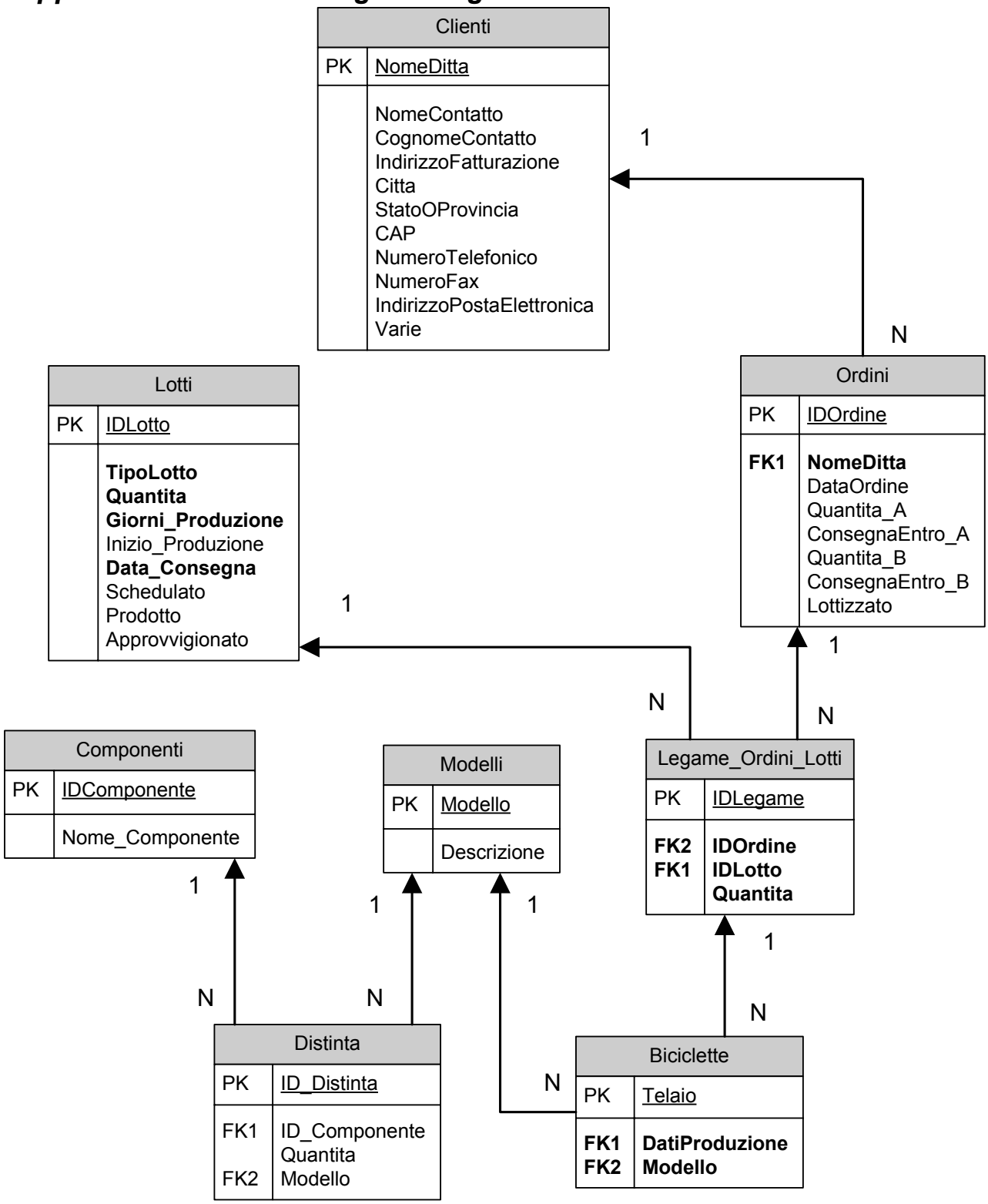


Progetto Logico

Descrizione del Progetto Logico

Dall'ERD è dunque possibile in ultima analisi evidenziare le tabelle di utilizzo del sistema e le relazioni esterne che intercorrono tra di esse.

Rappresentazione del Progetto Logico



Elenco Tabelle

Tabella BICICLETTE

	BICICLETTE	Archivio delle biciclette Prodotte	TIPO CAMPO
P	TELAIO	Numero identificativo della bicicletta prodotta	TESTO(50)
	DATIPRODUZIONE	Legame con i lotti di produzione	INTERO LUNGO
	MODELLO	Modello della bicicletta prodotta A o B	TESTO(50)

Tabella CLIENTI

	CLIENTI	Archivio dei Clienti	TIPO CAMPO
P	NOMEDITTA	Nome ditta	TESTO(50)
	NOMECONTATTO	Nome Cliente	TESTO(30)
	COGNOMECONTATTO	Cognome Cliente	TESTO(50)
	INDIRIZZOFATTURAZIONE	Indirizzo Fatturazione	TESTO(255)
	CITTA	Città	TESTO(50)
	STATOOPROVINCIA	Stato/Prov.	TESTO(20)
	CAP	CAP	TESTO(20)
	NUMEROTELEFONICO	Num. telefonico	TESTO(30)
	NUMEROFAX	Numero fax	TESTO(30)
	INDIRIZZOPOSTAELETTRONICA	Indirizzo e-mail	TESTO(50)
	VARIE	Note	TESTO(255)

Tabella COMPONENTI

	COMPONENTI	Archivio Componenti delle Biciclette	TIPO CAMPO
P	IDCOMPONENTE	Numero progressivo componente	CONTATORE
	NOME_COMPONENTE	Nome del componente	TESTO(50)

Tabella DISTINTA

	DISTINTA	Archivio della distinta di produzione	TIPO CAMPO
P	ID_DISTINTA	Numero progressivo della distinta	CONTATORE
	ID_COMPONENTE	Numero progressivo componente	NUMERICO
	QUANTITA	Quantità Componente	NUMERICO
	MODELLO	Modello della Bicicletta	TESTO(50)

Tabella LEGAME_ORDINI_LOTTI

	LEGAME_ORDINI_LOTTI	Entità di collegamento	TIPO CAMPO
P	IDLEGAME	Numero progressivo legame	CONTATORE
	IDORDINE	Numero progressivo Ordine	NUMERICO
	IDLOTTO	Codice identificavo del lotto	TESTO(50)
	QUANTITA	Quantità dell'ordine nel lotto	NUMERICO

Tabella LOTTI

	LOTTI	Archivio dei Lotti	TIPO CAMPO
P	IDLOTTO	Codice identificavo del lotto	TESTO(50)
	TIPOLOTTO	Modello della Bicicletta	TESTO(1)
	QUANTITA	Quantità delle biciclette che compongono il lotto	NUMERICO
	GIORNI_PRODUZIONE	Giorni minimi di produzione del lotto	NUMERICO
	INIZIO_PRODUZIONE	Giorno di inizio Produzione	DATETIME
	DATA_CONSEGNA	Data Massima di produzione lotto	DATETIME
	SCHEDULATO	Flag di schedulazione	SI/NO
	PRODOTTO	Flag di produzione	SI/NO
	APPROVVIGIONAMENTO	Flag di approvvigionamento	SI/NO

Tabella MODELLI

	MODELLI	Archivio dei modelli	TIPO CAMPO
P	MODELLO	Modello della Bicicletta	TESTO(50)
	DESCRIZIONE	Descrizione del modello	TESTO(50)

Tabella ORDINI

	ORDINI	Archivio degli ordini	TIPO CAMPO
P	IDORDINE	Numero progress	CONTATORE
	NOMEDITTA		TESTO(50)
	DATAORDINE		DATETIME
	QUANTITA_A		NUMERICO
	CONSEGNAENTRO_A		DATETIME
	QUANTITA_A		NUMERICO
	CONSEGNAETRO_B		DATETIME
	QUANTITA_B		NUMERICO
	LOTTIZZATO		SI/NO

Progetto fisico

Descrizione del progetto fisico

Lo schema sotto riportato indica lo schema fisico delle entità che costituiscono il Data Base del sistema BikesAgent.
Per ogni singola entità verrà dunque riportata la sintassi CREATE TABLE per la creazione della relativa tabella.

Rappresentazione del progetto fisico

```
#
# Struttura fisica dell'entità : 'BICICLETTE'
#
CREATE TABLE `BICICLETTE` ( `Telaio` CHAR(50), `DatiProduzione` INTEGER DEFAULT
'0', `Modello` CHAR(50));



#
# Struttura fisica dell'entità : 'CLIENTI'
#
CREATE TABLE `CLIENTI` ( `NomeDitta` CHAR(50), `NomeContatto` CHAR(30),
`CognomeContatto` CHAR(50), `IndirizzoFatturazione` CHAR(255), `Citta` CHAR(50),
`StatoOProvincia` CHAR(20), `CAP` CHAR(20), `NumeroTelefonico` CHAR(30), `NumeroFax`
CHAR(30), `IndirizzoPostaElettronica` CHAR(50), `Varie` CHAR(255));



#
# Struttura fisica dell'entità : 'COMPONENTI'
#
CREATE TABLE `COMPONENTI` ( `IDComponente` INTEGER AUTO_INCREMENT,
`Nome_Componente` CHAR(50), PRIMARY KEY (`IDComponente`));



#
# Struttura fisica dell'entità : 'DISTINTA'
#
CREATE TABLE `DISTINTA` ( `ID_Distinta` INTEGER AUTO_INCREMENT,
`ID_Componente` INTEGER DEFAULT '0', `Quantita` INTEGER DEFAULT '0', `Modello`
CHAR(50), PRIMARY KEY (`ID_Distinta`));
```

```
#
# Struttura fisica dell'entità : 'LEGAME_ORDINI_LOTTI'
#
CREATE TABLE `LEGAME_ORDINI_LOTTI` ( `IDLegame` INTEGER AUTO_INCREMENT,
`IDOrdine` INTEGER NOT NULL  DEFAULT '0', `IDLotto` CHAR(50) NOT NULL , `Quantita`
INTEGER NOT NULL  DEFAULT '0', PRIMARY KEY (`IDLegame`));



#
# Struttura fisica dell'entità : 'LOTTI'
#
CREATE TABLE `LOTTI` ( `IDLotto` CHAR(50), `TipoLotto` CHAR(1) NOT NULL , `Quantita`
INTEGER NOT NULL  DEFAULT '0', `Giorni_Produzione` INTEGER NOT NULL  DEFAULT
'0', `Inizio_Produzione` DATETIME, `Data_Consegna` DATETIME NOT NULL , `Schedulato`
BIT, `Prodotto` BIT, `Approvvigionato` BIT);



#
# Struttura fisica dell'entità : 'MODELLI'
#
CREATE TABLE `MODELLI` ( `Modello` CHAR(50), `Descrizione` CHAR(50));



#
# Struttura fisica dell'entità : 'ORDINI'
#
CREATE TABLE `ORDINI` ( `IDOrdine` INTEGER AUTO_INCREMENT, `NomeDitta`
CHAR(50), `DataOrdine` DATETIME, `Quantita_A` INTEGER DEFAULT '0',
`ConsegnaEntro_A` DATETIME, `Quantita_B` INTEGER DEFAULT '0', `ConsegnaEntro_B`
DATETIME, `Lottizzato` BIT, PRIMARY KEY (`IDOrdine`));
```

APPENDICE

Progettazione in SL e in RDF:

Introduzione

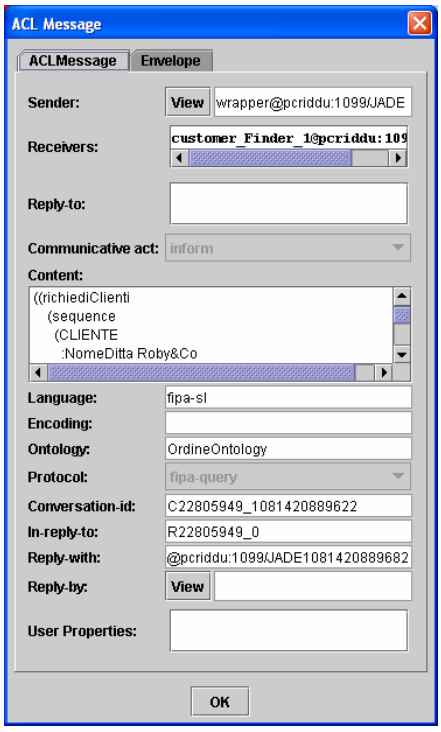
Nell'arco dello sviluppo del progetto BikeAgents e dello studio di tutte le tecnologie relative a questo, si è avuta occasione di poter esaminare due linguaggi molto usati per la comunicazione fra agenti: SL ed RDF. Si è avuta la possibilità di approfondire la progettazione di sistemi basati su questi linguaggi e la loro realizzazione pratica, cogliendo quindi i vantaggi e gli svantaggi nell'utilizzo dell'uno piuttosto che dell'altro linguaggio.
Scopo di questo documento è quello di evidenziare i problemi riscontrati, le soluzioni individuate, le nostre opinioni e i nostri suggerimenti per l'implementazione di sistemi ad agenti con questi linguaggi.

Sviluppo del progetto in SL

Prima che si possa progettare un sistema è necessario conoscere le tecnologie e gli strumenti che si andranno ad utilizzare. E' stato quindi necessario documentarsi sul funzionamento di JADE, sugli strumenti di controllo e di monitoraggio degli agenti e sulle API per la programmazione. Lo studio dell'architettura ad agenti realizzata con JADE è fortemente connessa con il linguaggio SL. Molti, se non tutti gli esempi, sono sviluppati in SL, così come i tutorial per lo studio e la realizzazione di ontologie.
Si è quindi iniziato, per semplicità, lo studio e la realizzazione di piccoli agenti di prova con linguaggio SL, acquisendo confidenza e utile praticità realizzativa nello sviluppo di comunicazioni in SL.
D'altra parte l'SL non pone vincoli riguardo la corretta implementazione dei protocolli di comunicazione FIPA, i quali astraggono dal linguaggio utilizzato.
Queste idee ci hanno permesso di sviluppare un primo progetto mediante SL, per poi svilupparne una versione in RDF.

Risultato ottenuto

L'utilizzo del linguaggio SL non ha dato alcuna limitazione realizzativa e si è riusciti a progettare e implementare l'intero progetto. Infatti non si sono trovate sostanziali limitazioni nell'utilizzo di PTK come strumento progettuale.
JADE e tutti gli strumenti di controllo sembrano essere costruiti per essere utilizzati con SL.
Ad esempio lo sniffer di JADE ha una finestra molto piccola che è però sufficiente a visualizzare messaggi complessi in SL. Il ritorno a capo alla fine di ogni token ne permette una buona lettura e una immediata individuazione delle parole chiave.
Si riporta a destra un esempio di visualizzazione di un messaggio in SL tramite sniffer: nella finestra content è possibile individuare il nome del predicato (*richiediClienti*), capire che segue una lista (*sequence*) e cosa contiene la lista (*CLIENTE..)* e i suoi attributi (*:NomeDitta Roby&Co*).



Riportiamo di seguito l'intero messaggio per come viene generato:

```
((richiediClienti
     (sequence
       (CLIENTE
         :NomeDitta Roby&Co
         :NomeContatto Roberto
         :CognomeContatto Caico
         :IndirizzoFatturazione "Via F.F.Orsi"
         :Citta Palermo
         :StatoOProvincia PA
         :CAP "90123"
         :NumeroTelefonico "091166144"
         :NumeroFax "091005800"
         :IndirizzoPostaElettronica robertocaico@tin.it
         :Varie "")
       (CLIENTE
         :NomeDitta FrancioTech
         :NomeContatto Francesco
         :CognomeContatto Termine
         :IndirizzoFatturazione "Via Cangiamila 89"
         :Citta " Palermo"
         :StatoOProvincia " PA"
         :CAP "90136"
         :NumeroTelefonico " 091123456"
         :NumeroFax " 09133322211"
         :IndirizzoPostaElettronica franciofra@libero.it
         :Varie ""))))
```

La fase di debug e di testing è stata notevolmente avvantaggiata e accelerata dall'utilizzo del linguaggio SL insieme agli strumenti di JADE.

Passaggio al linguaggio RDF

Idealmente il passaggio da un linguaggio ad un altro, per la comunicazione tra agenti, è un'operazione semplice e trasparente. La regola vuole che si utilizzi il codec adatto, si importi e si istanzi la classe relativa.
In pratica questa operazione è tutt'altro che indolore.
La causa principale è da imputare alla bontà dei codec, che non supportano alcune funzionalità riguardanti l'ontologia, inoltre non è facile trovare una documentazione adeguata e completa con esempi funzionati ed esaustivi.
Per ovviare a questi problemi è necessario, quindi, un riadattamento di tutte le classi dell'ontologia e fare numerosi tentativi per capire la causa degli errori riscontrati.

Il primo codec da noi utilizzato è quello documentato nei tutorial di JADE.
Questo consta di tre file .jar che devono trovarsi nella directory *jade\add-ons\RDFCodec\lib*:

- rdf.jar
- rdf-api-2001-01-19.jar
- xerces.jar

quest'ultima libreria inoltre è molto datata e la sua compilazione ha richiesto alcuni passaggi particolari.

L'utilizzo di questo codec ha presentato i seguenti problemi:

- Non è supportato DATE come tipo predefinito. E' necessario quindi introdurre un concetto data nell'ontologia e convertire manualmente le date durante le operazioni di codifica e decodifica
- Non è supportato LONG come tipo predefinito
- Non sono supportati i campi opzionali, causando la codifica di campi nulli e generando un errore in fase di decodifica.

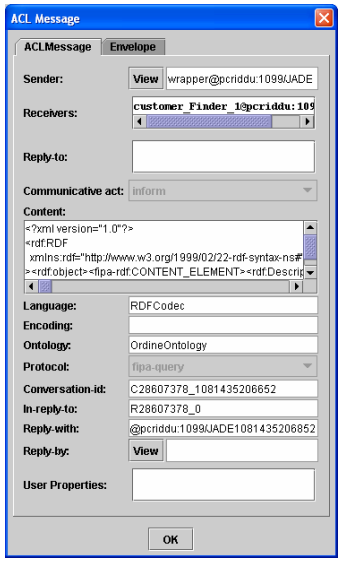
Il secondo codec utilizzato è composto da un solo file nella directory:
jade\add-ons\RDFCodec2\lib\RDFCodec.jar.
Questo non ha i problemi riscontrati con il primo, ma ha difficoltà nel distinguere token uguali applicati a diversi contesti.
Quindi ad esempio la parola "cliente", nome di un concetto, non potrà più essere utilizzata in un'altra azione o predicato per individuare una lista di clienti, ovvero la parola cliente non potrà essere utilizzata in nessun altra parte dell'ontologia.
La nostra scelta è ricaduta su questi ultimi codec che, fatte le opportune modifiche alle classi di ontologia, funzionano correttamente.
E' però necessario fare attenzione all'utilizzo di caratteri speciali che sono riservati all'xml, questi generano infatti errori in fase di decodifica che sono difficili da individuare perché legati al dato stesso e non alla struttura dell'ontologia o al funzionamento dei codec.
Ad esempio non è possibile inserire il nome "Roby&Co" dato che il carattere "&" è riservato per l'xml. E' stato necessario sostituire il nome con "RobyCo".
Prese in considerazione queste differenze e svolte le opportune modifiche si è potuto realizzare il progetto in RDF.

Risultato ottenuto

Allo stato attuale è possibile cambiare il linguaggio di comunicazione modificando per ogni agente solamente la riga di codice che specifica il codec:

- `Codec codec = new SLCodec();`
- `Codec codec = new RDFCodec();`

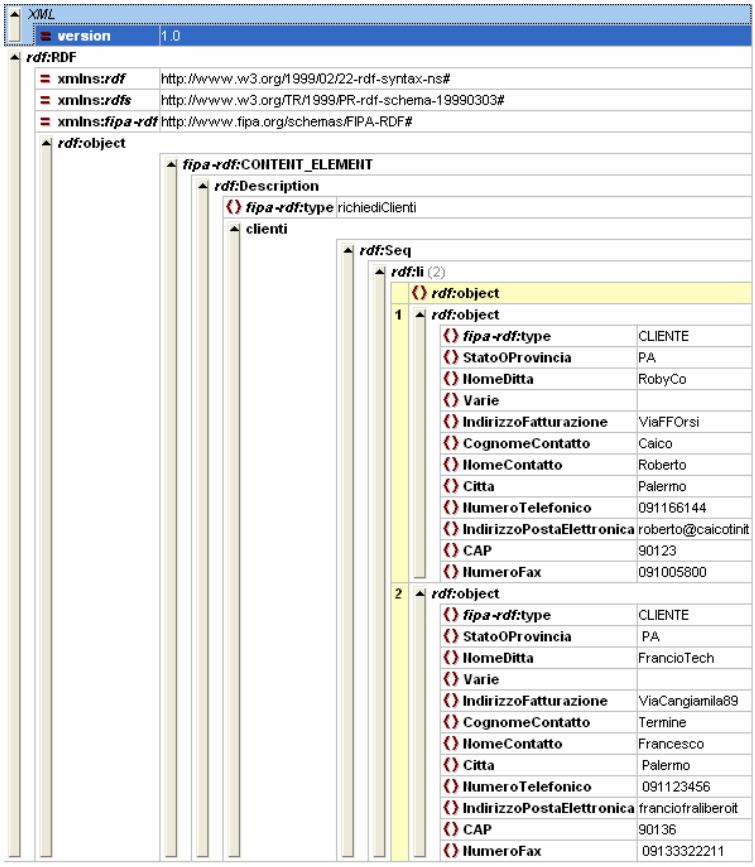
La codifica in RDF del messaggio presenta però una notevole difficoltà di lettura all'interno della struttura JADE, così com'è possibile vedere in figura. Inoltre la struttura è priva di un layout facilmente comprensibile, di seguito viene riportato il testo così come viene codificato:




```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"  xmlns:fipa-rdf="http://www.fipa.org/schemas/FIPA-RDF#"><rdf:object><fipa-rdf:CONTENT_ELEMENT><rdf:Description><fipa-rdf:type>richiediClienti</fipa-rdf:type><clienti><rdf:Seq><rdf:li><rdf:object><fipa-rdf:type>CLIENTE</fipa-rdf:type><StatoOProvincia>PA</StatoOProvincia><NomeDitta>Roby&Co</NomeDitta><Varie></Varie><IndirizzoFatturazione>Via F.F.Orsi</IndirizzoFatturazione><CognomeContatto>Caico</CognomeContatto><NomeCon
```

```
tatto>Roberto</NomeContatto><Citta>Palermo</Citta><NumeroTelefonico>091166144</N
umeroTelefonico><IndirizzoPostaElettronica>robertocaico@tin.it</IndirizzoPostaEl
ettronica><CAP>90123</CAP><NumeroFax>091005800</NumeroFax></rdf:object></rdf:li>
<rdf:li><rdf:object><fipa-rdf:type>CLIENTE</fipa-rdf:type><StatoOProvincia>
PA</StatoOProvincia><NomeDitta>FrancioTech</NomeDitta><Varie></Varie><IndirizzoF
atturazione>Via Cangiamila
89</IndirizzoFatturazione><CognomeContatto>Termine</CognomeContatto><NomeContatt
o>Francesco</NomeContatto><Citta> Palermo</Citta><NumeroTelefonico>
091123456</NumeroTelefonico><IndirizzoPostaElettronica>franciofra@libero.it</Ind
irizzoPostaElettronica><CAP>90136</CAP><NumeroFax>
09133322211</NumeroFax></rdf:object></rdf:li></rdf:Seq></clienti></rdf:Descripti
on></fipa-rdf:CONTENT_ELEMENT></rdf:object></rdf:RDF>
```


D'altro canto è possibile con alcuni passaggi riuscire a importare il contenuto del messaggio all'interno di uno dei numerosi tool di visualizzazione di xml, come ad esempio XMLSpy, e ottenere una rappresentazione del messaggio molto intuitiva.



L'RDF consente quindi di poter utilizzare il messaggio anche in altri contesti sfruttando la portabilità dell'xml in ogni ambito applicativo.
Per il nostro progetto si ritiene però sufficiente l'utilizzo di SL, non dovendo sfruttare tutte le possibili funzionalità dell'RDF.

SL e PTK

PTK è stato progettato per essere utilizzato contestualmente a RDF. Ma le differenze di un progetto in RDF rispetto a uno in SL sono minime.
Nel Communication Ontology Diagram è sufficiente sostituire il nome del codec da utilizzare in ogni comunicazione. Non vi sono altre modifiche, né evidenti conseguenze di una modifica di questo tipo..
L'utilizzo di PTK non tiene conto del codec utilizzato e la generazione dell'ontologia prescinde da questo. Inoltre le restrizioni legate alle ridotte funzionalità dei codec RDF rispetto a quello SL comporta maggior lavoro in fase di progettazione dell'ontologia e un maggior impegno per la correzione dei bug della generazione automatica del codice.