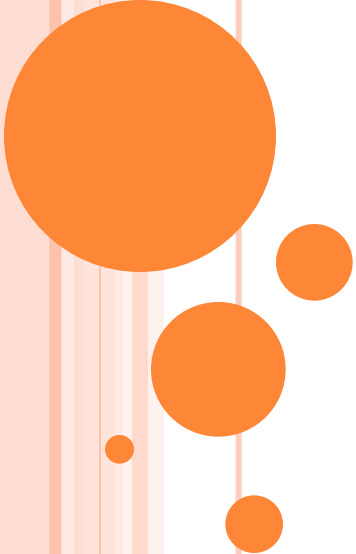




LA METODOLOGIA ADELFE

Sistemi Multiagente LS

Paola Castrignano

COS'È UNA METODOLOGIA

- Viene spesso confusa col processo di sviluppo del software (casi d'uso, uml..), ma non è solo questo.
- Processo di sviluppo del software: insieme di procedure, politiche, strutture, tecnologie per concepire, sviluppare, fare il deployment e mantenere il software.
- Ha lo scopo di dare un approccio coerente per risolvere problemi in un contesto di sviluppo software.
- Software (development) Process Model: è il template col quale dobbiamo sviluppare il nostro lavoro.

METODOLOGIA

Si sviluppa in uno scenario dove c'è un gap tra modelli e tecnologie in quanto progettiamo ad agenti ma implementiamo ad oggetti.

Linguaggi veri e propri ancora non esistono.

Ha due componenti importanti:

- le *astrazioni* (i mattoncini fondamentali con cui andiamo a costruire il sistema).
- il *processo* (tutti i passi che devono essere fatti).

QUALI SONO LE ASTRAZIONI CHIAVE?

Dobbiamo capire come modellare i dati che dobbiamo utilizzare. L'astrazione che utilizziamo denota il livello cognitivo a cui ci riferiamo.

Sono necessarie nuove astrazioni che gli oggetti non rappresentano:

- *Agenti*: entità autonome con un proprio flusso, situate e interagenti tra loro.
- *Ambiente*: le risorse che l'agente percepisce e che ha a disposizione.
- *Protocolli di interazione*: modi per poter contattare gli agenti. Non ci sono metodi.

AOSE

Le metodologie AOSE sono utilizzate per analizzare, progettare, sviluppare sistemi multi-agente.

Partono da un meta-modello che relaziona le astrazioni che saranno utilizzate per modellare il software e permette di verificare e testare la completezza di una metodologia.

Basta istanziare il meta-modello per soddisfare i requisiti dello specifico problema.

Il meta-modello dei MAS include concetti come: ruoli, goal, task, piani e comunicazioni.

MULTI-AGENT SYSTEM (MAS)

Sistema composto di agenti cooperativi e competitivi che interagiscono per raggiungere degli obiettivi comuni o individuali.

Si può dire essere autonomo, collocato, pro-attivo ed organizzato socialmente così come gli agenti.

Queste caratteristiche degli agenti portano ad un insieme di caratteristiche di alto livello quali:

- adattabilità, flessibilità, scalabilità, manutenibilità, il probabile manifestarsi di un comportamento personale.

MAS ADATTIVO

E' un MAS in cui gli agenti sono auto-organizzanti , sistemi aperti, fonti di eventi inaspettati.

In questi sistemi è l'ambiente a perturbare il sistema, a modificare gli equilibri.

Questo implica che venga modellato.

- Il “ forte adattamento ” di un sistema multi-agente è la capacità del sistema di tener conto a eventi imprevedibili al fine di reagire ad ambienti in evoluzione per realizzare il giusto task.

Per i MAS con una forte capacità di adattamento, è stata sviluppata una teoria chiamata *AMAS*.

LA TEORIA AMAS

- Obiettivo: realizzare MAS aventi le “classiche” caratteristiche per la costruzione di società di agenti situati.

Come un MAS è immerso nell’ambiente e deve raggiungere un comportamento adeguato (da riprodurre il comportamento di una società simulata) o di uno funzionale (eseguendo il giusto task, quello per il quale il sistema è stato costruito).

Deve portare ad un’attività collettiva coerente che realizza il giusto task.

LA TEORIA AMAS (2)

- La specificità risiede nel fatto che non c'è il codice della funzione globale del sistema ad agenti.

La funzione globale emerge dal comportamento collettivo dei diversi agenti che compongono il sistema.

Ogni agente possiede la capacità di *auto-organizzazione* (capacità di riorganizzare localmente le interazioni con gli altri a seconda del task individuale che deve risolvere).

Questa capacità permette al sistema di adattarsi e di realizzare una funzione che non è codificata nell'agente. La funzione globale è quindi *emergente*.

DEFINIZIONE DI AUTO-ORGANIZZAZIONE

Un sistema è *auto-organizzante* se il cambiamento alla propria organizzazione avviene spontaneamente piuttosto che essere originato da un'entità esterna.

Gli elementi fondamentali dei sistemi auto-organizzanti:

- riduzione dell'entropia ovvero sia l'aumento dell'ordine;
- autonomia: non c'è delegazione del controllo per quanto riguarda la creazione e l'aumento dell'organizzazione interna;
- adattivo: si auto-organizzano in base ai cambiamenti dell'ambiente;
- dinamico: è un processo non uno stato finale.

DEFINIZIONE DI EMERGENZA

La base dei sistemi auto-organizzanti è il concetto di *emergenza*.

L'*emergenza* si riferisce al sorgere di strutture, pattern, proprietà che siano nuovi e coerenti che emergono durante il processo di auto-organizzazione nei sistemi complessi. Implicito è il concetto di macro-livello e micro-livello.

Il micro-livello è quello delle interazioni locali che sappiamo essere alla base del fenomeno.

Il macro-livello è quello in cui avviene il fenomeno globale che è prodotto dalle micro-interazioni anche se non sappiamo come questi 2 livelli si collegano.

I 2 livelli non hanno un modello di riferimento.

REQUISITI PER L'EMERGENZA

L'emergenza può essere esposta nei sistemi con i seguenti requisiti:

- *Non-linearità*: le interazioni potrebbero essere non lineari e sono rappresentate come feedback loops.
- *Auto-organizzazione*: la capacità di auto-regolare e adattare il comportamento.
- *Oltre l'equilibrio*: non è interessato ad uno stato finale ma all'evolvere della dinamica del sistema.
- *Attrattori*: stati del sistema che sono dinamicamente stabili.

Fenomeni → Emergenti

Auto-organizzanti → Sistemi

LA TEORIA AMAS

E' basata sulla capacità che un agente possiede di essere localmente “*cooperativo*” in grado di riconoscere fallimenti di cooperazione chiamati “*Non Cooperative Situation*” (*NCS*) e di trattarle.

Un agente può rilevare 3 tipi di *NCS*:

- quando un segnale percepito dal suo ambiente non è inteso e non letto senza ambiguità;
- quando l'informazione percepita non induce l'agente ad un'attività di processo;
- quando i risultati conclusivi portano ad agire in modo inutile per l'ambiente.

LA TEORIA AMAS

Gli *agenti cooperativi* per costruire AMAS sono composti di 5 parti che contribuiscono al loro comportamento:

- *skills* (competenze), ciò che l'agente è in grado di fare;
- *rappresentazioni del mondo*, la conoscenza che ha di sé, degli altri e del suo ambiente;
- *linguaggio di interazione*, per comunicare con gli altri o con il suo ambiente;
- *attitudini*, le capacità che possiede a fronte della sua conoscenza;
- un atteggiamento sociale guidato da ciò che chiamiamo *cooperazione*.

PARADIGMA AD AGENTI: SCELTA MIGLIORE PER MODELLARE SOS?

Le caratteristiche dei sistemi auto-organizzanti ed emergenti che è possibile trovare e mappare sui sistemi computazionali:

- autonomia e incapsulamento del comportamento: i vari agenti sono autonomi perché non hanno un'idea del comportamento globale, quindi è inutile controllarli.
- azioni locali e percezioni: sono locali, quindi l'interazione è locale.
- distribuzione: c'è bisogno di un ambiente distribuito che supporti le interazioni.
- supporto per l'organizzazione e concetti di cooperazione

Metodologia ADELFE

PERCHÉ ADELFE?

Nel dominio multiagente esistono metodologie ma poche sono in grado di trattare concetti come l'ambiente in evoluzione, dinamico.

Il software adattivo è utilizzato in situazioni in cui sia l'ambiente è imprevedibile che il sistema è aperto.

E' il motivo principale per cui è richiesta una metodologia per la progettazione di AMAS chiamata ADELFE.

OBIETTIVO DI ADELFE

- Coprire tutte le fasi della progettazione classica del software dalla fase dei requisiti fino al deployment.

Nata per i sistemi auto-organizzanti.

Dedicata al design di sistemi complessi aperti e non ben specificati (come i sistemi adattivi), perché in essi a priori non sappiamo quanti agenti ci sono e che ruoli copriranno.

ADELFE

ADELFE è un acronimo francese che significa

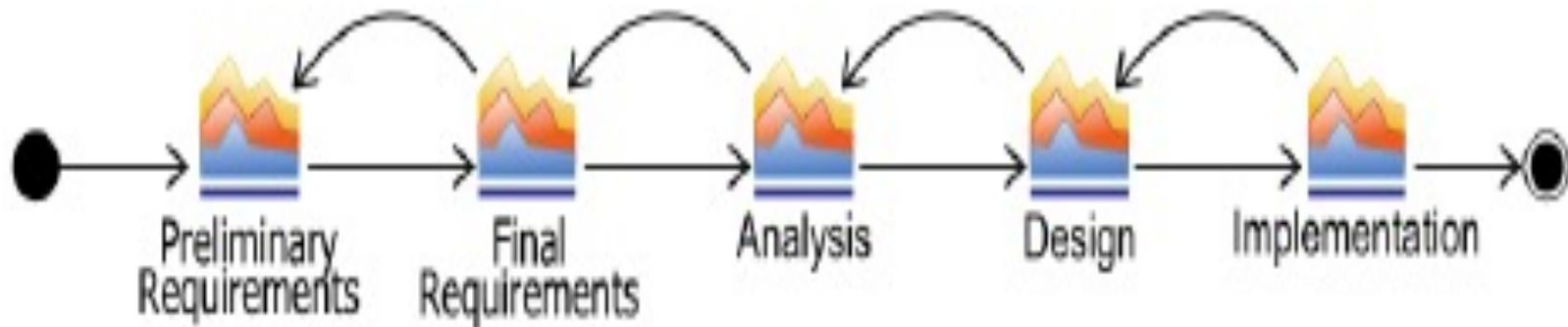
- “*Toolkit for Designing Software with Emergent Functionalities*”.

Basata sul RUP (Rational Unified Process), utilizza notazioni UML e l'estensione di UML già fatto in AUML, in particolare le notazioni AIP (Agent Interaction Protocols) e aggiunge alcuni step specifici per il design di sistemi adattivi.

Segue il vocabolario di SPEM: le fasi sono work-definition, le attività step e così via.

E' fondata sulla vista di un sistema multi-agente come un'organizzazione dinamica di vari agenti cooperativi.

FASI DI ADELFE



FLUSSI DI LAVORO

- I *requisiti del flusso di lavoro*: elemento fondamentale nei passi dell'ingegneria del software.

Nella teoria AMAS il processo di adattamento inizia dalle interazioni tra il sistema e l'ambiente. Qui è importante dare un modello dell'ambiente.

- Nell'*analisi del flusso di lavoro*, sono aggiunti due passi per il RUP: l'identificazione degli agenti e l'adeguatezza della teoria AMAS.
- Nel *design del flusso di lavoro*, il modello agente e il modello NCS sono aggiunti al RUP.

Dispone di un tool che fa delle domande che portano a capire se è veramente necessario usare la teoria degli agenti cooperativi.

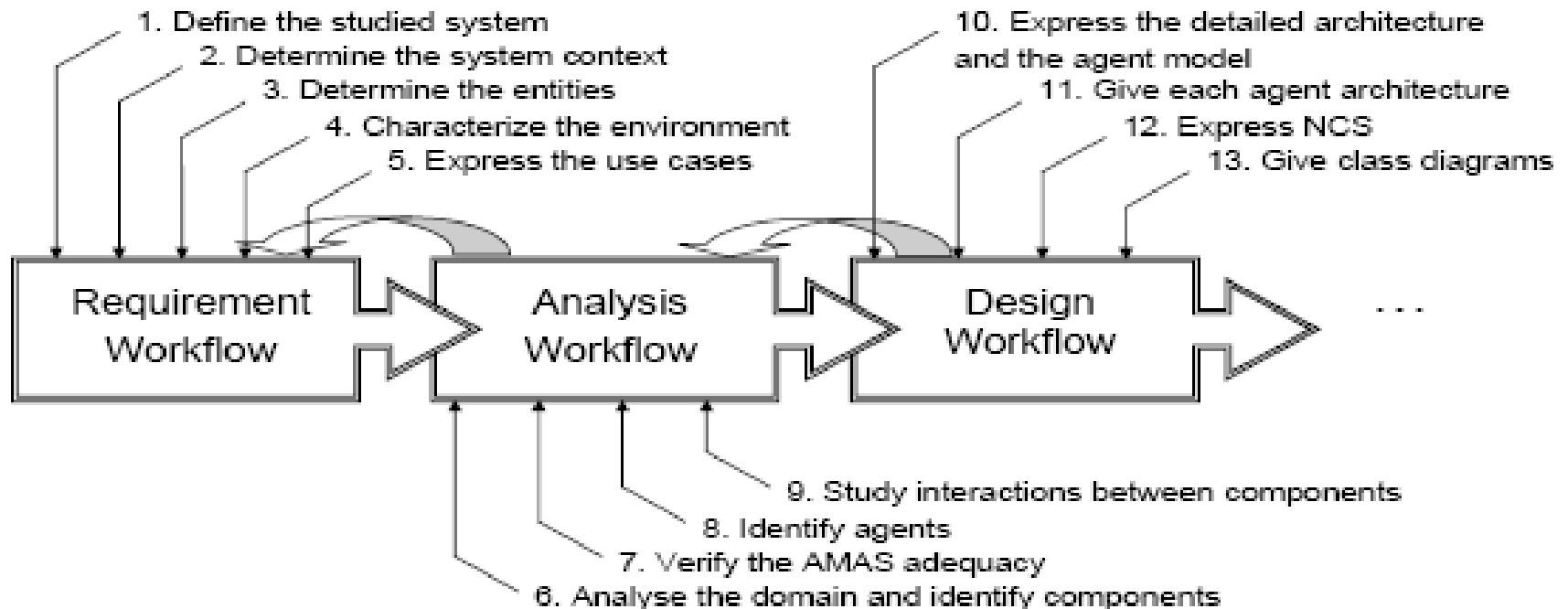
FLUSSI DI LAVORO (2)

Un MAS che non è in interazione cooperativa con l'ambiente deve adattarsi ad esso. Secondo la teoria AMAS la funzione globale del sistema non è codificata, solo il comportamento locale degli agenti che compongono il sistema.

L'adattamento sarà gestito da questi agenti attraverso il modello NCS.

ADELFE fornisce il designer di generici *cooperative failures*; ma anche tabelle con campi per compilare il relativo nome dell'NCS, il suo tipo generico, lo stato in cui l'agente deve essere individuato, le condizioni della sua individuazione e quali azioni l'agente deve eseguire per trattarlo.

PANORAMICA DEI 3 PRINCIPALI FLUSSI DI LAVORO



STEP-BY-STEP DETTAGLI DELLA METODOLOGIA

- Il primo step è dedicato alla definizione dei requisiti usuali dello sviluppo software di una metodologia.

E' chiamata *fase dei Requisiti Preliminari*.

ADELFE non aggiunge nulla al lavoro di definizione dei requisiti preliminari come descritti dal RUP.

FASE DEI REQUISITI PRELIMINARI

Costituisce una sintesi dei requisiti (esigenze degli utenti, parole chiavi e limiti).

- Obiettivo: descrizione del dominio del problema così come richiesto essere risolto e come specificato dalle esigenze degli utenti finali.

Definire come deve essere il sistema, per trasformare questa visione in un modello di caso d'uso, e per organizzare e gestire i requisiti (funzionali o non) e le loro priorità.

Il designer deve definire la funzione del sistema studiato e il modello del suo ambiente.

DEFINIZIONE DEL SISTEMA STUDIATO

E' il risultato dell'analisi dei requisiti dell'insieme degli artefatti.

L'output di questa attività è un insieme di parole chiavi che definiscono il sistema.

FASE DEI REQUISITI FINALI

Viene inclusa la descrizione dell'ambiente e l'identificazione dei diversi elementi che lo popolano.

Attività aggiunta al RUP in quanto l'ambiente è un concetto molto importante nella teoria AMAS, deve essere studiato e compreso prima della definizione dei casi d'uso.

- Scopo: determinare le entità, definire il contesto e caratterizzare l'ambiente.

La caratterizzazione inizia dall'identificazione delle entità che interagiscono con il sistema e le restrizioni di queste interazioni.

DEFINIZIONE E MODELLAZIONE DELL'AMBIENTE

Consiste nella definizione dettagliata dell'ambiente costituita da tre fasi:

- la determinazione delle entità;
- la definizione del contesto;
- la definizione delle caratteristiche dell'ambiente.

Si concentra su quello che può essere in interazione con il sistema studiato, quindi il contesto mediante una caratterizzazione di flussi di dati e interazioni tra *entità passive*, *attive* e il sistema.

DETERMINAZIONE DELLE ENTITÀ

L'*entità* è un attore così come in UML, ma può essere classificato come passivo o attivo.

- Un' *entità attiva* può agire in maniera autonoma e dinamicamente con il sistema.
- Un' *entità passiva* è considerata come una risorsa del sistema che può essere usata o modificata dalle entità attive.

La classificazione delle entità è essenziale in AMAS in quanto gli agenti saranno parte del sistema trattati come entità attive.

DEFINIZIONE DEL CONTESTO

La definizione del contesto analizza l'ambiente attraverso l'interazione tra entità e il sistema definendo diagrammi di collaborazione (entità/sistema) e diagrammi di sequenza (entità/entità) UML.

Definisce questi diagrammi sulla base del risultato dell'attività precedente in cui le entità sono state pre-definite con il supporto di una serie di parole chiavi fornite .

DEFINIZIONE DELLE CARATTERISTICHE DELL'AMBIENTE

Il designer deve caratterizzare l' ambiente sviluppato in termini di:

- *accessibilità*: l'ambiente può ottenere informazioni sullo stato dell'ambiente;
- *continuo*: il numero delle interazioni tra il sistema e le entità è infinito.
- *deterministico*: il sistema non è in grado di conoscere gli effetti delle sue azioni sulle entità attive;
- *dinamico*: l'evoluzione delle entità attive non dipende dal sistema, dal punto di vista del sistema è imprevedibile.

DETERMINAZIONE DEI CASI D'USO

L'obiettivo di questa attività, che termina la fase dei requisiti, è di chiarire le diverse funzionalità del sistema a cui deve rispondere.

Il designer deve pensare a tutti gli eventi imprevisti e dannosi che potrebbero accadere causando situazioni non cooperative per gli agenti.

Queste cooperation failures sono eccezioni.

La determinazione dei casi d'uso viene modificata con l'aggiunta di uno step, in cui le cooperation failures devono essere identificate utilizzando specifiche notazioni.

ELABORAZIONE PROTOTIPI UI

L'elaborazione di prototipi di interfaccia utente (UI) modella specifiche dell' interfaccia utente grafica (GUI) usate nelle interazioni.

Le GUI vengono valutate come funzionali o non funzionali (ergonomia, design, ecc).

A volte in questa fase è necessario tornare all'attività precedente per migliorare l'interfaccia utente UI.

SINTESI FASE DEI REQUISITI PRELIMINARI E FINALI

WD₁: Preliminary Requirements

- A₁: Define user requirements
- A₂: Validate user requirements
- A₃: Define consensual requirements
- A₄: Establish keywords-set
- A₅: Extract limits constraints

WD₂: Final Requirements

A6: Characterize environment

- S1: Determine entities
- S2: Define context
- S3: Characterize environment

A7: Determine use cases

- S1: Draw inventory of use cases
- S2: Identify cooperation failures
- S3: Elaborate sequence diagrams

A8: Elaborate UI (user interface) prototypes

A9: Validate UI prototypes

FASE DI ANALISI

In questa fase deve avvenire l'identificazione degli agenti.

Si sviluppa una comprensione del sistema, la sua struttura in termini di componenti, è inclusa l'attività di verifica di adeguatezza AMAS per identificare agenti e l'interazione tra le entità.

ANALISI DEL DOMINIO E STUDIO DELL'ARCHITETTURA

L'analisi del dominio è una visione statica e un'astrazione del mondo reale e le entità collegate.

Considerando separatamente ogni caso d'uso da definizioni di scenari, il designer deve dividere il sistema in entità.

Il risultato dell'attività è un insieme di entità organizzati in diagrammi di classe preliminari.

VERIFICA DELL'ADEGUATEZZA DELLA TEORIA AMAS

Ha lo scopo di aiutare il designer a capire se la teoria AMAS è sufficiente a risolvere il suo problema perché, per alcune applicazioni, questo tipo di programmazione può essere inutile.

La tecnologia adatta AMAS di verifica delle attività del sistema di ADELFE, viene eseguita utilizzando uno strumento che studia l'adeguatezza del sistema considerando due livelli: *globale* e *locale (o componenti)*.

LIVELLI LOCALE E GLOBALE

- A *livello globale* lo studio deve rispondere alla domanda: “ E’ necessaria al sistema un’implementazione di una tecnologia AMAS? ”
- A *livello locale* la domanda è: “ Qualche componente necessita di essere implementato come AMAS? ” Per esempio è utile una certa decomposizione o ricorsione durante il design?

Se il tool risponde in modo affermativo alla prima domanda lo sviluppatore può continuare ad applicare il processo.

Se anche la risposta alla seconda domanda è affermativa la metodologia ADELFE dovrebbe essere applicata ai componenti come AMAS dall’evoluzione di cui hanno bisogno.

IDENTIFICAZIONE ENTITÀ AGENTE

L'identificazione degli agenti è stata anche introdotta per il RUP e deve analizzare le entità che saranno considerate un agente del sistema.

Gli agenti non sono noti in precedenza quindi il designer deve identificarli.

Entità che dimostrano proprietà come autonomia, obiettivo locale da perseguire, interazione con altre entità, visione parziale dell'ambiente e capacità di negoziare sono quelli da considerare come *agenti potenziali*.

Per trasformarsi effettivamente in un agente cooperativo, il potenziale agente deve essere propenso a fallimenti di cooperazione.

IDENTIFICAZIONE AGENTE (2)

Dallo studio delle sue interazioni con l' ambiente e con altre entità, il designer deve determinare se questa entità può incontrare tali situazioni che saranno considerate come Situazioni Non Cooperative a livello di agente.

Le entità incontrano tutti questi criteri che saranno identificati come agenti e le classi ad essi connessi contrassegnate come agenti.

L'attenzione di sviluppo del sistema AMAS è in agenti cooperativi che provengono dalle precedenti entità definite e le classi elaborate.

IDENTIFICAZIONE DI AGENTE (3)

Gli agenti cooperativi sono entità che soddisfano almeno i requisiti di autonomia, obiettivo locale e interazione con altre entità.

Dopo l'accertamento di tutti i possibili agenti, le classi sono contrassegnate con lo stereotipo agente cooperativo.

ADEGUATEZZA TEORIA AMAS A LIVELLO LOCALE

Se il primo step di adeguatezza alla teoria AMAS indica una possibile decomposizione, ogni agente deve essere analizzato come un sistema.

STUDIO DELLE INTERAZIONI TRA LE DIVERSE ENTITÀ

Attività che studia le interazioni tra entità attive/passive, tra entità attive e tra agenti.

L'interazione è rappresentata da diagrammi sequenza e diagrammi collaborazione.

Dato che il RUP è un caso d'uso guidato, per ciascun caso d'uso, deve essere definito un diagramma di sequenza in modo da mostrare la vista interna e le interazioni all'interno del sistema.

Le interazioni degli agenti sono descritte dal diagramma protocollo AUML.

SINTESI FASE DI ANALISI

A10: Analyze the Domain

- S1: Identify classes
- S2: Study interclass relationships
- S3: Construct preliminary class diagrams

A11: Verify the AMAS adequacy

- S1: Verify it at the global level
- S2: verify it at the local level.

A12: Identify Agents

- S1: Study entities in the domain context
- S2: Identify potentially cooperative agents
- S3: Determine agents

A13: Study Interactions between Entities

- S1: Study active/passive entities relationships
- S2: Study active entities relationships
- S3: Study agents relationships

FASE DI DESIGN

La prima attività del processo di design individua l'architettura dettagliata del sistema, la creazione di pacchetti sub-sistemi, oggetti, agenti e rapporti tra loro, al fine di migliorare i diagrammi classe con i nuovi elementi dopo che i nuovi agenti sono stati accolti.

ARCHITETTURA DETTAGLIATA E MODELLO MULTI-AGENTE

- Obiettivo: richiede di identificare i componenti software e descriverli.

Il risultato fornisce l'architettura del sistema in termini di blocchi necessari, classi, agenti e interazioni. E' incluso il modello agente, che rappresenta i rapporti tra gli agenti.

La precedente definizione di architettura può essere raffinata da determinare se alcuni design patterns e/o componenti riutilizzabili possono essere utilizzati.

Viene proposto uno specifico design pattern denominato *architettura agente cooperativo*.

ARCHITETTURA AGENTE COOPERATIVO

Questa attività aiuta il designer a compilare in una generica architettura data per un agente usata nella teoria AMAS.

Nell'architettura un agente è composto di un modello di competenze, modelli di rappresentazione, un linguaggio di interazione, attitudini e un'attitudine sociale chiamata cooperazione.

A seguito di una metodologia object-oriented, competenze, attitudini o rappresentazioni possedute da un agente potrebbero essere rappresentate da metodi.

ARCHITETTURA AGENTE COOPERATIVO (2)

Caratteristica degli agenti in AMAS: essere in grado di apprendere nuove competenze, nuove rappresentazioni o nuove attitudini ad adattarsi a un ambiente in evoluzione.

I metodi non possono cambiare, se sono caratteristiche dinamiche, essi devono adottare nuove rappresentazioni: essi devono essere AMAS stessi.

In tal caso, ADELFE raccomanda il designer di applicare nuovamente la metodologia di realizzare un sistema multi-agente adattivo per implementare le competenze, le rappresentazioni o le attitudini di un agente.

STUDIO DEI LINGUAGGI DI INTERAZIONE

Lo scambio di informazioni tra agenti deve essere descritto.

I linguaggi di interazione devono definire i protocolli usati dagli agenti per comunicare tra di loro.

Può essere implementato da una serie di classi o da un design pattern, che è chiuso a specifici strumenti di comunicazione tra agenti come l'implementazione ACL.

L'atteggiamento sociale di un agente è guidato dalla cooperazione, deve essere in grado di risolvere Situazioni Non Cooperative e un modello appositamente definito nell'attività successiva.

STUDIO DEI LINGUAGGI DI INTERAZIONE (2)

Per ogni scenario si descrivono questi scambi utilizzando diagrammi protocollo AUMML.

I diagrammi protocollo sono allegati al pacchetto (non classi) perché sono generici.

Il linguaggio definizione non è necessario quando le comunicazioni degli agenti avvengono attraverso l'ambiente.

DESIGN AGENTS

L'attività Design Agents è una specifica attività, consente al designer di raffinare le classi stereotipate CooperativeAgent individuate nelle attività precedenti.

I diversi moduli di un agente devono essere definiti dalla descrizione delle sue competenze, attitudini, linguaggi di interazione, design di rappresentazioni, design di caratteristiche e design di Situazioni Non Cooperative.

DESIGN AGENTS (2)

Metodi e attributi possono descrivere le competenze di un agente con una notazione stereotipata <<skill>>.

Le competenze sono le conoscenze del sistema che consente all'agente di svolgere un'azione.

Simili come le competenze, le attitudini, i linguaggi di interazione, design di rappresentazioni e design di caratteristiche sono definite con una notazione stereotipata.

Le attitudini sono le capacità degli agenti di ragionare su una specifica conoscenza del sistema o su una situazione reale.

DESIGN AGENTS (3)

Il designer analizza i protocolli definiti così come gli attribuiti per un agente sono associati ad uno stato-macchina.

I metodi e gli attributi di collegamento con un protocollo di interazione deve essere stereotipato <<interaction>>.

I metodi e gli attributi relativi alla fase di percezione e azione sono rappresentati da <<perception>> e <<action>> .

DESIGN SITUAZIONI NON COOPERATIVE

Il passo Design Situazioni Non Cooperative (NCS) è il più importante per l'attività di definizione degli agenti, perché si tratta di una capacità specifica degli agenti cooperativi.

Un modello guida il designer nelle definizioni di tutte le situazioni che sembrano essere “dannose” per l’atteggiamento sociale cooperativo dell’ agente.

Quando l'ambiente è imprevedibile o il sistema è aperto, gli algoritmi tradizionali falliscono perché il designer non trova algoritmi che considerano tutte le possibilità.

SITUAZIONI NON COOPERATIVE

L'obiettivo è costruire sistemi che fanno del loro meglio quando incontrano una difficoltà.

Questa difficoltà, nei programmi classici può essere considerata come un'eccezione.

Dal punto di vista dell'agente, chiamiamo eccezioni le Situazioni Non Cooperative quali incomprensioni, ambiguità, inutilità, conflitti e concorrenza. Il designer deve riempire le condizioni descritte per ogni NCS.

SITUAZIONI NON COOPERATIVE (2)

Il designer deve descrivere ciò che un agente deve fare per raggiungere il suo obiettivo locale, ma anche situazioni locali che l'agente deve evitare e se eventualmente individuate anche come rimuoverle.

Il designer deve compilare una tabella per ogni Situazione Non Cooperativa (NCS).

La tabella contiene lo stato dell'agente, la descrizione testuale dell' NCS, le condizioni e le azioni connesse con l'NCS.

Le condizioni descrivono i diversi elementi che consentono di rilevare il livello locale dell'NCS.

Le azioni descrivono ciò che l'agente deve fare per rimuovere l' NCS.

IDENTIFICAZIONE DELLA FORMA DELL' NCS

Name	Permission denied
State Description	Execute the activity An agent faces this situation when the activity that it intends to execute cannot be accomplished with the permissions of the user in question
Conditions Actions	User with no knowledge with the system. The agent must supply to the user a list of all the users who have connection with this and that they have permission to execute the task.

FAST PROTOTYPING

L'attività di Prototipazione Rapida usa OpenTool per testare il comportamento degli agenti.

La versione personalizzata di OpenTool può trasformare automaticamente un diagramma protocollo in uno stato-grafico che può essere eseguito per simulare il comportamento degli agenti.

Alcuni metodi possono essere implementati mediante un linguaggio OTscript che è un linguaggio con un insieme di azioni basate su OpenTool.

COMPLETE DESIGN DIAGRAMS

Prevede di completare l'architettura dettagliata arricchendo i diagrammi classe e sviluppando diagrammi stato grafico necessari per progettare il comportamento dinamico.

Obiettivo: riflettere sui diversi cambiamenti dello stato di un'entità quando interagisce con altri.

I diagrammi di classe sono perfezionamenti dei precedenti diagrammi di classe.

ADELFE non ha bisogno di fornire ulteriori funzionalità per costruirli perché il designer segue lo stesso metodo, come quello utilizzato nelle metodologie object-oriented.

SINTESI FASE DI DESIGN

A14: Study detailed architecture and multi-agent model

- S1: Determine packages
- S2: Determine classes
- S3: Use design-patterns
- S4: Elaborate component and class diagrams

A15: Study interaction languages

A16: Design Agents

- S1: Define skills
- S2: Define aptitudes
- S3: Define interaction languages
- S4: Define representations
- S5: Define Non cooperative situations

A17: FAST Prototyping

A18: Complete design diagrams

- S1: Enhance design diagrams
- S2: Design dynamic behaviors

IMPLEMENTAZIONE

Fase simile a ciò che viene fatto nel RUP.

Si pensa di fornire al designer uno strumento di prototipazione rapida che sarà integrato al software OpenTool.

OpenTool è uno strumento grafico che supporta la notazione UML.

Al momento la metodologia è stata testata solo su piccoli esempi ma si pensa di ingrandire le sue applicazioni.

CONTRIBUTI E LIMITI

Copre tutto il ciclo di vita “classico” del software e i riusi delle notazioni UML.

E' anche una metodologia ricorsiva o iterativa, quando il designer può implementare un agente deve riutilizzare l'intera metodologia per svilupparlo.

Guida il designer sulla scelta del se e dove la tecnologia è richiesta nel sistema che si stà sviluppando. Ciò spiega l'importanza del controllo di adeguatezza nella fase d'analisi.

Aiuta a trovare quali componenti del sistema chiedono di essere trattati come agenti appartenenti alla teoria AMAS (agenti cooperativi).

CONTRIBUTI E LIMITI (2)

La specifica architettura agente può essere vista come uno o più design pattern forniti al designer.

Guida a costruire agenti; fornisce una funzionalità per dotare il comportamento di un agente con il modello NCS.

Attualmente, è molto difficile elencare tutte le possibili azioni per ogni stato dell'ambiente.

CONTRIBUTI E LIMITI (3)

- ↓ Non consente la progettazione di ogni applicazione agent-based (può essere eliminato dall' accoppiamento con un'altra metodologia).
- ↓ Non è general purpouse.
- ↓ Soffre dalle limitazioni provenienti da UML (molti modelli per capire un sistema).

Se non c'è un tool di supporto che propaghi una modifica diventa molto oneroso cambiare anche piccole cose, dato che la complessità dei sistemi ad agenti è elevata.

- ↓ Non c'è concetto di ruolo nel sistema.

VALUTAZIONE DI ADELFE

Segue la tabella che presenta un'analisi generale della metodologia ADELFE. Mostra le domande (i concetti) e le risposte per alcuni principali aspetti valutati come “S”, “N” o “W”.

“S”: la questione è supportata dall'approccio, la metodologia fornisce abbastanza costrutti, meccanismi e orientamento.

“N” : sebbene l'approccio supporta gli aspetti affrontati, fino a una certa misura, solo parzialmente fornisce modi per modellare quell'aspetto.

“W” : l'approccio non fornisce costrutti, meccanismi e orientamento o ne fornisce molto pochi.

ESEMPI DI DOMANDE DI VALUTAZIONE DI ADELFE

	ADELFE
QA1 - Proactiveness	S
QA2- Human Autonomy vs software autonomy	S
QA3 - Autonomy reasoning	S
QA4 - Different levels of Abstraction	N
QA5 - Identifying participants in the domain	N
QA6 - Capturing, understanding and registering terminology	N
QA7 - Domain analysis	N
QA8 - Finding requirements	S
QA9 -Human-machine cooperation	S
QA10 - Database design	N
QA11 - Database evolution	N
QA12 - Database design and legacy	N
QA13 - Reasoning about different non-functional aspects	S
QA14 - Mobility	W
QA15 - User interface design	N
QA16 - Generating test cases	N
QA17 - User interface design (usability)	N
QA18 - Architectural design and reasoning	N
QA19 - Eliciting and reasoning about Non-Functional aspects	N
QA20 - Architectural design and reasoning (flexibility)	N
QA21 - Architectural design and reasoning (cost and confidence)	N
QA22 - Validating specification over the life cycle	S
QA23 -Tracing changes in the requirements into design	W
QA24 - Tracing changes from design to code	N
QA25 - Concurrency	S
QA26 - Tracing back to requirements	W
QA27 - Software Modularity	N
QA28 - Formal Verification and Validation	N
QA29 - Project Management	W
QA30 - Working in distributed teams	N
QA31 - Tool support	S
QA32 - Learning curve	N
QA33 - Integration with other methodologies	N
QB7 - Lightweight versions of methodology(simpler problems)	N

CONFRONTI CON ALTRI APPROCCI

La metodologia di design dovrebbe essere interessata a sviluppare una metodologia dettagliata e completa dall'analisi fino all'implementazione e manutenzione software.

E' riconosciuto che i flussi di lavoro di base di una metodologia sono i requisiti, l'analisi, il design e lo sviluppo (chiamato anche implementazione) e il dispiegamento del flusso di lavoro.

Alcune metodologie coprono l'intero processo di ingegneria del software come TROPOS, MESSAGE e ADELFE.

CONFRONTI CON ALTRI APPROCCI (2)

- TROPOS insiste sulla fase dei requisiti divisa in requisiti preliminari e finali.
- MESSAGE copre almeno le fasi di analisi e design, ma dovrebbe anche spiegare le relazioni con l'implementazione, il controllo e la distribuzione.
- In GAIA, la fase di analisi consiste nel riempire i modelli ruolo e interazioni. L'agente, i modelli servizio e conoscenza sono sviluppati nella fase di design.

CONFRONTI CON ALTRI APPROCCI (3)

MESSAGE e ADELFE forniscono orientamenti per l'identificazione delle aree di applicazione dove la tecnologia agente per MESSAGE e la tecnologia per sistemi adattativi ADELFE è più adatta di altre tecnologie ad esempio tecnologie object-oriented.

In ADELFE i designer non specializzati in tecnologia agente/MAS sono guidati da un tool che li aiuta a decidere se è utile per il design del software. Differisce dalla maggior parte delle metodologie in cui la domanda: “Il problema richiede una tecnologia ad agente o MAS?” Non è chiesto perché l'ipotesi iniziale è che l'agente o le tecniche multi-agente sono disponibili e rilevanti per l'applicazione.

CONFRONTI CON ALTRI APPROCCI (4)

- In ADELFE, a livello agente nella fase di design, il modello agente proposto per il designer è un'architettura agente cooperativo.
- Nelle metodologie come GAIA, MESSAGE, TROPOS l'architettura di agenti implementati non è definita ed è abbastanza aperta.

TROPOS offre diversi stili di architettura (struttura piatta, piramide, joint-venture ...) per la sua fase di design architeturale.

MESSAGE e ADELFE sono basate sulla metodologia RUP e usano la notazione UML o l'estensione di UML già fatto in AUML in particolare le notazioni AIP.

CONFRONTI CON ALTRI APPROCCI (5)

- TROPOS adotta la metodologia i^* per la fase dei requisiti e AUML per il design dettagliato. Esprime la dinamica e l'apertura dell'applicazione nella fase dei requisiti con il modello di ambiente e con particolari goal.
- GAIA è basata sulla metodologia FUSION. Indica che il dominio oggetto della metodologia è statico e che la metodologia è dedicata al dominio chiuso in cui le competenze e le credenze degli agenti sono statiche a runtime.
- In ADELFE l'aspetto dinamico è considerato a 3 livelli:
 - a livello interazioni sistema-ambiente;
 - a livello di sistema di composizione;
 - a livello agente.

CONFRONTI SUL CONCETTO DI RUOLO

- Il ruolo è un concetto importante, rappresenta un livello astratto della descrizione dell'agente.

I ruoli sono definiti nella fase di analisi delle metodologie GAIA, MESSAGE e TROPOS.

- In ADELFE la definizione dei ruoli non è necessaria per sviluppare il sistema perché l'agente potrebbe cambiare i suoi ruoli a run-time.

Se gli agenti nel sistema possono essere osservati a run-time, l'osservatore può dare loro dei ruoli.

Il concetto più vicino al ruolo è il concetto di competenza.

Le competenze non rappresentano il ruolo perché sono azioni elementari che un agente è in grado di fare.

CONFRONTI SUL CONCETTO DI AMBIENTE

- La modellazione dell'ambiente non è un punto centrale nelle metodologie esistenti.

In TROPOS e ADELFE, è elaborato nella fase dei requisiti.

In TROPOS è descritto in termini di attori, obiettivi e interdipendenze.

In ADELFE in termini di attori attivi e passivi e i diagrammi sequenza esplicitano il tipo di scambi tra sistema e ambiente.

In MESSAGE, il modello dominio cattura alcune entità del sistema ambiente e le interazioni con l'ambiente sono descritte per ogni ruolo in termini di input sensoriali e conoscenze, le risorse di proprietà e accessi, task e azioni.

In GAIA non vi è modello di ambiente.

CONFRONTI SULL' IDENTIFICAZIONE DEGLI AGENTI

In GAIA gli agenti sono già identificati e la metodologia non prevede nulla.

TROPOS non fornisce linee guida per identificare gli agenti, si trovano all'interno di un insieme di attori.

In MESSAGE e ADELFE, la definizione di agente definisce le caratteristiche che saranno attribuite alle entità che il designer sceglierà di considerare come agenti.

In ADELFE, dopo la fase di adeguatezza, il designer conosce il sistema che deve essere adattivo. Gli agenti devono essere trovati nella parte del sistema in cui l'adattabilità è richiesta.

CONCLUSIONI

Poche delle metodologie esistenti agent-oriented trattano concetti come il dinamismo o l'ambiente in evoluzione.

- *L'obiettivo* del presente approfondimento è stato di presentare la metodologia ADELFE che è una metodologia multi-agent-oriented adatta a sistemi multi-agente adattivi basati sulla teoria AMAS.

CONCLUSIONI (2)

ADELFE:

- può essere utilizzata anche da un designer non specializzato nella costruzione di AMAS;
- prende in considerazione l'alta dinamica dell'ambiente attraverso l'applicazione della teoria AMAS;
- utilizzando l'attività di adeguatezza è in grado di dire se è necessario un'AMAS per costruire il proprio sistema;
- aiuta il designer a trovare quali componenti del sistema devono essere trattati come agenti cooperativi e guida a costruirli.

CONCLUSIONI (3)

ADELFE è stata sviluppata per applicazioni con:

- ambiente dinamico, rendendo inefficace l'enumerazione esaustiva di tutte le situazioni che il sistema può incontrare;
- sistema aperto, costituito da uno spostamento del numero di componenti;
- task del sistema complesso da raggiungere da non riuscire a garantire un perfetto design;

Quindi si tratta di un metodologia idonea per lo sviluppo di sistemi multi-agente adattivi.

SVILUPPI FUTURI

- Si prevede lo sviluppo di una nuova metodologia per il design di una società di agenti che mostrano un'attività coerente, consentendo di auto-adattarsi all'ambiente.

Finora, è stata utilizzata in due casi di studio: il design di un sistema Intranet e di un timetabling problem.

- si prevede la sua sperimentazione in un progetto europeo (SYNAMEC) per sviluppare un sistema meccanico auto-progettato.
- E' destinato ad essere un toolkit di sviluppo software con nuove funzionalità e non solo una “semplice” metodologia.

Sarà in grado di:

- fornire strumenti e librerie per facilitare design e sviluppo dei sistemi.
- assistere il designer a capire se sono più adatte altre metodologie a costruire il sistema che desidera.

BIBLIOGRAFIA

- ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering.
Carole Bernon, Marie-Pierre Gleizes, Sylvain Peyruqueou, Gauthier Picard.
- Evaluating ADELFE Methodology in the Requirements Identification.
V.M. B. Werneck, A. Y. Kano, L. M. Cysneiros.
- ADELFE Design, AMAS-ML in Action. A Case Study.
Sylvain Rougemaille, Jean-Paul Arcangeli, Marie-Pierre Gleizes, Frédéric Migeon.
- The ADELFE Methodology For an Intranet System Design.
Carole Bernon, Marie-Pierre Gleizes, Gauthier Picard, Pierre Glize.
- Le metodologie: seminario
A.Molesini