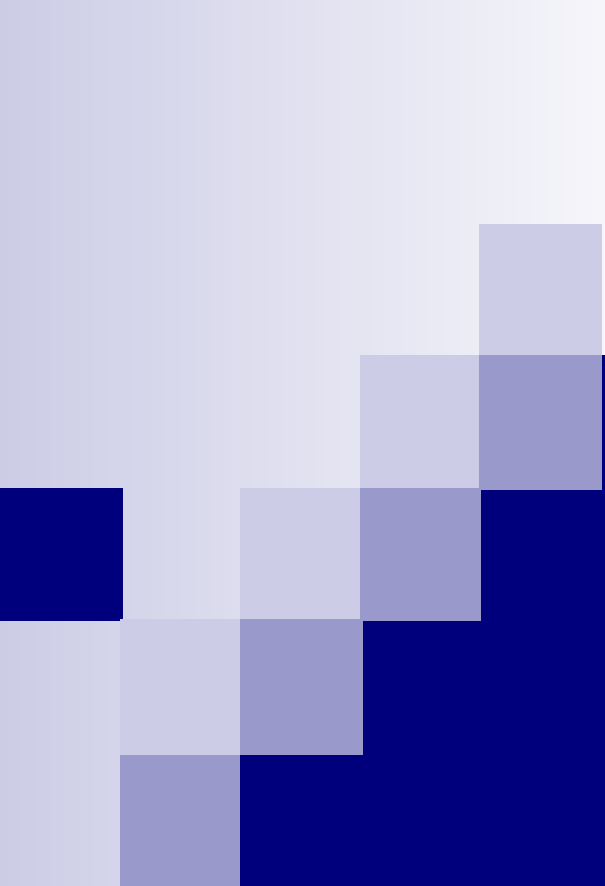




SODA vs. TROPOS

Approfondimento svolto da:
Magnani Monica

Sistemi Multiagente L.S.

Perché questo approfondimento?

- I sistemi ad Agenti sono relativamente nuovi
- Ogni metodologia ha un certo scopo, certe metafore che lo ispirano e lo rendono più adatto per uno scopo
- Quindi: confrontare tante (intanto 2) metodologie tra loro aiuta anche a scegliere la metodologia giusta per il problema giusto, perché non esiste la metodologia ideale, cioè ancora le metodologie ad agenti presentano alcune lacune (ad esempio SODA non dice come è fatto internamente un agente). Perciò si sta cercando di poter integrare all'interno di un processo di sviluppo più metodologie.



Metodologia

- Scenario in cui si sviluppa una metodologia:
 - Gap enorme tra i sistemi che gli ingegneri del software, programmatori devono risolvere e le tecnologie che hanno a disposizione



Metodologia

- “A methodology is a collection of methods covering and connecting different stages in a process” [Ghezzi]
- Lo scopo di una metodologia è quello di prescrivere un certo approccio coerente per risolvere un problema nel contesto di un processo software attraverso la selezione e il mettere in relazione una serie di metodi.



Metodologia

Le metodologie ha due importanti componenti:

- *Astrazioni*: gli elementi del processo dell'approccio che si va ad utilizzare (ad es. Agenti e Artefatti)
- *Processo*: mi deve dire cosa devo fare in pratica, cioè tutti i passi che devono essere fatti

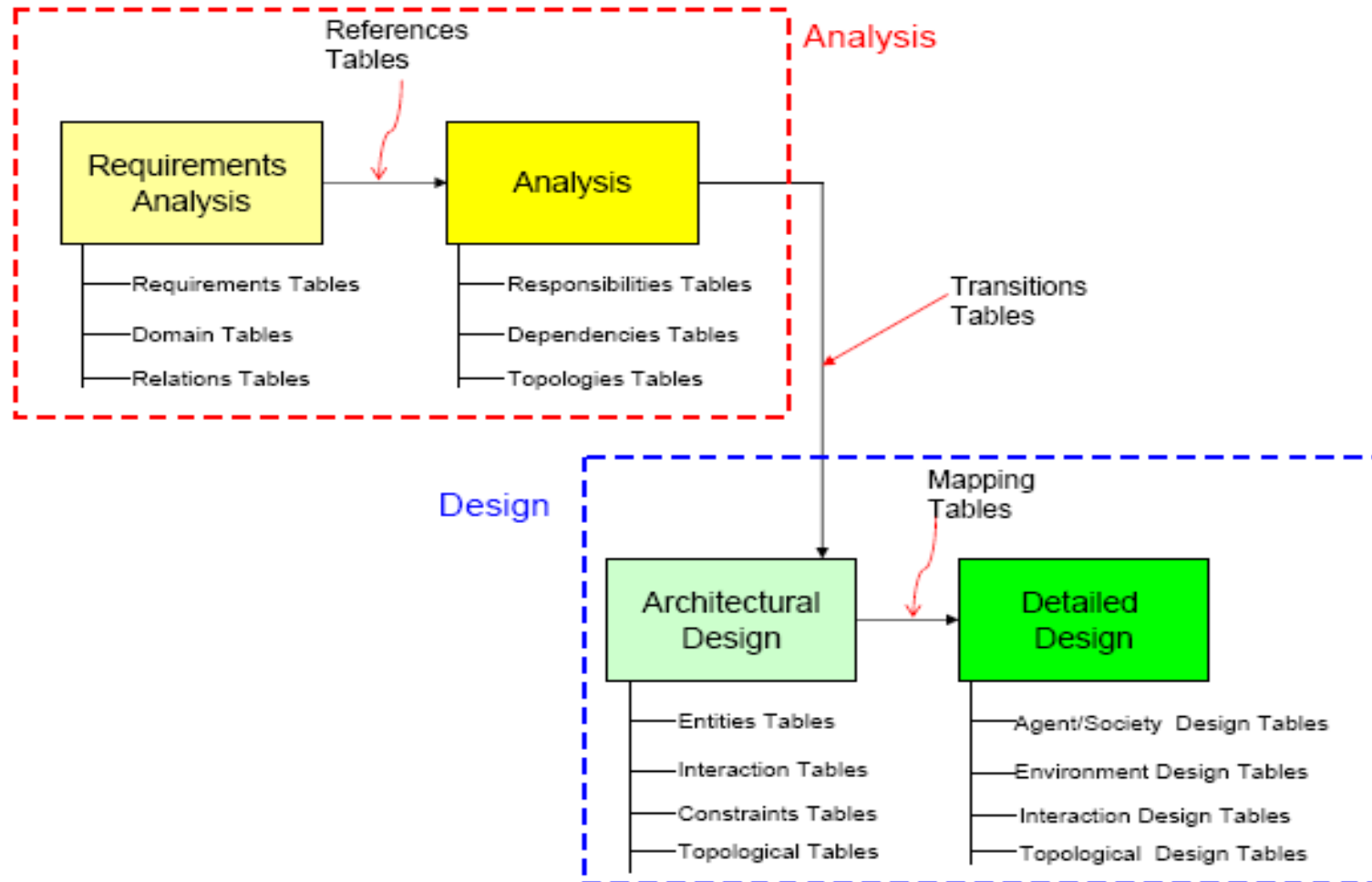
Meta-Modello

- Il Meta-Modello è l'analisi, costruzione e sviluppo dei frames, regole, vincoli, modelli e teoria applicabili e utili per la modellazione di una classe predefinita di problemi.
- Utilizzato per costruire una rappresentazione standard che mi aiuterà per confrontare più metodologie o unirle tra di loro.
- Una meta-modello è qualcosa che abilita il check e la verifica della completezza e dell'espressività di una metodologia e soprattutto che mi permette di capire la sua semantica, come relazioni che ci sono tra i suoi concetti, e anche in qualche modo di andare a vedere se in una metodologia ci possono essere o meno dei buchi.



SODA

SODA



SODA

- I principi ispiratori di SODA sono due:
 - La parte di modellazione dello spazio di interazione degli agenti quindi come gli agenti comunicano tra di loro e come comunicano con l'ambiente
 - Modellare e progettare l'ambiente (NOTA: SODA considera l'ambiente fin dalla parte dei requisiti al contrario di diverse metodologie)

A&A in SODA

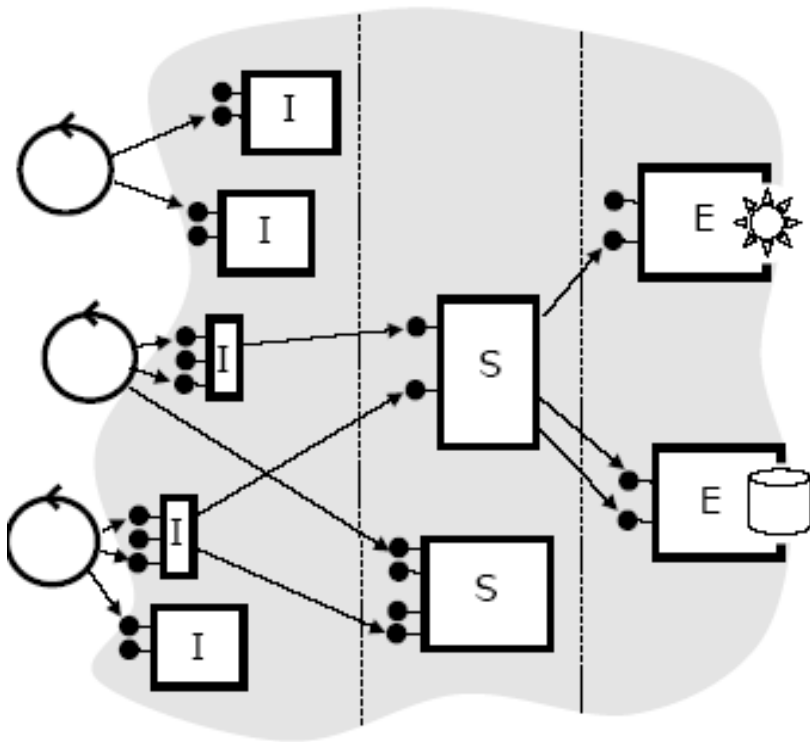
- *Agenti*: che modellano attività individuali e sociali
- *Artefatti*: che “incollano” gli agenti tra loro e anche il MAS con l’ambiente. Quindi:
 - Mediano tra gli agenti e il MAS
 - Costituiscono la società degli agenti
 - Wrappano le risorse esterne portandole a livello cognitivo dell’ambiente (in questo modo sono in grado di progettare meglio tutte le interazioni che ci sono)
- *Workspace*: in cui in SODA ci arriviamo solo nella fase di Detailed Design. Gli agenti come gli artefatti li possiamo vedere come entità solo a livello implementativo altrimenti sarebbe sbagliato concettualmente dal punto di vista dell’ingegneria del software.(Perché in analisi dei requisiti si deve modellare il problema non la soluzione)



Artefatti in SODA

- Gli artefatti li possiamo pensare come qualcosa che hanno la forma di oggetto o tool che gli agenti usano, condividono per:
 - Supportare le loro attività
 - Portare a termine i loro obiettivi

Classificazione Artefatti in SODA



- Artefatti individuali
- Artefatti sociali
- Artefatti ambientali

Astrazioni in SODA

SODA caratterizza le astrazioni in tre parti:

- Prima categoria modella tutto ciò che possiamo definire come parte attiva del sistema
- Seconda categoria: modella la parte reattiva e funzionale
- Terza categoria modella le interazioni e anche la parte chiamata parte organizzazionale ma anche la parte normativa del sistema

In ognuno dei modelli delle fasi di SODA ci sarà sempre almeno ogni astrazione per ogni categoria, a partire dai requisiti.

Astrazioni in SODA

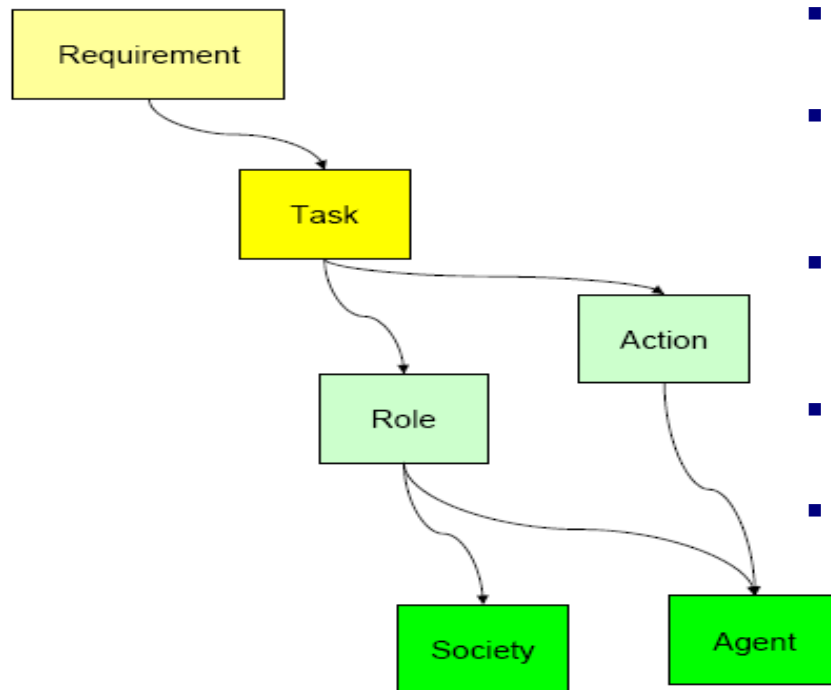
- *Fase di Analisi:*

- Analisi dei requisiti: I requisiti del sistemi e l'ambiente esterno sono analizzate e modellate
- Analisi: I requisiti del sistema sono modellati in termini di task, funzioni, topologie e dipendenza

- *Fase di Design:*

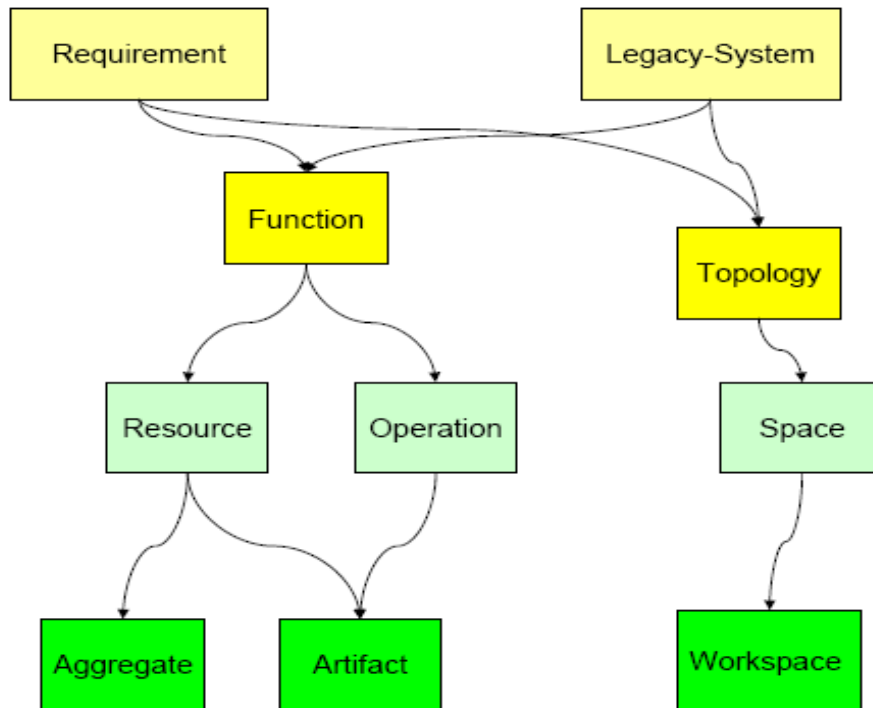
- Design Architettuale: in questa fase noi analizziamo il dominio della soluzione, il sistema è modellato in termini di regole, azioni, risorse, operazioni, interazioni
- Detailed Design: in questa fase noi progettiamo il sistema in termini di agenti, società, artefatti, composizioni, aggregazioni, workspace, usi, Link to, manifestarsi e Speak to.

Astrazioni Attive in SODA



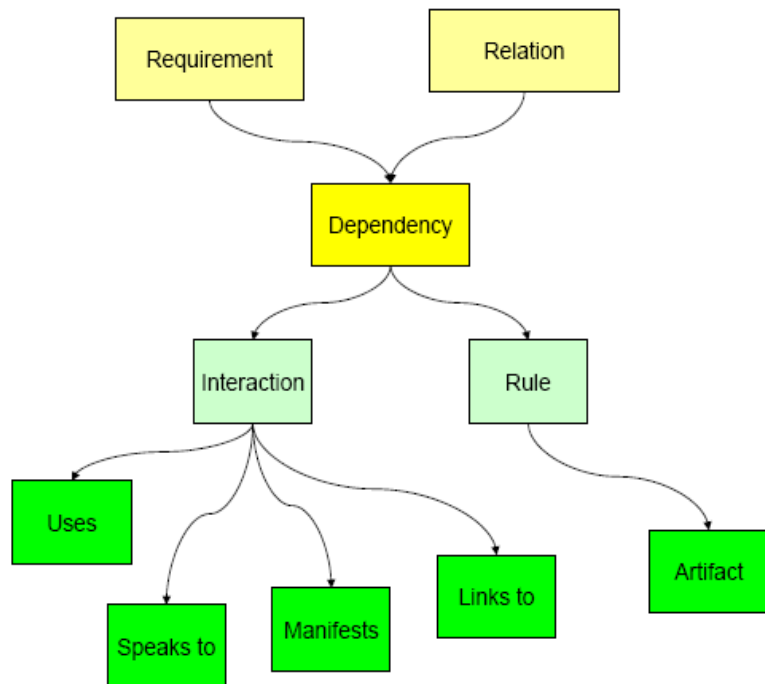
- **Requisiti**
- **Task:** definisce le funzionalità che derivano da quel requisito
- **Azioni:** definisce le azioni che si devono fare per riuscire a ottenere quel task
- **Ruoli:** assegna la responsabilità di eseguire quella determinata funzionalità
- **Società:** riunisce i ruoli che cooperano tra loro in una società
- **Agente:** mappa uno o più ruoli a un agente. Inoltre all'agente saranno assegnate tutte le azioni che dovrà compiere per portare a termine il task

Astrazioni Reattive



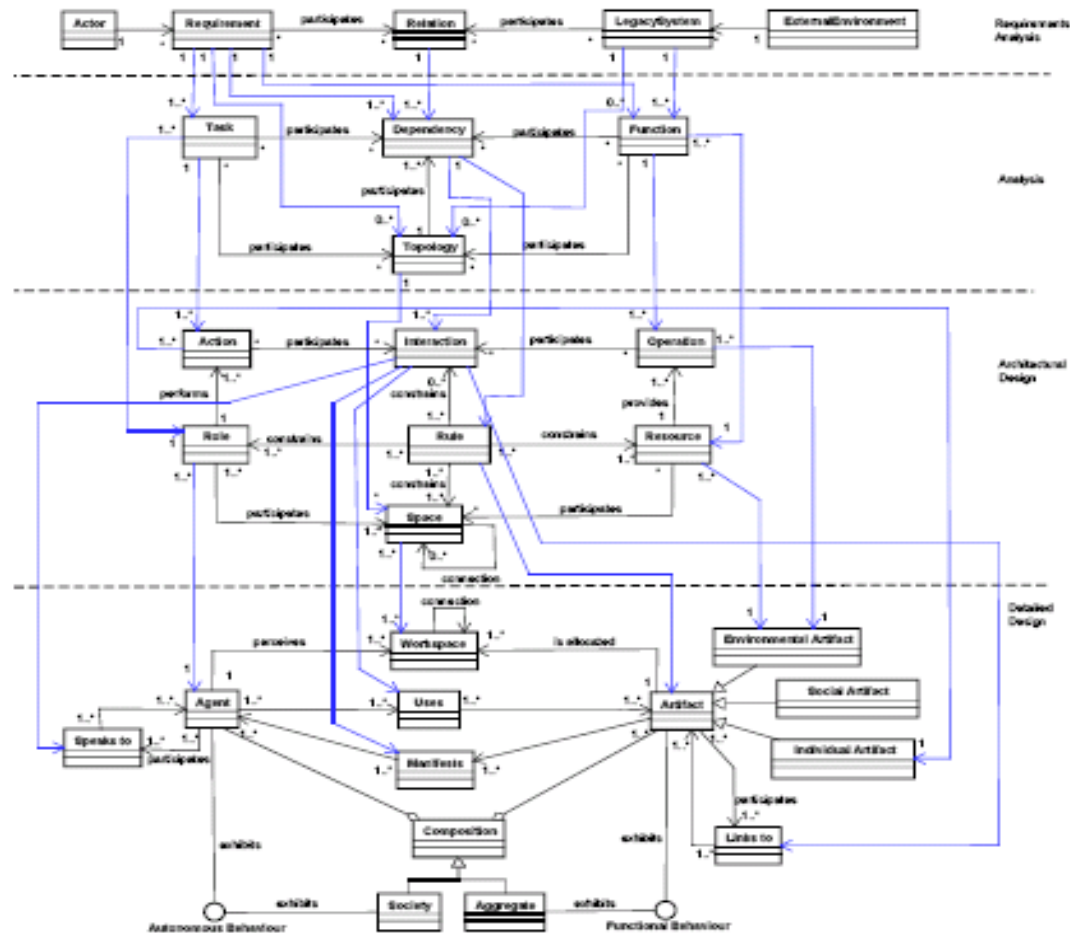
- **Requisiti:** da utilizzare per progettare qualcosa di nuovo
- **Legacy-System:** rappresentano quello che c'è già nell'ambiente
- **Function:** derivano sia dai requisiti che dai Legacy-System. Rappresentano le funzionalità che gli artefatti dovranno rendere disponibili a partire dai requisiti
- **Topologia**
- **Risorsa:** le funzioni possono essere affidate a questa astrazione
- **Operazioni:** servono per “raffinare” le funzioni per poi metterle a disposizione
- **Artefatti:** La risorsa andrà a generare gli artefatti con cui andrò a eseguire tutte le interazioni che andrò a progettare
- **Aggregato:** è la controparte della società nella parte reattiva.
- **Spazi:** la topologia viene mappata negli spazi in cui andrò a organizzare il sistema in cui andrò ad allocare tutte le parti reattive
- **Workspace:** saranno i nodi, spazi virtuali

Astrazioni di Interazioni



- **Requisiti:** ci guidano molto a livello di interazione
- **Relazione:** entità che in qualche modo collega tra loro i requisiti e i Legacy-System. (I requisiti devono dipendere dai legami che ci sono già quindi si inizia a definire ad alto livello le prime interazioni)
- **Dipendenze:** i requisiti e le varie relazioni concernano nelle dipendenze
- **Interazione:** atto fisico di comunicazione
- **Regole**
- **Utilizzo:** ad es. l'agente usa l'artefatto
- **Speaks to:** ad es. l'agente può parlare con altri agenti
- **Manifestarsi** dell'artefatto
- **Link to:** più artefatti
- **Artefatti:** è l'entità che dovrà far rispettare le regole (ad es. gli artefatti di coordinazione)

Meta-Modello

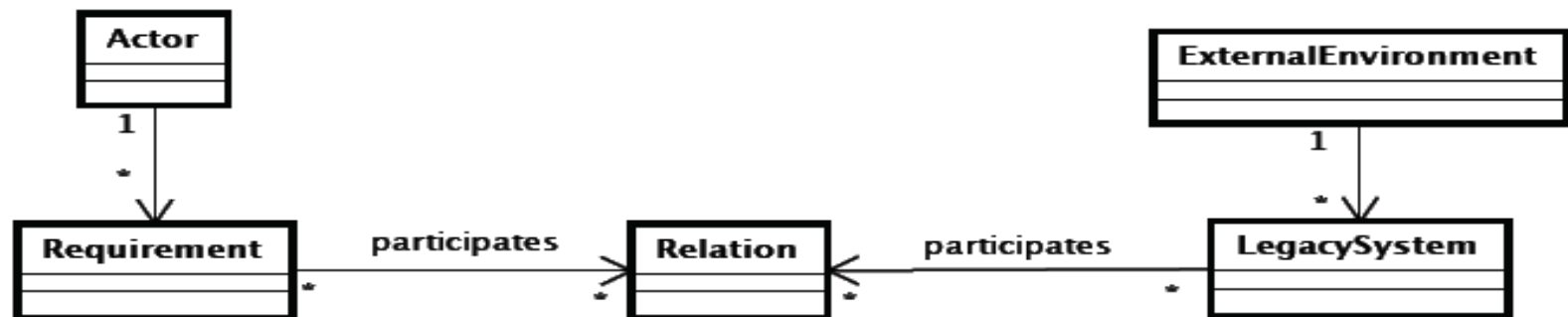




Visioni Possibili sul Meta-Modello

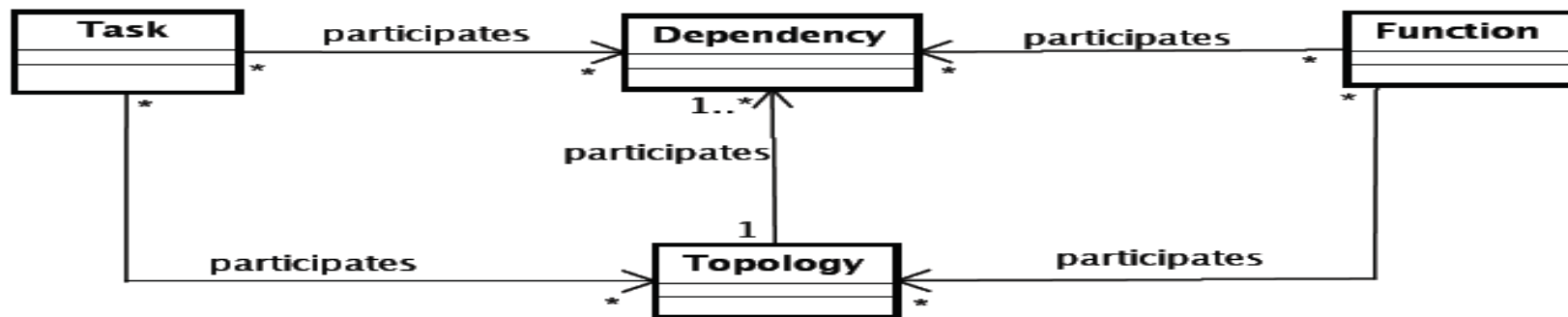
- Visione Verticale del processo
- Visione Orizzontale all'interno delle singole fasi.

Analisi dei requisiti nel Meta-Modello



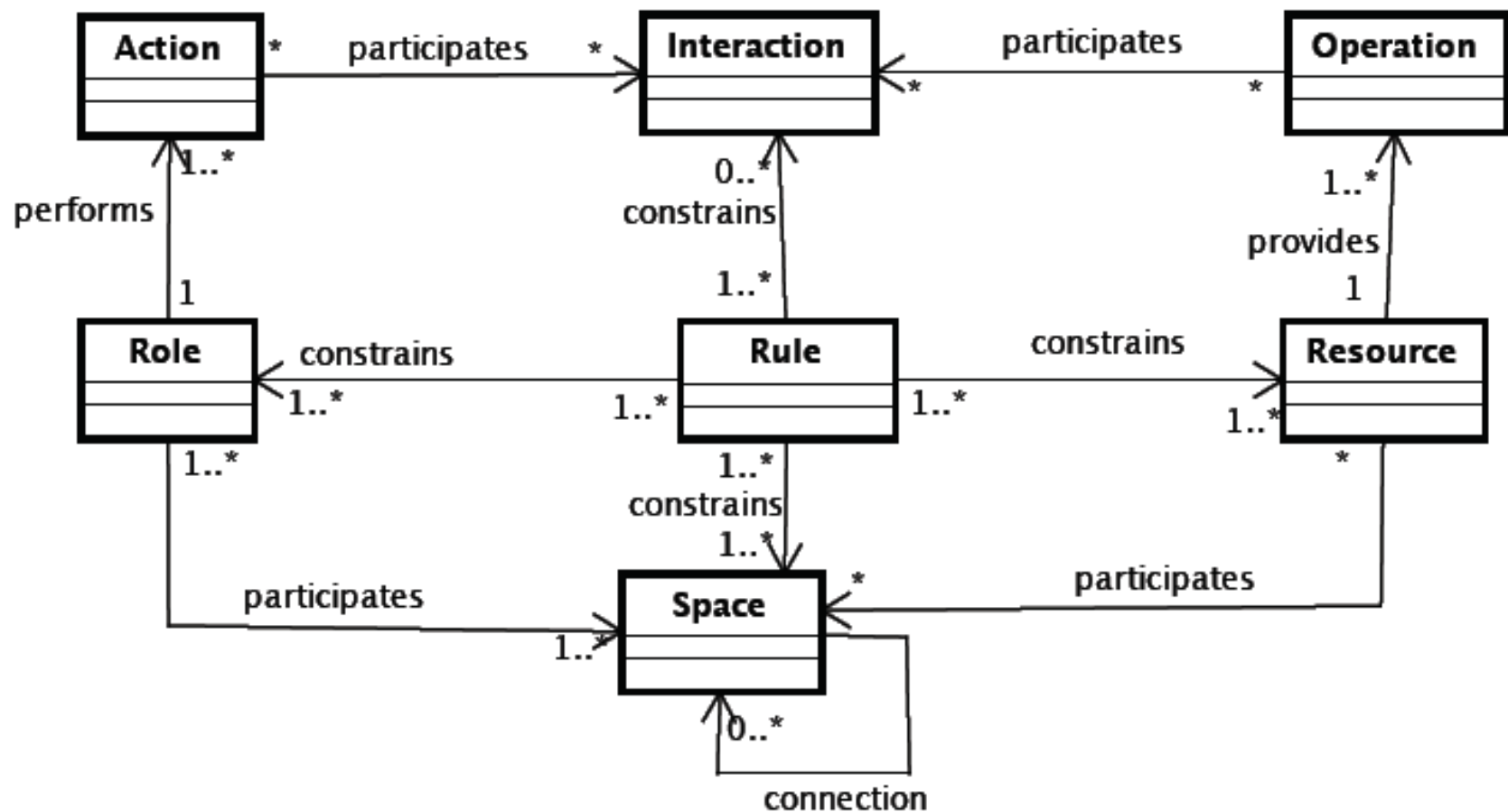
- **Actor e ExternalEnvironment:** indicano quali sono i soggetti da cui sono arrivati certi requisiti e legacySystem
- **Relation:** In soda è importante l'interazione anche nella modellazione perciò con questa classe vogliamo definire il fatto che ci sia una relazione tra requisiti e i legacySystem (e non solo, anche ad es. tra due requisiti). Viene usato proprio come formalismo una classe perché si vuole rappresentare l'interazione come unità di prima classe.

Analisi nel Meta-Modello



- **Task:** attività che deriva dai requisiti. A volte ha bisogno del supporto di alcune funzioni.
- **Dipendenza:** rappresenta le relazioni ad un livello astratto
- **Funzioni:** attività reattiva
- **Topologia:** Può derivare dalle funzioni, può influire e creare delle regole che mi serviranno nella fase successiva e può anche andare ad influire sulle dipendenze.

Design Architettrale nel Meta-Modello

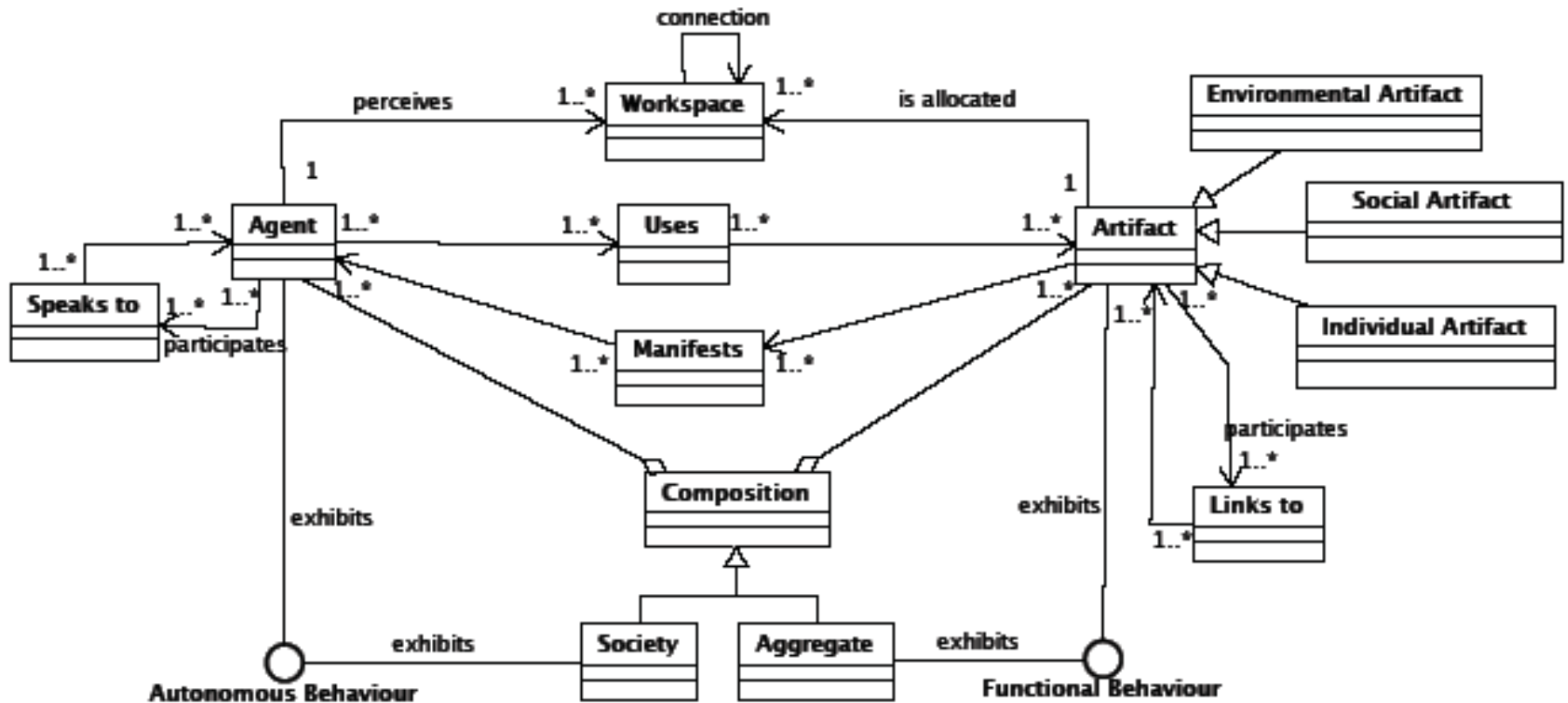




Design Architettrale nel Meta-Modello

- **Regole:** definiscono le astrazioni responsabili per lo svolgimento di uno o più task
- **Risorsa:** astrazione che fornisce alcune funzioni
- **Azioni:** è l'attività che un ruolo potrebbe svolgere
- **Operazioni:** rappresenta l'operazione che una risorsa può fornire
- **Interazione:** rappresenta l'atto di interazione tra ruoli, tra risorse e tra ruoli e risorse
- **Ruoli**
- **Spazi**

Detailed Design nel Meta-Modello



Detailed Design nel Meta-Modello

- **Agente:** entità autonoma a cui sono assegnati dei ruoli
- **Artefatto:** mette a disposizione degli agenti delle funzioni
- **Aggregato:** è un insieme di artefatti (se lo vedo come scatola chiusa vedrò che il suo comportamento ai “morsetti” è come quello di un singolo artefatto solo più complesso)
- **Società:** rappresenta l’aggregazione di più agenti, ed esibisce un comportamento autonomo come se fosse un agente
- **Composition:** serve per dire che Società e Aggregati sono la composizione di agenti e Artefatti
- **Workspace:** è l’ambiente concettuale del sistema
- Poi ci sono tutte le interazioni possibili tra agenti e agenti, tra artefatti e artefatti e tra agenti e artefatti:
 - **Uso**
 - **Link to**
 - **Speak to**
 - **Manifestare**

Processo di Sviluppo

- [Fuggetta]

*The **software development process** is the coherent set of policies, organisational structures, technologies, procedures and deliverables that are needed to conceive, develop, deploy and maintain a software product*

“Il processo di sviluppo del software è l’insieme coerente di politiche, strutture organizzazionali, tecnologie, prodotti e decisioni che sono necessari per concepire, progettare, sviluppare e mantenere un prodotto software”

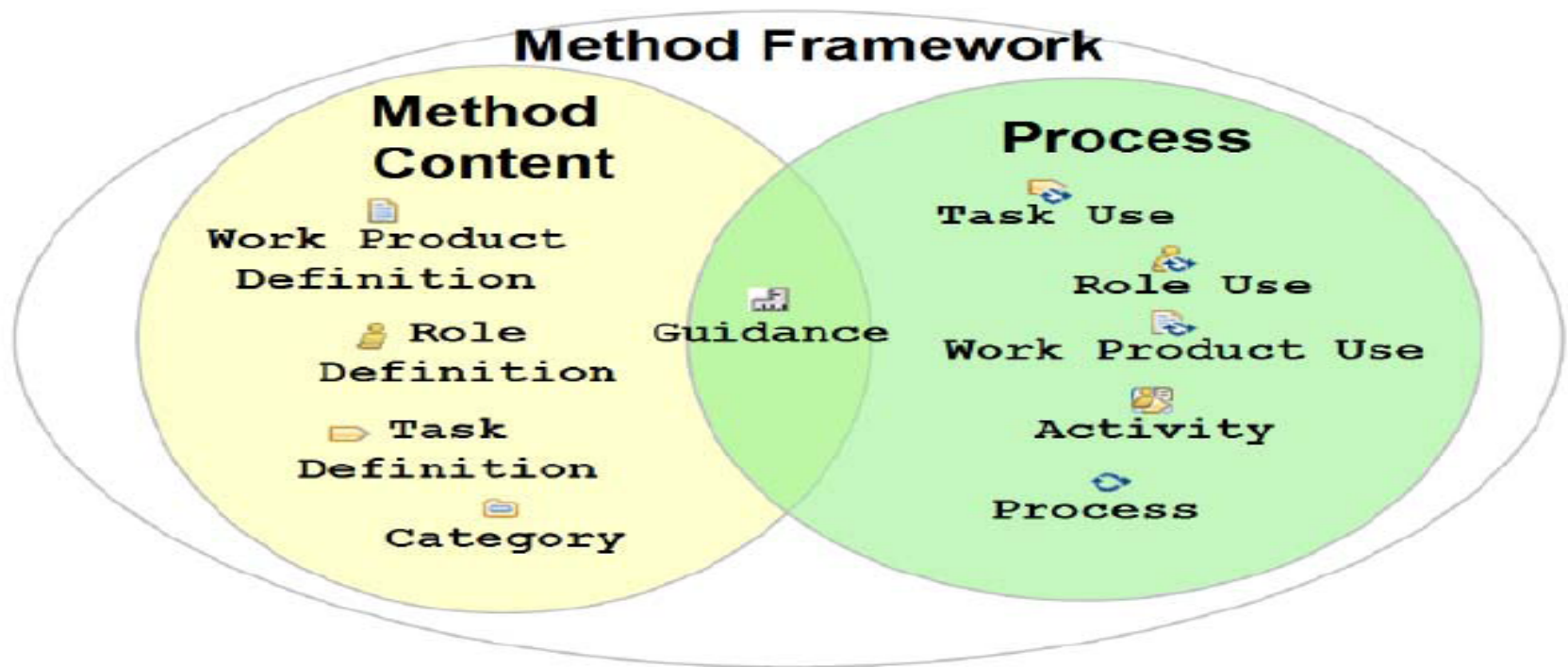
Come si rappresenta il processo di sviluppo?

- Si è avuto bisogno di una specie di UML orientato all'attività dell'essere umano che produce software, non alla rappresentazione del software ma alla rappresentazione del processo di costruzione del software. Il linguaggio che viene utilizzato è SPEM.

SPEM

- SPEM è basato sull'idea che il processo di sviluppo del software è una collaborazione tra activity abstract enties chiamate ruoli che fanno cose chiamate attività e producono, o lavorano su cose chiamate work products. (Impostazione Activity Theory)
- Gli scopi di SPEM
 - Rappresentare uno specifico processo di sviluppo
 - Supportare il mantenimento di diversi processi tra di loro non correlati
 - Fornire meccanismi e strumenti per avere famiglie di processi correlati che possono essere riutati e manipolati mantenendo la consistenza
- SPEM ci da uno strumento per costruire framework concettuale e anche una notazione pragmatica per poter eventualmente unire metodologie diverse.

SPEM



Questa è un'ontologia del meta-modello cioè quali sono i concetti che uso per definire questo meta-modello che poi uso per mettere insieme le metodologie

Notazione di SPEM

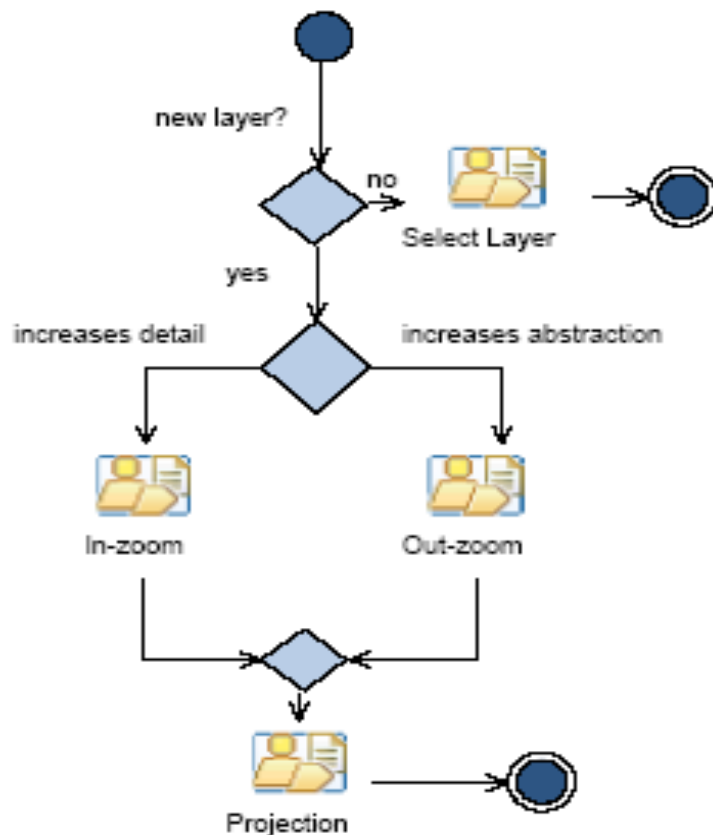
Stereotype	Symbol
Activity	
Category	
Composite role and Team	 
Guidance	
Milestone	
Process	
Process Component	
Process Pattern	
Role Definition and Use	 
Task Definition and Use	 
Tool Definition	
WorkProduct Definition and Use	 



Processo di SODA

- L'organizzazione in strati, *layering*, con livelli di astrazione diversi è un modo sufficientemente interessante per pensare di costruire sistemi artificiali complessi computazionali.
- SODA supporta un meccanismo di stratificazione.

Layering in SODA



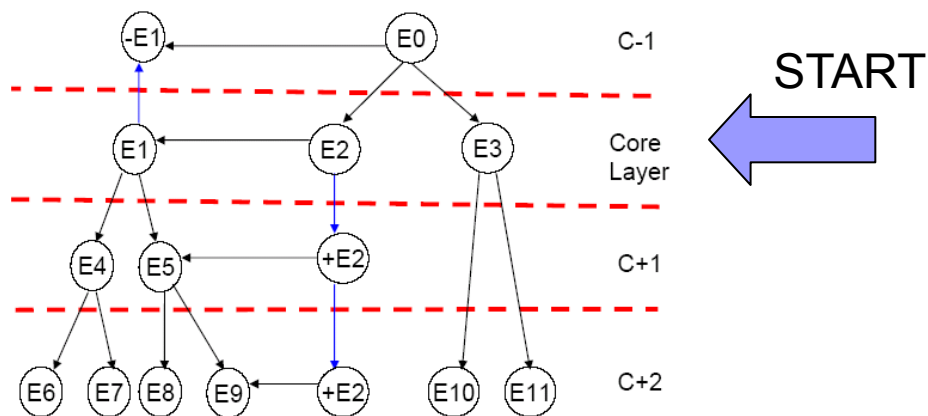
- Nel meta-modello se vogliamo rappresentare il processo di stratificazione a livello di processo servono due tipi di zoom:

- Inzoom: è quello che mi consente di aumentare il livello di dettaglio
- Outzoom: è quello che mi consente di aumentare il livello di astrazione

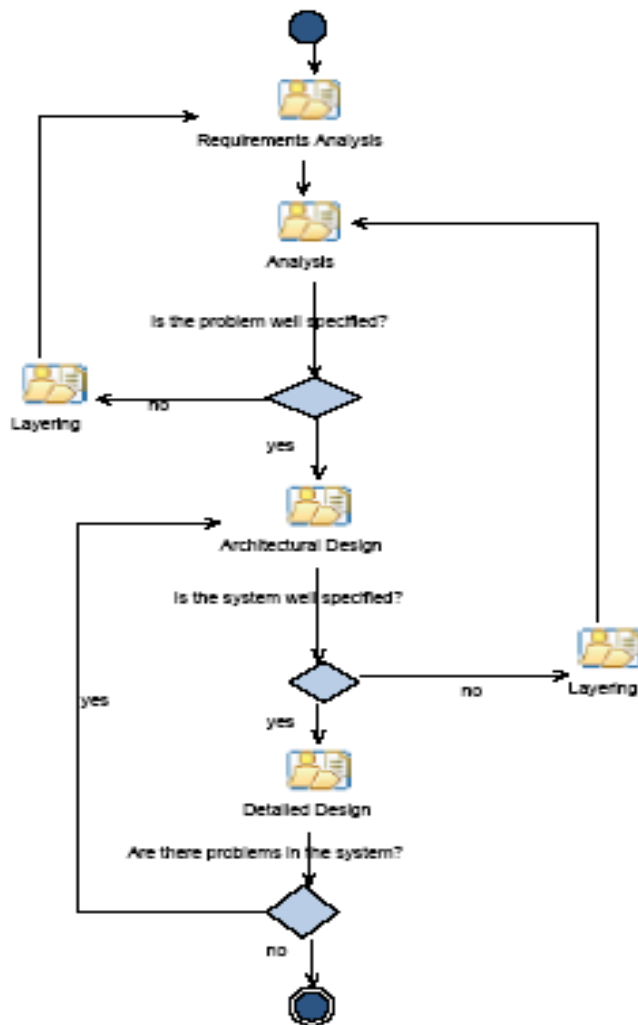
e un project mechanism per proiettare una entità nei vari livelli.

Viste del sistema

- Riesco in questo modo ad avere due viste del sistema:
 - Vista orizzontale: si riesce a capire quali sono le entità a un determinato livello
 - Vista verticale: si riesce a capire come può essere suddivisa una determinata entità ad un livello in più di dettaglio o a che astrazione appartiene a un livello di astrazione più alto.



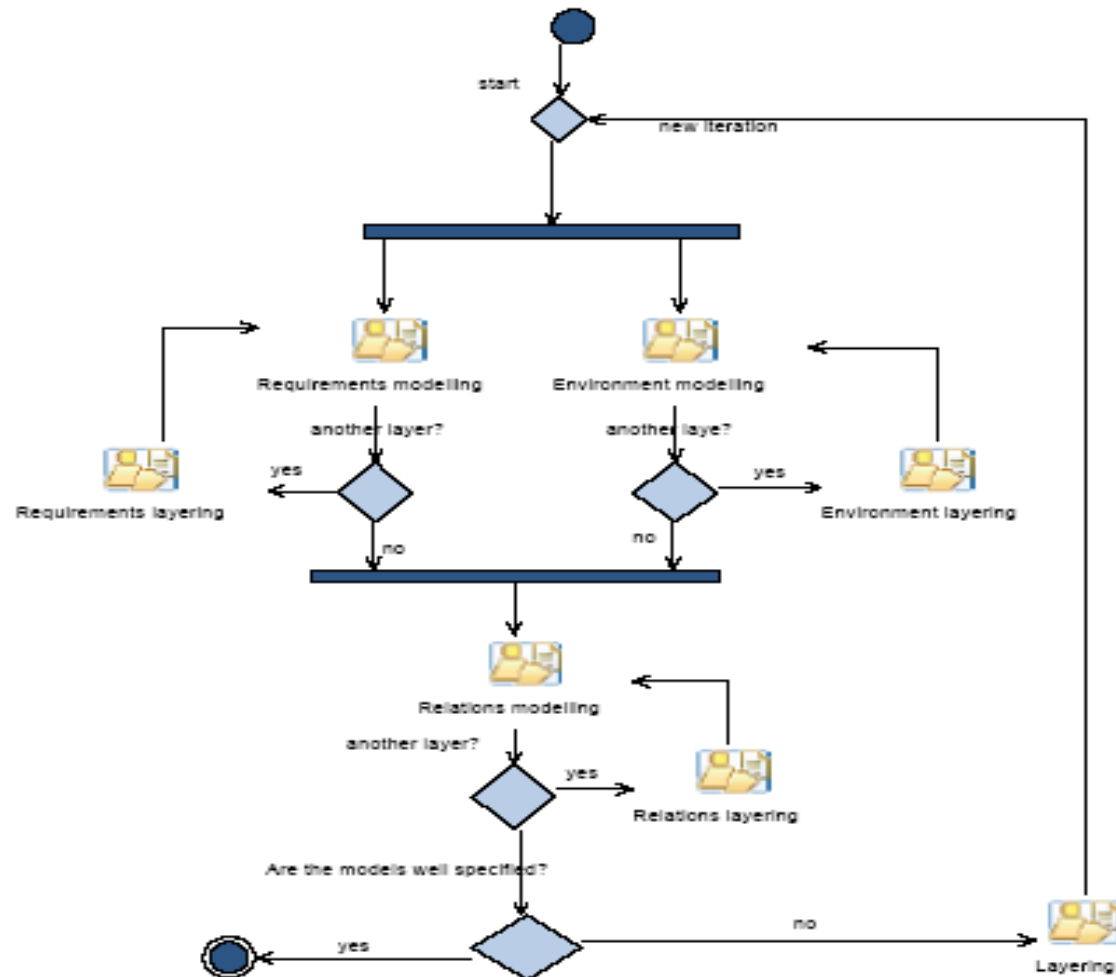
Processo SODA



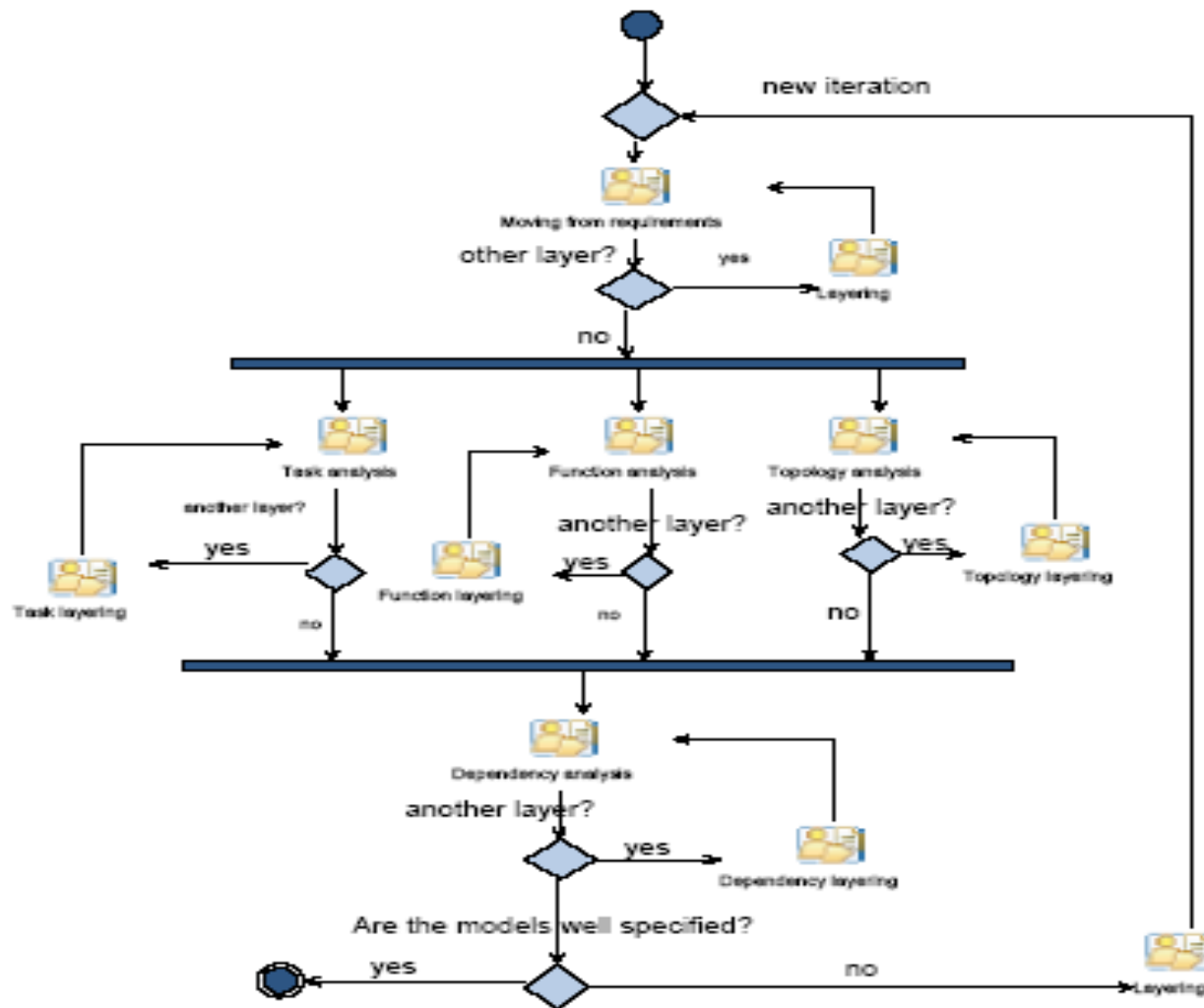
Ci sono le quattro fasi:
Analisi dei requisiti,
Analisi, Design
Architetturale, Detailed
Design.

C'è un processo per ogni
sottofase, in ogni
sottofase viene specificato
cosa deve fare e ogni
sottofase è una attività
modellata in termini di
SPEM.

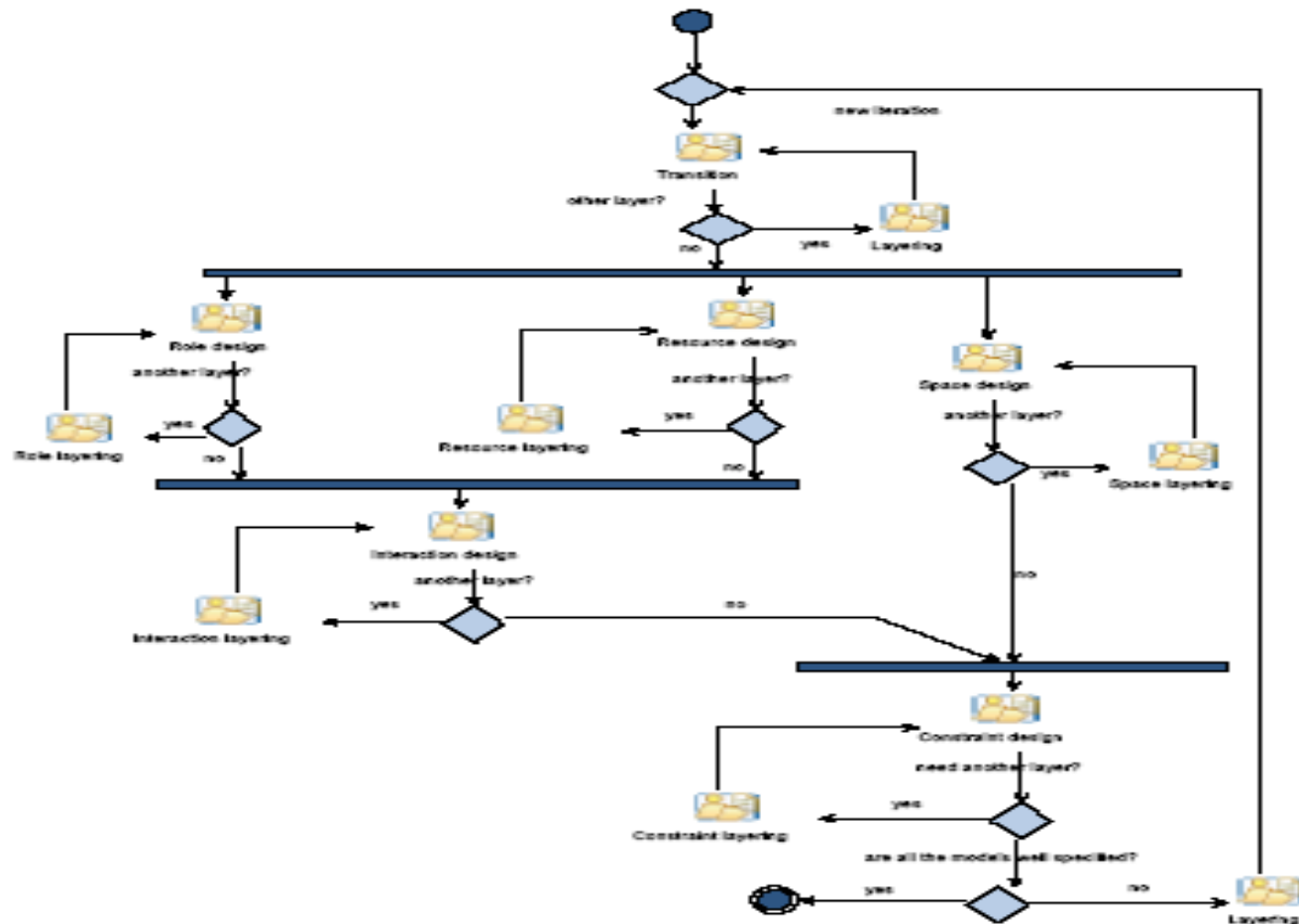
Analisi dei Requisiti



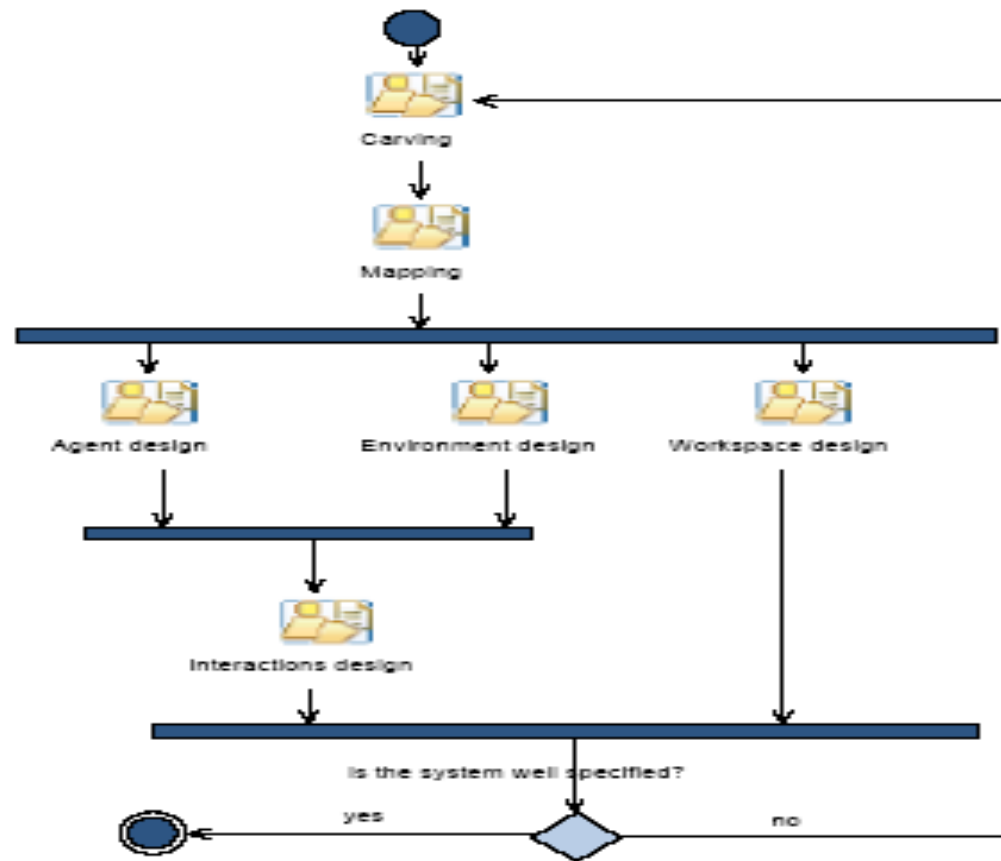
Analisi



Design Architetturale

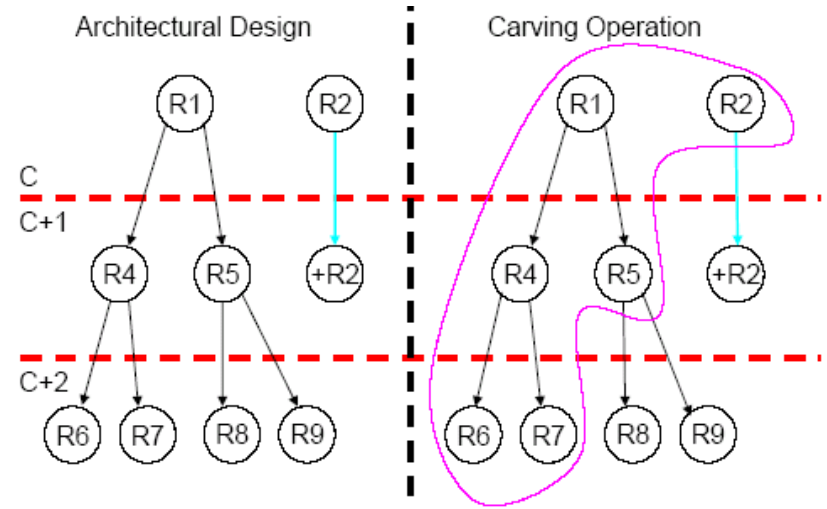


Detailed Design



Carving

- Definito il processo bisogna dire quale dei vari livelli prendere (ci interessa modellare un sistema non tanti)
- Ogni layer che sia completo mi definisce una software architecture, però ovviamente dobbiamo cercare il livello di astrazione più adatta.
- Ogni carvin rappresenta una specifica possibilità di sviluppo.



Notazione di SODA

- La notazione di SODA è tabellare. Tabellare perché:
 - È un chiaro modo per fare l'analisi e la progettazione del sistema
 - E' molto semplice lavorarci per i tool automatici (diventa un prerequisito)
 - È molto semplice per mantenere consistenza e completezza

Tabelle nella fase di Analisi

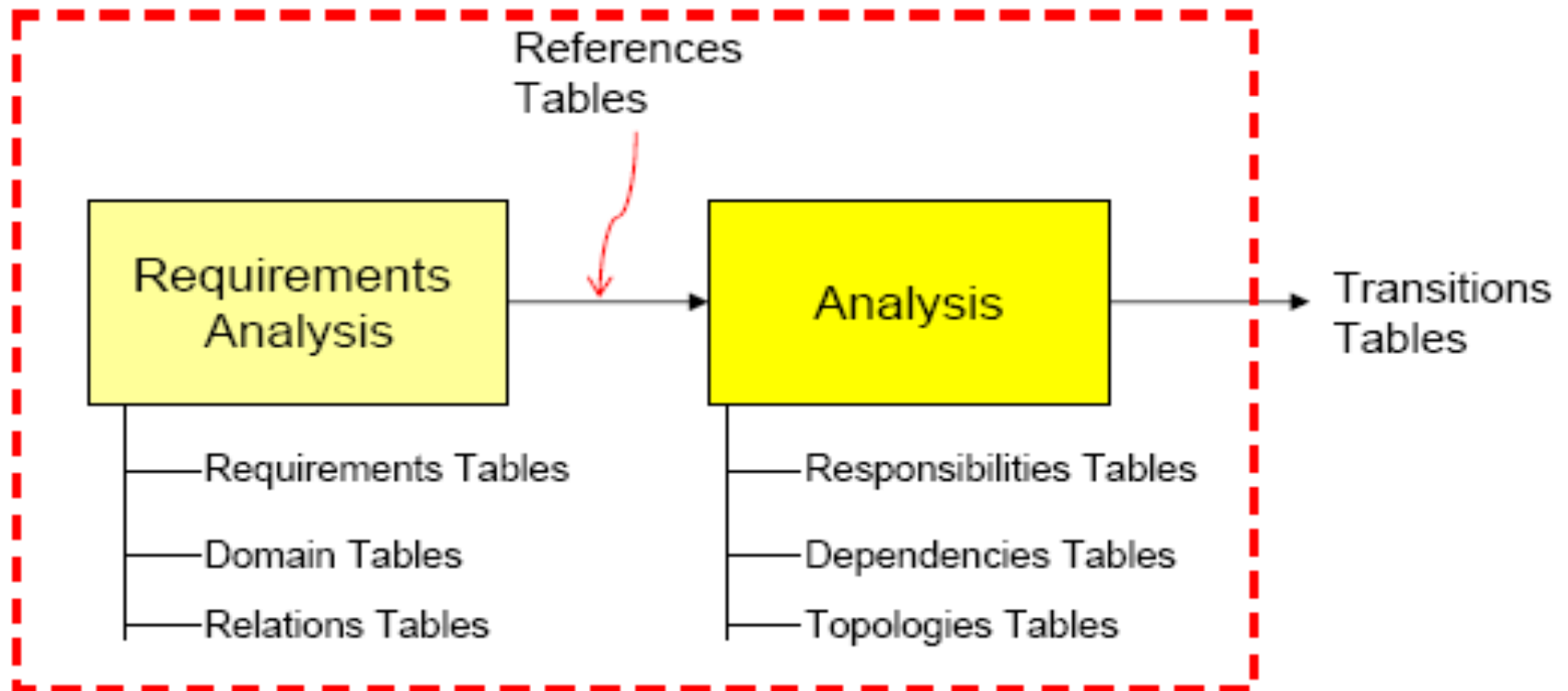
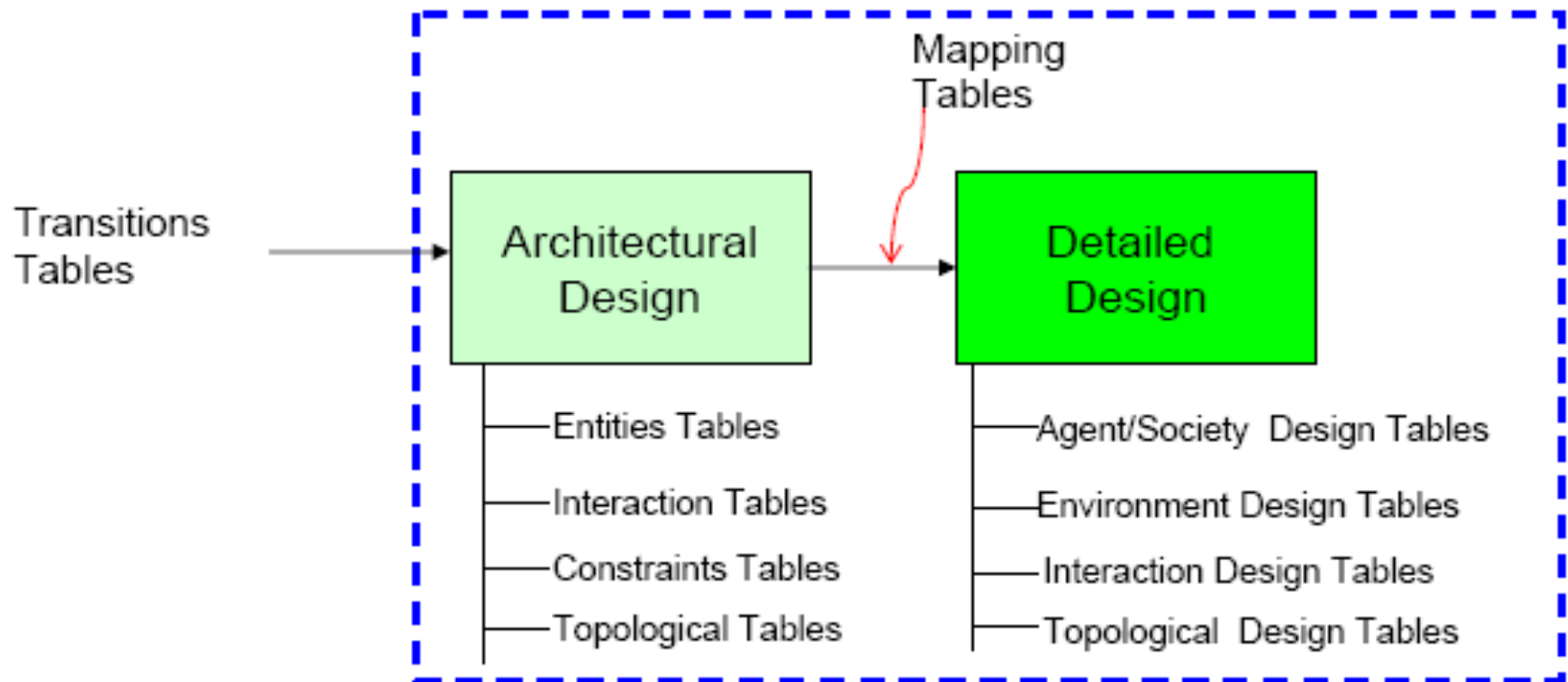
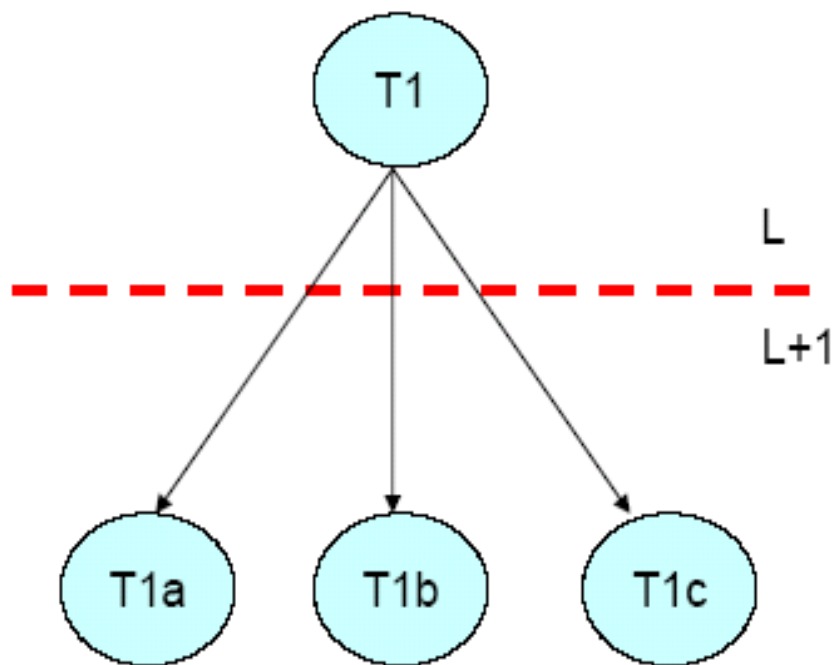


Tabelle nella fase di Design



Notazione Tabellare



- Zooming Table: $(L)Z_t$

Layer L	Layer L+1
$T1$	$T1a, T1b, T1c, \dots$



TROPOS

Tropos

Tropos si caratterizza per tre idee chiave:

- Le nozioni di agente, goal, piani e altre nozioni mentalistiche sono usate nella modellazione concettuale lungo tutte le fasi di sviluppo del software, dalla primissima fase di raccolta dei requisiti fino all'implementazione.
- L'adozione di un approccio allo sviluppo del software guidato dai requisiti anziché dai vincoli dettati dalla piattaforma di implementazione scelta.
- La costruzione di modelli concettuali seguendo un approccio trasformatzionale di tipo incrementale in cui si parte da un modello iniziale dell'ambiente e lo si raffina ed estende incrementalmente fino ad ottenere un artifact eseguibile.

Concetti di Tropos (1/3)

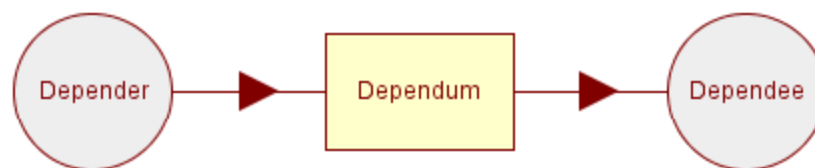
- *Actor*: modella un'entità che possiede obiettivi strategici e intenzionalità all'interno del sistema o dell'organizzazione. Si distinguono tre specializzazioni dell'attore:
 - *Agent*: caratterizzato da delle proprietà come autonomia, reattività, proattività, abilità sociale nel contesto in cui opera.
 - *Role*: caratterizzazione astratta del comportamento di un attore sociale all'interno di un contesto specifico del dominio applicativo
 - *Position*: rappresenta un insieme di ruoli tipicamente ricoperti da un singolo agente.

Concetti di Tropos (2/3)

- *Goal*: rappresenta l'interesse strategico dell'attore che partecipa ad un sistema o ad una organizzazione. Si distingue tra hardgoal e softgoal.
- *Plan*: insieme di azioni che permettono di soddisfare un goal
- *Resource*: è un'entità fisica oppure una informazione (insieme di dati)
- *Capability*: rappresenta l'abilità di un'attore nel definire, scegliere ed eseguire uno o più piani per soddisfare un goal. La capability si attiva in presenza di un evento.
- *Belief*: rappresenta la conoscenza di un attore circa lo stato del mondo.

Concetti di Tropos (3/3)

- Dependency: (relazione più importante della metodologia Tropos) una dipendenza fra due attori indica che un attore dipende dall'altro per soddisfare un goal, eseguire un piano oppure fornire una risorsa. L'attore che delega il proprio obiettivo si chiama depender, mentre colui che riceve la dipendenza è il dependee. L'oggetto della dipendenza è il dependum.



Le fasi di Tropos

Tropos copre cinque fasi del processo di sviluppo del software:

- *Early requirements*: consente di analizzare e modellare i requisiti del contesto laddove il sistema software verrà calato.
- *Late requirements*: si rivolge invece allo studio dei requisiti del sistema software da realizzare.
- *Architectural design e Detailed design*: con l'obiettivo di progettare l'architettura del sistema software.
- *Implementation*: dove si conclude con l'implementazione del codice.



Early requirements

In questa fase l'ingegnere identifica gli stakeholder (entità che hanno un interesse a partecipare in un sistema software) e li modella come attori sociali che dipendono fra loro per goal da soddisfare, piani da eseguire e risorse da fornire.

Il risultato di questa fase è un particolare modello incentrato su attori sociali e loro dipendenze.



Late requirements

Il modello concettuale prodotto precedentemente viene esteso per includere nuovi attori che rappresentano il sistema e dipendenze fra quest'ultimi e quelli dell'ambiente. Tali dipendenze definiscono completamente i requisiti funzionali e non-funzionali.



Architectural design (1/2)

Studio dell'architettura globale del sistema in termini di subsystem interconnessi attraverso dati e flussi di controllo. I subsystem sono rappresentati nel modello con attori di sistema, mentre le interconnessioni con dipendenze.

Architectural design (2/2)

Questa fase si articola in tre passi:

1. Viene definita l'architettura del sistema software. Sono introdotti nuovi attori risultanti dall'analisi nelle fasi precedenti.
2. Identificazione delle capability necessarie agli attori per soddisfare i loro goal ed eseguire i piani. Il risultato è una tabella in cui su un lato sono riportati gli attori e sull'altro le capability loro assegnate. (Si stanno assegnando solo vincoli logici non fisici).
3. Assegnazione degli agenti e delle capability. Tale processo va sotto il nome di agentificazione poiché si definiscono definitivamente quali sono gli attori che verranno implementati nella fase successiva sotto forma di agenti run-time. (Agentificazione non è unica e dipende dal progettista). Anche qui il risultato è una tabella nella quale si elencano gli agenti e le capability assegnate nel passo 2.



Detailed Design

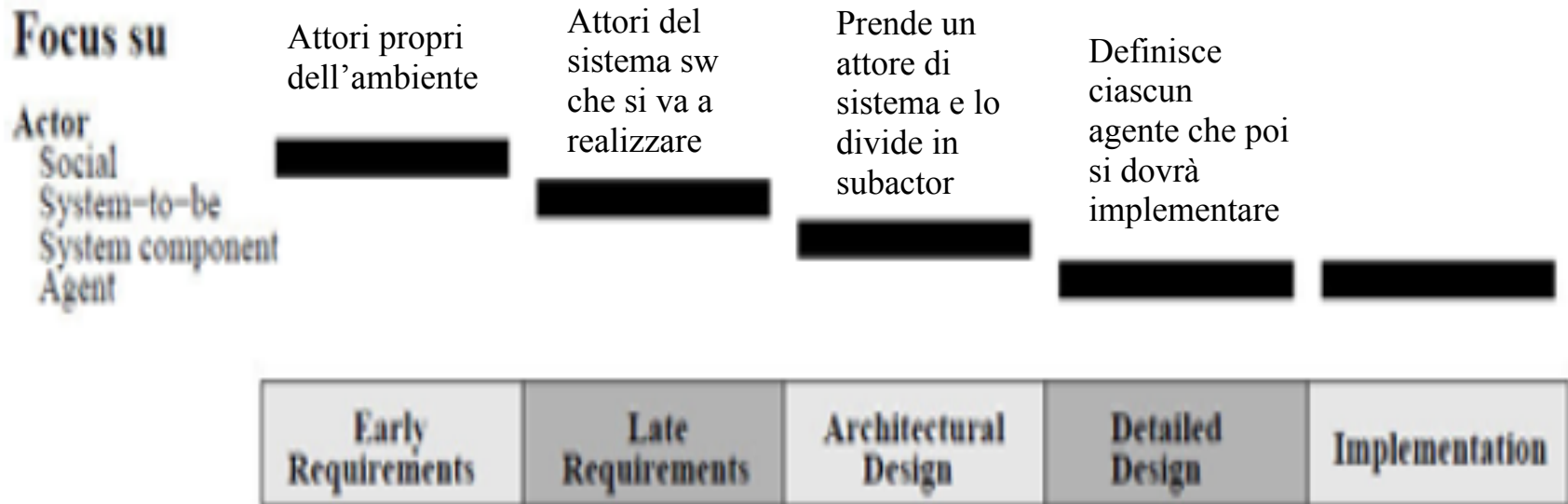
- In questa fase si specifica ciascun agente nel suo interno, le capability, goal, belief, plan e le interazioni che lo coinvolgono.



Implementation

Implementazione del sistema elaborato nella fase precedente, utilizzando ad esempio delle piattaforme di sviluppo AOP (JACK).

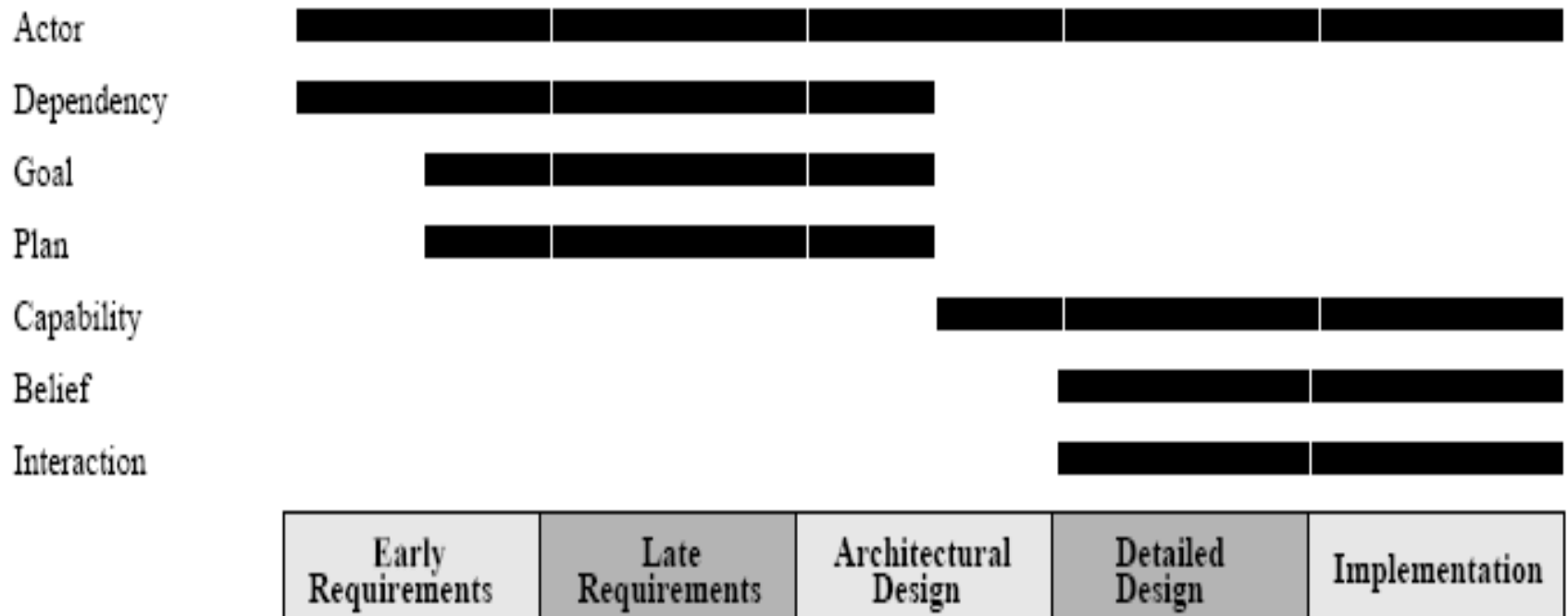
Modellazione degli attori lungo le fasi di sviluppo



Questa figura mostra quali tipologie di attori e agenti sono approfonditi in ciascuna fase della metodologia.

Attività di Modellazione

Attività di modellazione



Questa fig. mostra il ruolo delle attività di modellazione all'interno delle fasi del processo di sviluppo.

Attività di modellazione

- *Actor Modelling*: consiste nell'identificare e analizzare sia gli attori dell'ambiente sia quelli del sistema. (L'imm. slide 56 mostra quali tipologie di attori sono approfonditi in ciascuna fase della metodologia).
- *Dependency modelling*: consiste nell'identificazione degli attori che dipendono da altri per soddisfare goal, per eseguire piani e fornire risorse. Nella fase di Early requirements si modellano le dipendenze fra li attori sociali dell'organizzazione. Nei late requirements si fa lo stesso lavoro della fase precedente solo che ora si discutono gli attori del sistema futuro (system-to-be).

Attività di modellazione

- *Goal Modelling*: è l'attività di analisi dei goal dal punto di vista di un attore usando tre tecniche:
 - Means-ends analysis: punta a identificare goal, piani e risorse che forniscono i mezzi per soddisfare i goal.
 - Contribution: permette di trovare goal che contribuiscono positivamente, o negativamente al soddisfacimento del goal radice. (trovare i sottogoal)
 - AND-OR decomposition: (tecnica dell'AI) consente di costruire grafi AND-OR di subgoal per capire come sia possibile soddisfare la radice

L'attività si esaurisce nell'architectural design quando si decompongono gli attori di sistema in subactor.

Attività di modellazione

- *Plan modelling*: si può considerare un'attività simile a quella condotta sul goal.
- *Capability modelling*: inizia verso la fine dell'architectural design quando i subactor di sistema sono stati specificati in termini dei loro goal e dipendenze con gli altri attori. Ciascun subactor introdotto in questa fase potrebbe essere capace di definire, scegliere ed eseguire un piano per soddisfare i propri goal. Nella fase poi di detailed design i subactor sono ulteriormente specificati.
- *Belief modelling*: consiste nel rappresentare le credenze di ogni agente in riferimento allo stato del mondo che lo circonda. I belief guidano alla scelta dei piani da eseguire e degli agenti stessi da implementare. Attività di detailed Design con agenti specificati a microlivello.
- *Interaction modelling*: specifica il protocollo comunicativo

Linguaggio di modellazione

- Il linguaggio di modellazione visuale utilizzato in Tropos è un linguaggio di specifica semiformale per il quale sono stati definiti:
 - *Ontologia*: insieme di concetti per la modellazione. Gli elementi dell'ontologia di Tropos sono: actor, plan, resource, dependency, capability e bielef.
 - *Meta-modello*: definisce la sintassi del linguaggio. Il meta-modello fornisce uno strumento per controllare che i modelli siano sintatticamente corretti.
 - *Notazione grafica e diagrammi*: stabilisce come raffigurare graficamente gli elementi dei modelli.
 - *Insieme di regole d'uso*: guidano la costruzione di un modello concettuale nelle varie fasi del processo

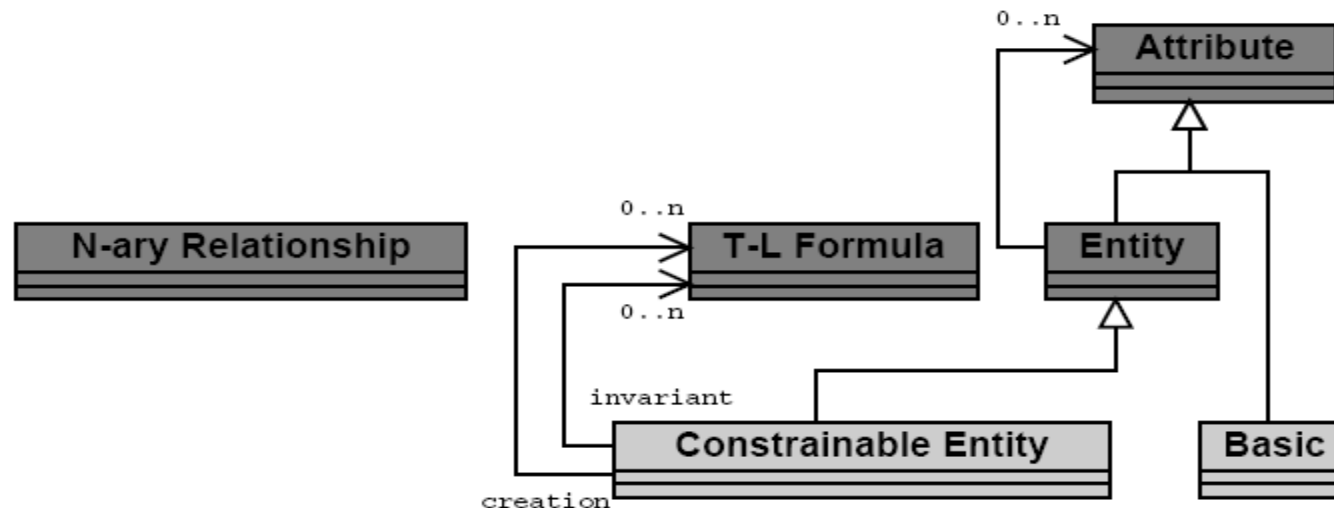
Meta-Modello

- Il meta-modello del linguaggio è definito da una architettura composta da quattro livelli:

Livelli	Descrizione	Esempi
meta-metamodello	Definisce il linguaggio per la specifica del meta-modello.	Attribute, Entity.
meta-modello	Un'istanza del meta-metamodello. È il cuore del linguaggio.	Actor, Goal, Dependency.
dominio	Un'istanza del meta-modello. Definisce un modello del dominio applicativo.	PAT, Citizen: istanze di Actor.
oggetto	Un'istanza di un'entità definita al livello del dominio.	Mary: istanza Citizen.

Meta-metamodello (1/3)

- Il meta-metamodello di tropos è la struttura più astratta dell'intero linguaggio; lo scopo primario è definire gli elementi minimi del linguaggio necessari per la specifica del meta-modello. Costituisce le fondamenta sulle quali costruire il meta-modello che ne è una particolare istanza.

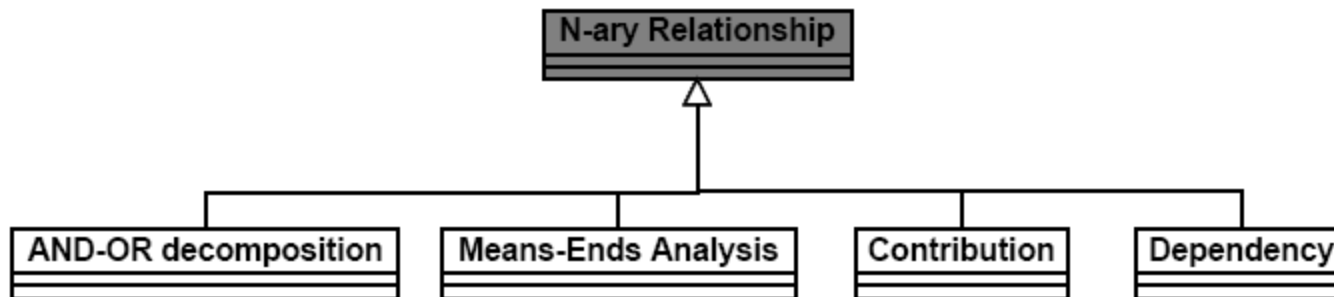


Meta-metamodello (2/3)

- *Entity*: rappresenta gli elementi intenzionali e non intenzionali che esistono nell'ambiente o nell'organizzazione.
- *Attribute*: rappresentano gli attributi che saranno legati a ogni istanza di Entity.
- *Basic*: altri attributi.
- *Constrainable Entity*: Entity è la super-classe di constrainable Entity. Ogni sua istanza è definita da un insieme di attributi e proprietà. I suoi attributi sono ereditati da una parte dalla super classe Entity, dall'altra dalla classe T-L Formula. Le proprietà fissano delle regole che l'istanza della classe Constrainable entity deve soddisfare al momento della creazione e/o durante il ciclo di vita.

Meta-metamodello (1/3)

- *R-ary Relationship*: raggruppa tutte le relazioni possibili che si possono definire sulle istanze di Entity a livello di dominio

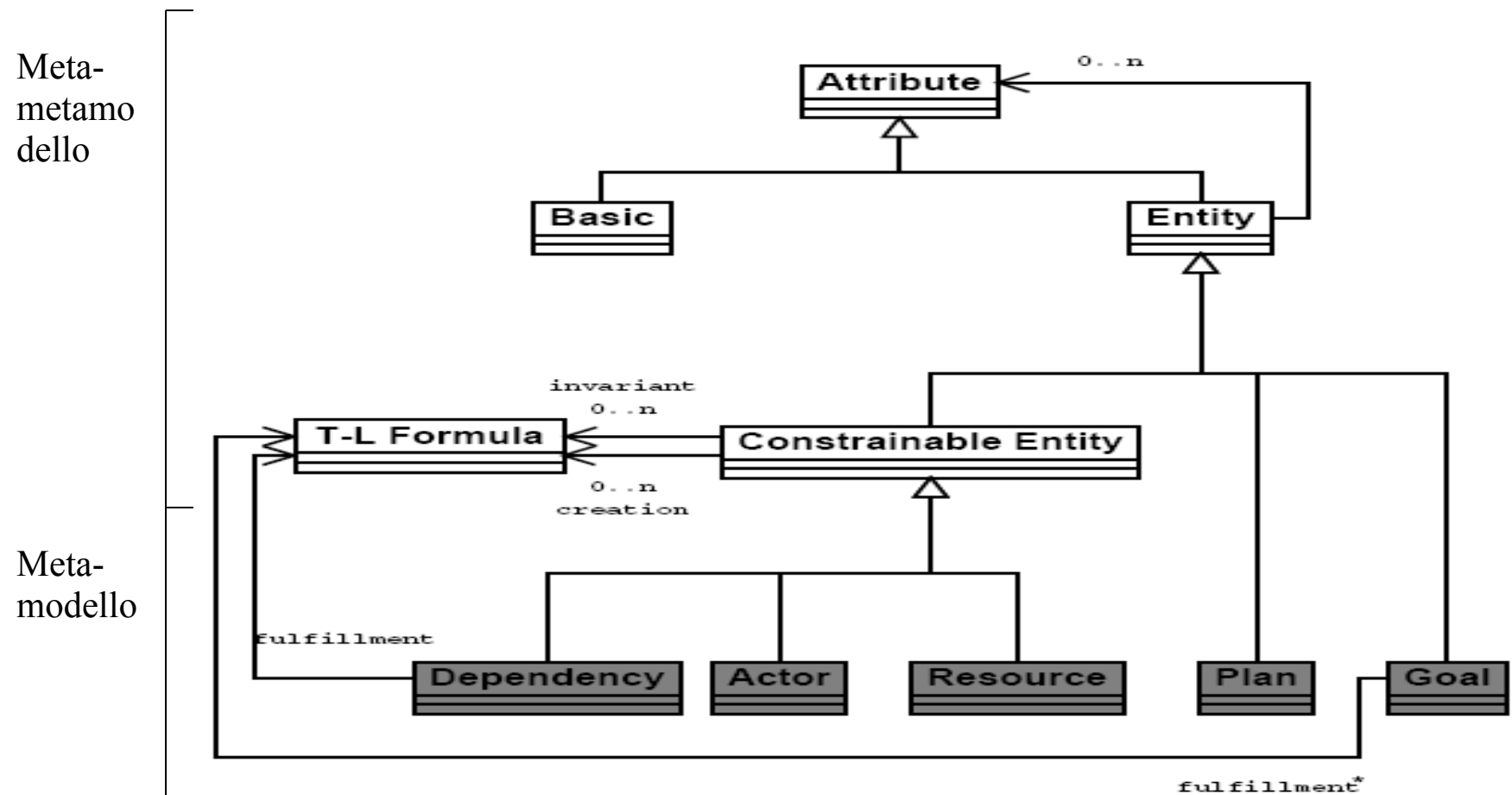




Meta-Modello (1/5)

- Il meta-modello di Tropos è un livello logico che si focalizza sui concetti ontologici essenziali e indipendenti dal dominio applicativo.
- Il meta-modello è un'istanza del meta-metamodello e al suo interno troviamo meta-classi che modellano gli elementi essenziali: actor, goal, plan, resource e dependency.

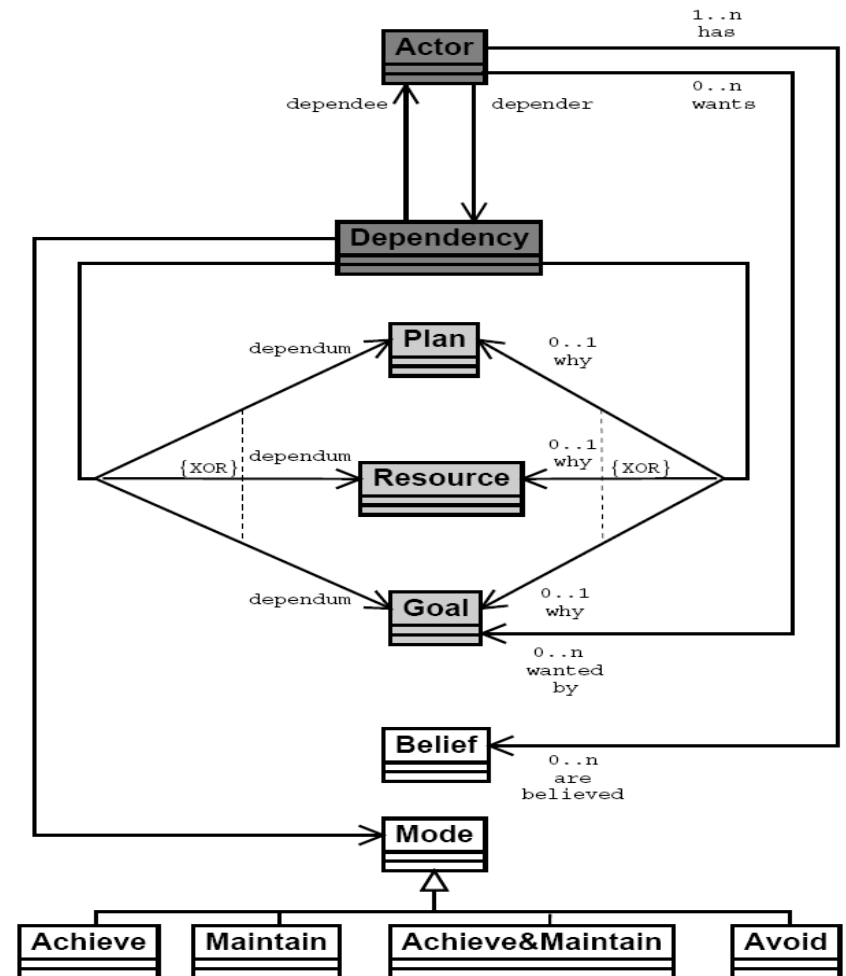
Meta-Modello (2/5)



Fulfillment: in questo modo è possibile descrivere formalmente quando un goal è soddisfatto o meno.

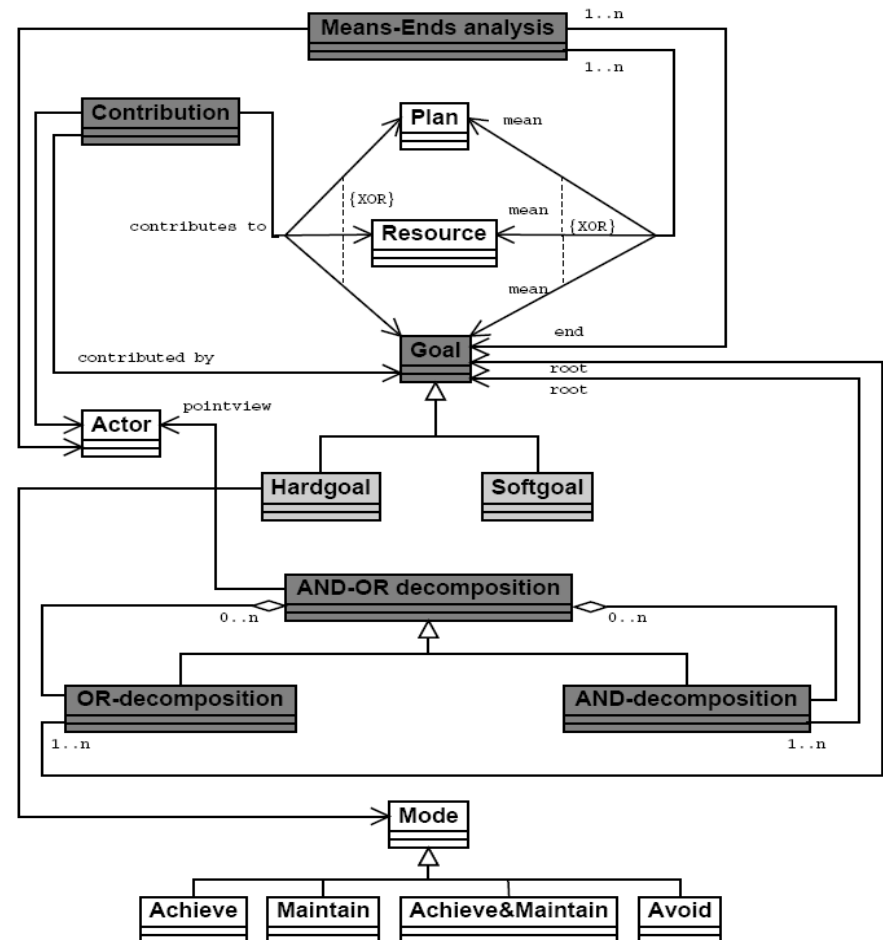
Meta-Modello (3/5)

- *Actor*: è un'entità che possiede degli obiettivi strategici all'interno del sistema oppure dell'organizzazione che ospita il sistema software stesso; inoltre è sorgente di intenzionalità



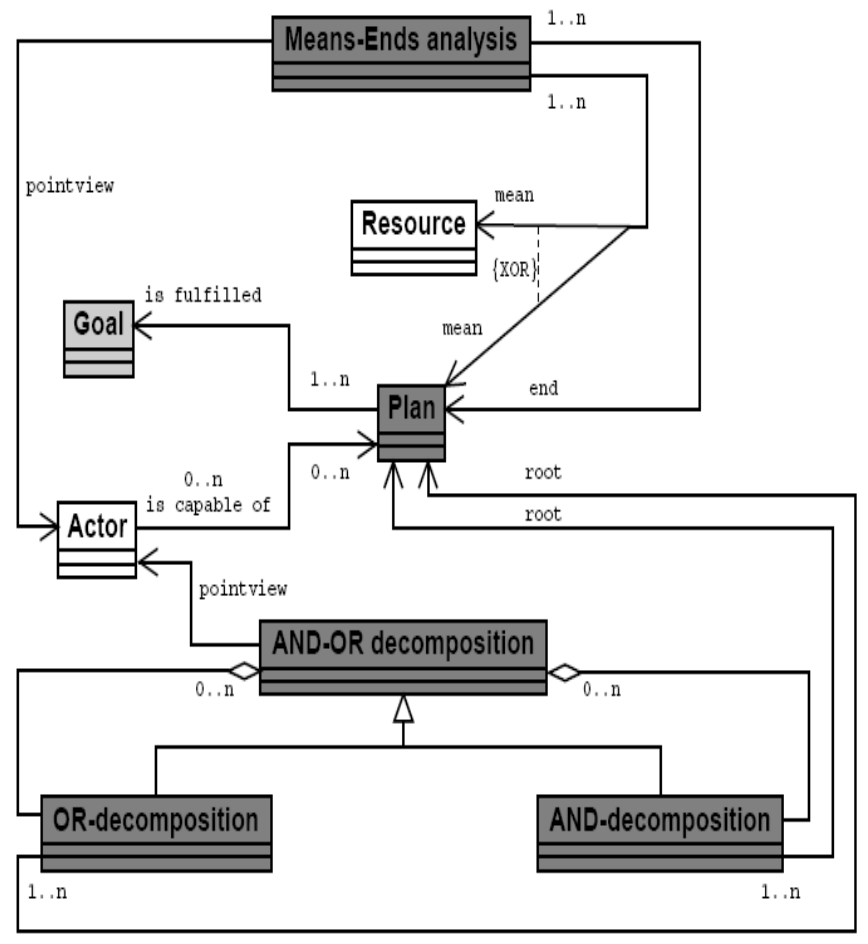
Meta-Modelo (4/5)

- Il *goal* modella gli obiettivi che un attore vuole raggiungere. Può essere specializzata in hard e softgoal.



Meta-Modello (5/5)

- Il *piano* modella cosa è possibile fare per soddisfare un goal, una volta scelta l'opportuna strategia. Un attore è capace di scegliere, definire ed eseguire dei piani.



Il Livello di Dominio

- Il livello del dominio di Tropos è un'istanza del meta-modello dipendente dal dominio applicativo. Tutti i suoi elementi sono istanze di classi del meta-modello.

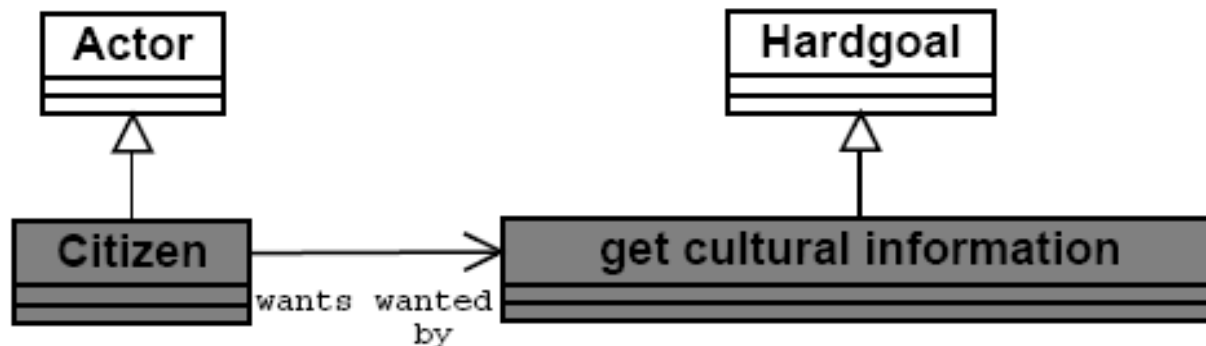
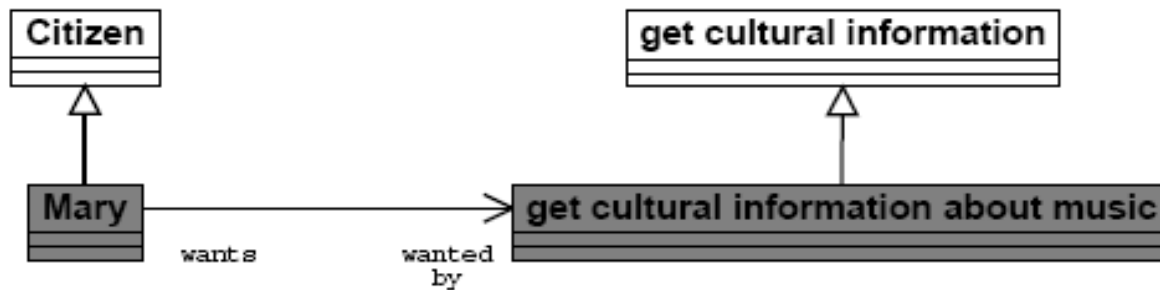


Diagramma delle classi UML che specifica il *livello del dominio* di Tropos. L'attore Citizen vuole l'hardgoal get cultural information.

Il Livello Oggetto

Il livello oggetto di Tropos è un'istanza del dominio: lo scopo è scendere ancora nel livello di astrazione e nei dettagli per raffinare il modello.



Notazione di Tropos

- Tropos usa una notazione grafica e si avvale dell'uso di cinque diagrammi che catturano sia aspetti statici che dinamici del modello:
 - Actor Diagram
 - Goal Diagram
 - Plan Diagram
 - Capability Diagram
 - Agent Interaction Diagram



Figura 2.1: Rappresentazione grafica di Attore, Softgoal, Piano e Risorsa.

Actor Diagram (1/2)

- Il diagramma degli attori è una rappresentazione grafica del modello dove sono messi in evidenza gli attori, i loro goal e la rete di dipendenze fra attori; enfatizza gli aspetti statici del modello.
- L'actor diagram si presta ad accompagnare l'analista lungo tutte le fasi del processo di sviluppo.
- Risulta utile soprattutto in fase di progettazione dell'architettura logica e fisica, laddove si individuano e assegnano le capability ai vari attori.

Actor Diagram (2/2)

Il diagramma degli attori ha tre obiettivi strategici:

1. Raffigurare gli attori dell'ambiente e del sistema con relativi goal e dipendenze a seconda della fase in cui si lavora.
2. Mantenere traccia ed eventualmente integrare con nuovi attori, dipendenze e goal usciti da raffinamenti condotti nel diagramma dei goal che vedremo tra poco.
3. Aiutare il progettista ad identificare le capability per ciascun attore. (architectural design)

Goal Diagram (1/2)

- Il diagramma dei goal è il luogo dove si analizzano in profondità i goal e le dipendenze mediante le tecniche usuali di Means-Ends analysis, AND-OR decomposition e Contribution.
- Utilizzato per poter decomporre i goal in subgoal, individuare soluzioni alternative che soddisfino il goal iniziale. (visione microscopica del dominio applicativo)
- In output fornisce tipicamente nuove dipendenze fra attori oppure piani e risorse che consentono di soddisfare il goal iniziale.

Goal Diagram (2/2)

Il Goal diagram è utile nei seguenti casi:

- Prodotto il diagramma degli attori, si catturano dipendenze e goal per poi analizzarli dal punto di vista dell'attore mediante le usuali tecniche di Means-Ends analysis, Contribution e AND-OR decomposition.
- Nell'architectural design il diagramma dei goal è utile per arricchire quello degli attori in nuovi attori e dipendenze che partecipano all'individuazione delle capability.

Capability diagram (1/2)

- Il diagramma delle abilità è un diagramma delle attività AUMML che permette di modellare una capability (o un insieme di capability) dal punto di vista di uno specifico attore.
- E' previsto solo nella fase di detailed design quando è già stata introdotto la piattaforma architettuale.
- In sostanza, il capability diagram è assai utile quando si desidera approfondire gli aspetti dinamici legati ad una capability. Si possono rappresentare gli stati interni, come gli eventi vengono attivati, quali piani sono eseguiti e quali belief sono richiesti.

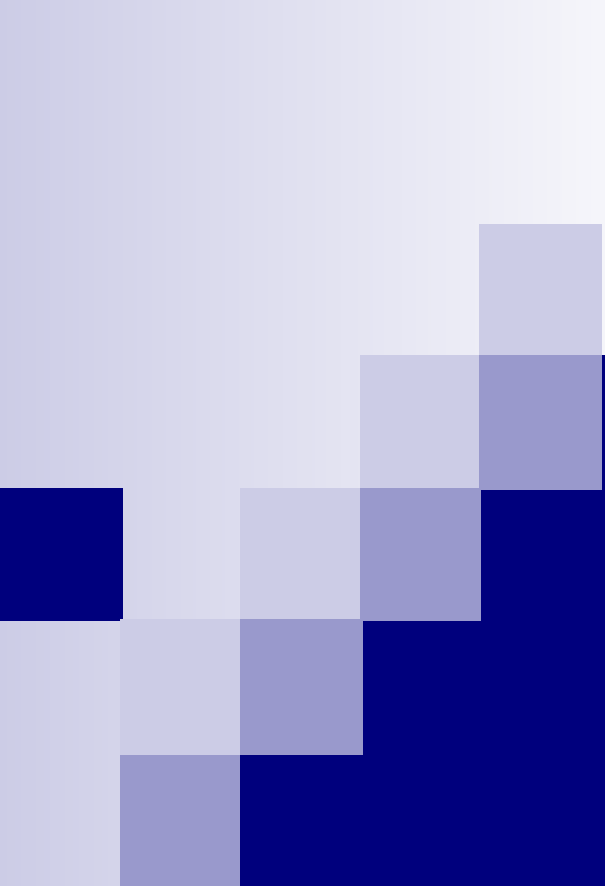
Plan Diagram

- Il diagramm dei piani è un diagramma delle attività AUMML che analizza gli stati interni di un piano contenuto in una capability.
- Il piano descrive cosa un agente deve fare quando occorre un evento interno o esterno; in particolare, un piano descrive una sequenza di azioni che lo conducono a soddisfare l'evento che lo ha lanciato.
- Una volta costruito il Capability diagram si devono specificare i piani che lo compongono. Si tratta di studiare a basso livello (siamo alla fase immediatamente prima dell'implementazione) quali sono le azioni, le decisioni logiche e i costrutti necessari per realizzare un determinato scopo.



Agent Interaction Diagram

- Il diagramma delle interazioni tra agenti è un diagramma di sequenza AUML nel quale la linea di vita di ciascun agente è indipendente dalla specifica interazione che deve essere modellata. Le azioni comunicative fra i vari agenti corrispondono a messaggi asincroni.
- Il diagramma delle interazioni tra agenti è usato per specificare i messaggi che gli agenti possono scambiare. E' molto utile per determinare le loro interazioni e l'ordine di spedizione dei messaggi stessi.



CONCLUSIONI

SODA

- Si focalizza sulla modellazione delle società di agenti
- Si focalizza sulla descrizione/percezione dell'ambiente che circonda il sistema
- Introduce il meta-modello ontologico Agent&Artifacts: che si basa di modellare i sistemi non solamente tramite il concetto di agente, ma usa anche la nozione di Artefatto per modellare esplicitamente l'ambiente.
- SODA mette a disposizione un meccanismo di zoom per modellare un sistema a diversi livelli di astrazione. A questo meccanismo si associa il meccanismo di proiezione che proietta da un livello al successivo le entità eventualmente non soggette a zoom per preservarne la consistenza.
- SODA intenzionalmente e diversamente dagli altri approcci non si interessa degli aspetti relativi alla struttura interna dei singoli agenti.

Tropos

- E' una metodologia fortemente orientata all'analisi dei requisiti del sistema, basata sui concetti di attori, dipendenze tra attori e dipendenze dalle risorse.
- La metodologia Tropos intende offrire un approccio strutturato allo sviluppo del software basato sulla creazione di modelli concettuali, definiti a partire da un linguaggio visuale utilizza alcuni concetti base quali: agent goal belief plan e interaction.
- Tropos usa la metafora dell'organizzazione umana, di conseguenza i modelli della organizzazione umana e le strutture di essa sono impiegate per progettare MAS, concetti come ruolo dipendenza sociale, e compiti di organizzazioni sono usati per non modellare l'ambiente nel quale il sistema lavora ma il sistema stesso.

Tropos

- TROPOS deve questo alle tre idee chiave:
 - l'impiego delle nozioni mentali di agente, piani, obiettivi etc. in tutte le fasi dello sviluppo del software;
 - Un approccio allo sviluppo del software guidato dai requisiti piuttosto che dai vincoli della particolare piattaforma di implementazione scelta;
 - la costruzione di modelli concettuali seguendo un approccio trasformatzionale di tipo incrementale.

Confronto tra le metodologie

	Analisi dei requisiti	Modellazione struttura sociale	Modellazione ambiente	Gestione della complessità	
	Gaia	Buona	Buona	Quasi assente	Assente
➡	Tropos	Buona	Da migliorare	Assente	Assente
	PASSI	Buona	Buona	Assente	Assente
➡	SODA	Buona	Buona	Buona	Buona

Confronto tra le metodologie

- La modellazione della struttura sociale viene trattata ampiamente da tutte le due metodologie sin dalle prime fasi di modellazione.
- La modellazione dell'ambiente costituisce un aspetto peculiare di SODA, al contrario di Tropos che non considera l'ambiente del sistema come una entità da modellare.
- Tropos è inadeguato a modellare i sistemi in cui sono presenti altre entità oltre agli attori e oltre al sistema stesso in quanto nella fase iniziale di analisi di requisiti considera gli attori sociali e le loro relative relazioni ma non il sistema.
- SODA sembra essere anche la sola metodologia a essersi posto il problema di disporre di adeguati strumenti di supporto della gestione della complessità introducendo un meccanismo di zooming e di proiezione.